```
001 // Railgun_Swampshine_BailOut, copyleft 2014-Apr-27, Kaze.
002 // 2014-Apr-27: The nasty SIGNED/UNSIGNED bug in 'Swampshines' which I illustrated several months ago in my fuzzy search article now is fixed here too:
003 /*
004 The bug is this (the variables 'i' and 'PRIMALposition' are uint32_t):
005 Next line assumes -19 >= 0 is true:
006 if ( (i-(PRIMALposition-1)) >= 0) printf ("THE NASTY BUG AGAIN: %d >= 0\n", i-(PRIMALposition-1));
007 Next line assumes -19 >= 0 is false:
008 if ( (signed int)(i-(PRIMALposition-1)) >= 0) printf ("THE NASTY BUG AGAIN: %d >= 0\n", i-(PRIMALposition-1));
009 And the actual fix:
010 ...
011 if ( count <= 0 ) {
012 // I have to add out-of-range checks...
013 // i-(PRIMALposition-1) >= 0
014 // &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
015 // i-(PRIMALposition-1)+(count-1) >= 0
016 // &pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4
017 // FIX from 2014-Apr-27:
018 // Because (count-1) is negative, above fours are reduced to next twos:
019 // i-(PRIMALposition-1)+(count-1) >= 0
020 // &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
021 // The line below is BUGGY:
022 //if ( (i-(PRIMALposition-1) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) && (&pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4) ) {
023 // The line below is OKAY:
024 if ( ((signed int)(i-(PRIMALposition-1)+(count-1)) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) ) {
025 ...
026 */
027 // Railgun_Swampshine_BailOut, copyleft 2014-Jan-31, Kaze.
028 // Caution: For better speed the case 'if (cbPattern==1)' was removed, so Pattern must be longer than 1 char.
029 #define NeedleThreshold2vs4swampLITE 9+10 // Should be bigger than 9. BMH2 works up to this value (inclusive), if bigger then BMH4 takes over.
030 char * Railgun_Swampshine_BailOut (char * pbTarget, char * pbPattern, uint32_t cbTarget, uint32_t cbPattern)
031 {
032 char * pbTargetMax = pbTarget + cbTarget;
033 register uint32_t ulHashPattern;
034 signed long count;
035
036 unsigned char bm_Horspool_Order2[256*256]; // Bitwise soon...
037 uint32_t i, Gulliver;
038
039 uint32_t PRIMALposition, PRIMALpositionCANDIDATE;
040 uint32_t PRIMALlength, PRIMALlengthCANDIDATE;
041 uint32_t j, FoundAtPosition;
042
043 if (cbPattern > cbTarget) return(NULL);
044
045 if ( cbPattern<4 ) {
046       // SSE2 i.e. 128bit Assembly rules here:
047       // ...
048                 pbTarget = pbTarget+cbPattern;
049       ulHashPattern = ( (*(char *)(pbPattern))<<8 ) + *(pbPattern+(cbPattern-1));
050       if ( cbPattern==3 ) {
051                 for ( ;; ) {
052                           if ( ulHashPattern == ( (*(char *)(pbTarget-3))<<8 ) + *(pbTarget-1) ) {
053                                     if ( *(char *)(pbPattern+1) == *(char *)(pbTarget-2) ) return((pbTarget-3));
054                           }
055                           if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) {
056                                     pbTarget++;
057                                     if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) pbTarget++;
058                           }
059                           pbTarget++;
060                           if (pbTarget > pbTargetMax) return(NULL);
061                 }
```

Railgun-Swampshine

- Fast & Strong 32bit MEMMEM
- Superfast for Needles up to 128 bytes
- Troll-proof i.e. strong in worst-case scenarios
- Derived from Hyperfast Railgun-Ennearch
- Pseudo BMH order 2 main loop code size: 218 bytes
- 64KB Pseudo BMH order 2/4
- Function size: 2720 bytes
- Swampwalker-BailOut heuristic
- Written in pure C by Kaze
- 100% FREE

```
062       } else {
063       }
064       for ( ;; ) {
065                 if ( ulHashPattern == ( (*(char *)(pbTarget-2))<<8 ) + *(pbTarget-1) ) return((pbTarget-2));
066                 if ( (char)(ulHashPattern>>8) != *(pbTarget-1) ) pbTarget++;
067                 pbTarget++;
068                 if (pbTarget > pbTargetMax) return(NULL);
069       }
070 } else { //if ( cbPattern<4 )
071       if ( cbPattern<=NeedleThreshold2vs4swampLITE ) {
072                 // BMH order 2, needle should be >=4:
073                 ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
074                 for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
075                 for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=1;
076                 i=0;
077                 while (i <= cbTarget-cbPattern) {
078                           Gulliver = 1; // 'Gulliver' is the skip
079                           if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]] != 0 ) {
080                                     if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1-2]] == 0 ) Gulliver = cbPattern-(2-1)-2; else {
081                                               if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
082                                                         count = cbPattern-4+1;
083                                                         while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
084                                                                   count = count-4;
085                                                         if ( count <= 0 ) return(pbTarget+i);
086                                               }
087                                     }
088                           } else Gulliver = cbPattern-(2-1);
089                           i = i + Gulliver;
090                           //GlobalI++; // Comment it, it is only for stats.
091                 }
092                 return(NULL);
093       } else { // if ( cbPattern<=NeedleThreshold2vs4swampLITE )
094
095 // Swampwalker_BAILOUT heuristic order 4 (Needle should be bigger than 4) [
096 // Needle: 1234567890qwertyuiopasdfghjklzxcv            PRIMALposition=01 PRIMALength=33  '1234567890qwertyuiopasdfghjklzxcv'
097 // Needle: vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv            PRIMALposition=29 PRIMALength=04  'vvvv'
098 // Needle: vvvvvvvvvvBOOMSHAKALAKAvvvvvvvvvv          PRIMALposition=08 PRIMALength=20  'vvvBOOMSHAKALAKAvvvv'
099 // Needle: Trollland                                 PRIMALposition=01 PRIMALength=09  'Trollland'
100 // Needle: Swampwalker                               PRIMALposition=01 PRIMALength=11  'Swampwalker'
101 // Needle: licenselessness                           PRIMALposition=01 PRIMALength=15  'licenselessness'
102 // Needle: alfalfa                                   PRIMALposition=02 PRIMALength=06  'lfalfa'
103 // Needle: Sandokan                                  PRIMALposition=01 PRIMALength=08  'Sandokan'
104 // Needle: shazamish                                 PRIMALposition=01 PRIMALength=09  'shazamish'
105 // Needle: Simplicius Simplicissimus                 PRIMALposition=06 PRIMALength=20  'icius Simplicissimus'
106 // Needle: domilliaquadringenquattuorquinquagintillion  PRIMALposition=01 PRIMALength=32  'domilliaquadringenquattuorquinqu'
107 // Needle: boom-boom                                 PRIMALposition=02 PRIMALength=08  'oom-boom'
108 // Needle: vvvvv                                     PRIMALposition=01 PRIMALength=04  'vvvv'
109 // Needle: 12345                                     PRIMALposition=01 PRIMALength=05  '12345'
110 // Needle: likey-likey                               PRIMALposition=03 PRIMALength=09  'key-likey'
111 // Needle: BOOOOOM                                   PRIMALposition=03 PRIMALength=05  'OOOOM'
112 // Needle: aaaaaBOOOOOM                              PRIMALposition=02 PRIMALength=09  'aaaaBOOOO'
113 // Needle: BOOOOOMaaaaa                              PRIMALposition=03 PRIMALength=09  'OOOOMaaaa'
114 PRIMALength=0;
115 for (i=0+(1); i < cbPattern-((4)-1)+(1)-(1); i++) { // -(1) because the last BB order 4 has no counterpart(s)
116 FoundAtPosition = cbPattern - ((4)-1) + 1;
117 PRIMALpositionCANDIDATE=i;
118 while ( PRIMALpositionCANDIDATE <= (FoundAtPosition-1) ) {
119       j = PRIMALpositionCANDIDATE + 1;
120       while ( j <= (FoundAtPosition-1) ) {
121                 if ( *(uint32_t *)(pbPattern+PRIMALpositionCANDIDATE-(1)) == *(uint32_t *)(pbPattern+j-(1)) ) FoundAtPosition = j;
122                 j++;
```

```
123       }
124       PRIMALpositionCANDIDATE++;
125 }
126 PRIMALlengthCANDIDATE = (FoundAtPosition-1)-i+1 +((4)-1);
127 if (PRIMALlengthCANDIDATE >= PRIMALlength) {PRIMALposition=i; PRIMALlength = PRIMALlengthCANDIDATE;}
128 if (cbPattern-i+1 <= PRIMALength) break;
129 if (PRIMALlength > 128) break; // Bail Out for 129[+]
130 }
131 // Swampwalker_BAILOUT heuristic order 4 (Needle should be bigger than 4) ]
132
133 // Here we have 4 or bigger NewNeedle, apply order 2 for pbPattern[i+(PRIMALposition-1)] with length 'PRIMALength' and compare the pbPattern[i] with length 'cbPattern':
134 PRIMALlengthCANDIDATE = cbPattern;
135 cbPattern = PRIMALength;
136 pbPattern = pbPattern + (PRIMALposition-1);
137
138 // Revision 2 commented section [
139 /*
140 if (cbPattern-1 <= 255) {
141 // BMH Order 2 [
142                 ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
143                 for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]= cbPattern-1;} // cbPattern-(Order-1) for Horspool; 'memset' if not optimized
144                 for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=i; // Rightmost appearance/position is needed
145                 i=0;
146                 while (i <= cbTarget-cbPattern) {
147                           Gulliver = bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]];
148                           if ( Gulliver != cbPattern-1 ) { // CASE #2: if equal means the pair (char order 2) is not found i.e. Gulliver remains intact, skip the whole pattern and fall back (Order-1) chars i.e. one char for Order 2
149                           if ( Gulliver == cbPattern-2 ) { // CASE #1: means the pair (char order 2) is found
150                                     if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) {
151                                               count = cbPattern-4+1;
152                                               while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
153                                                         count = count-4;
154 // If we miss to hit then no need to compare the original: Needle
155 if ( count <= 0 ) {
156 // I have to add out-of-range checks...
157 // i-(PRIMALposition-1) >= 0
158 // &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
159 // i-(PRIMALposition-1)+(count-1) >= 0
160 // &pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4
161
162 // FIX from 2014-Apr-27:
163 // Because (count-1) is negative, above fours are reduced to next twos:
164 // i-(PRIMALposition-1)+(count-1) >= 0
165 // &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
166 // The line below is BUGGY:
167 //if ( (i-(PRIMALposition-1) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) && (&pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4) ) {
168 // The line below is OKAY:
169 if ( ((signed int)(i-(PRIMALposition-1)+(count-1)) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) ) {
170
171       if ( *(uint32_t *)&pbTarget[i-(PRIMALposition-1)] == *(uint32_t *)(pbPattern-(PRIMALposition-1))) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
172                 count = PRIMALlengthCANDIDATE-4+1;
173                 while ( count > 0 && *(uint32_t *)(pbPattern-(PRIMALposition-1)+count-1) == *(uint32_t *)(&pbTarget[i-(PRIMALposition-1)]+(count-1)) )
174                           count = count-4;
175                 if ( count <= 0 ) return(pbTarget+i-(PRIMALposition-1));
176       }
177 }
178 }
179                                     }
180                                     Gulliver = 1;
181                           } else
182                                     Gulliver = cbPattern - Gulliver - 2; // CASE #3: the pair is found and not as suffix i.e. rightmost position
183                           }
```

```
                        i = i + Gulliver;
                        //GlobalI++; // Comment it, it is only for stats.
                }
                return(NULL);
// BMH Order 2 ]
} else {
                // BMH order 2, needle should be >=4:
                ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
                for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
                for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=1;
                i=0;
                while (i <= cbTarget-cbPattern) {
                        Gulliver = 1; // 'Gulliver' is the skip
                        if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]] != 0 ) {
                                if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1-2]] == 0 ) Gulliver = cbPattern-(2-1)-2; else {
                                        if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                                                count = cbPattern-4+1;
                                                while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                        count = count-4;
// If we miss to hit then no need to compare the original: Needle
if ( count <= 0 ) {
// I have to add out-of-range checks...
// i-(PRIMALposition-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4

// FIX from 2014-Apr-27:
// Because (count-1) is negative, above fours are reduced to next twos:
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// The line below is BUGGY:
//if ( (i-(PRIMALposition-1) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) && (&pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4) ) {
// The line below is OKAY:
if ( ((signed int)(i-(PRIMALposition-1)+(count-1)) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) ) {

        if ( *(uint32_t *)&pbTarget[i-(PRIMALposition-1)] == *(uint32_t *)(pbPattern-(PRIMALposition-1))) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                count = PRIMALlengthCANDIDATE-4+1;
                while ( count > 0 && *(uint32_t *)(pbPattern-(PRIMALposition-1)+count-1) == *(uint32_t *)(&pbTarget[i-(PRIMALposition-1)]+(count-1)) )
                        count = count-4;
                if ( count <= 0 ) return(pbTarget+i-(PRIMALposition-1));
        }
}
}
                                                }
                                        }
                                } else Gulliver = cbPattern-(2-1);
                                i = i + Gulliver;
                                //GlobalI++; // Comment it, it is only for stats.
                }
                return(NULL);
}
*/
// Revision 2 commented section ]

        if ( cbPattern<=NeedleThreshold2vs4swampLITE ) {

                // BMH order 2, needle should be >=4:
                ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
                for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
                for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=1;
```

```
                i=0;
                while (i <= cbTarget-cbPattern) {
                        Gulliver = 1; // 'Gulliver' is the skip
                        if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]] != 0 ) {
                                if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1-2]] == 0 ) Gulliver = cbPattern-(2-1)-2; else {
                                        if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                                                count = cbPattern-4+1;
                                                while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                        count = count-4;
// If we miss to hit then no need to compare the original: Needle
if ( count <= 0 ) {
// I have to add out-of-range checks...
// i-(PRIMALposition-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4

// FIX from 2014-Apr-27:
// Because (count-1) is negative, above fours are reduced to next twos:
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// The line below is BUGGY:
//if ( (i-(PRIMALposition-1) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) && (&pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4) ) {
// The line below is OKAY:
if ( ((signed int)(i-(PRIMALposition-1)+(count-1)) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) ) {

      if ( *(uint32_t *)&pbTarget[i-(PRIMALposition-1)] == *(uint32_t *)(pbPattern-(PRIMALposition-1)) ) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                count = PRIMALlengthCANDIDATE-4+1;
                while ( count > 0 && *(uint32_t *)(pbPattern-(PRIMALposition-1)+count-1) == *(uint32_t *)(&pbTarget[i-(PRIMALposition-1)]+(count-1)) )
                        count = count-4;
                if ( count <= 0 ) return(pbTarget+i-(PRIMALposition-1));
      }
}
}
                                        }
                                }
                        } else Gulliver = cbPattern-(2-1);
                        i = i + Gulliver;
                        //GlobalI++; // Comment it, it is only for stats.
                }
                return(NULL);

      } else { // if ( cbPattern<=NeedleThreshold2vs4swampLITE )

                // BMH pseudo-order 4, needle should be >=8+2:
                ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
                for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
                // In line below we "hash" 4bytes to 2bytes i.e. 16bit table, how to compute TOTAL number of BBs, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest fox' and Order=4 we have BBs = 11-4+1=8:
                //"fast"
                //"aste"
                //"stes"
                //"test"
                //"est "
                //"st f"
                //"t fo"
                //" fox"
                //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( *(unsigned short *)(pbPattern+i+0) + *(unsigned short *)(pbPattern+i+2) ) & ( (1<<16)-1 )]=1;
                //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>16)+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
                // Above line is replaced by next one with better hashing:
                for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>(16-1))+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
```

```
                i=0;
                while (i <= cbTarget-cbPattern) {
                        Gulliver = 1;
                        //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                        // Above line is replaced by next one with better hashing:
                        if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                                //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = cbPattern-(2-1)-2-4; else {
                                // Above line is replaced in order to strengthen the skip by checking the middle DWORD,if the two DWORDs are 'ab' and 'cd' i.e. [2x][2a][2b][2c][2d] then the middle DWORD is 'bc'.
                                // The respective offsets (backwards) are: -10/-8/-6/-4 for 'xa'/'ab'/'bc'/'cd'.
                                //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                // Above line is replaced by next one with better hashing:
                                // When using (16-1) right shifting instead of 16 we will have two different pairs (if they are equal), the highest bit being lost do the job especialy for ASCII texts with no symbols in range 128-255.
                                // Example for genomesque pair TT+TT being shifted by (16-1):
                                // T            = 01010100
                                // TT           = 01010100 01010100
                                // TTTT         = 01010100 01010100 01010100 01010100
                                // TTTT>>16     = 00000000 00000000 01010100 01010100
                                // TTTT>>(16-1) = 00000000 00000000 10101000 10101000 <--- Due to the left shift by 1, the 8th bits of 1st and 2nd bytes are populated - usually they are 0 for English texts & 'ACGT' data.
                                //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                // 'Maximus' uses branched 'if', again.
                                if ( \
                                ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6 +1]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6 +1]&0xFFFF) ) & ( (1<<16)-1 )] ) == 0 \
                                || ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4 +1]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4 +1]&0xFFFF) ) & ( (1<<16)-1 )] ) == 0 \
                                ) Gulliver = cbPattern-(2-1)-2-4-2 +1; else {
                                // Above line is not optimized (several a SHR are used), we have 5 non-overlapping WORDs, or 3 overlapping WORDs, within 4 overlapping DWORDs so:
// [2x][2a][2b][2c][2d]
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16) =     !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                       |
//     DWORD #3                                                                      |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16) =     !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//         DWORD #2                                                                      |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16) =     !SHR to be avoided!     |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//             DWORD #1                                                                  |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                              |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
//
// So in order to remove 3 SHR instructions the equal extractions are:
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) =  !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                       |
//     DWORD #3                                                                      |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) =  !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//         DWORD #2                                                                      |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) =  !SHR to be avoided!     |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//             DWORD #1                                                                  |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                              |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
                                //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-6; else {
// Since the above Decumanus mumbo-jumbo (3 overlapping lookups vs 2 non-overlapping lookups) is not fast enough we go DuoDecumanus or 3x4:
```

```
// [2y][2x][2a][2b][2c][2d]
// DWORD #3
//         DWORD #2
//                 DWORD #1
                                        //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]&0xFFFF) ) & ( (1<<16)-1 )] ) < 2 ) Gulliver = cbPattern-(2-1)-2-8; else {
                                                if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) {
                                                        // Order 4 [
                                                // Let's try something "outrageous" like comparing with[out] overlap BBs 4bytes long instead of 1 byte back-to-back:
                                                // Inhere we are using order 4, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest fox' and Order=4 we have BBs = 11-4+1=8:
                                                //0:"fast" if the comparison failed here, 'count' is 1; 'Gulliver' is cbPattern-(4-1)-7
                                                //1:"aste" if the comparison failed here, 'count' is 2; 'Gulliver' is cbPattern-(4-1)-6
                                                //2:"stes" if the comparison failed here, 'count' is 3; 'Gulliver' is cbPattern-(4-1)-5
                                                //3:"test" if the comparison failed here, 'count' is 4; 'Gulliver' is cbPattern-(4-1)-4
                                                //4:"est " if the comparison failed here, 'count' is 5; 'Gulliver' is cbPattern-(4-1)-3
                                                //5:"st f" if the comparison failed here, 'count' is 6; 'Gulliver' is cbPattern-(4-1)-2
                                                //6:"t fo" if the comparison failed here, 'count' is 7; 'Gulliver' is cbPattern-(4-1)-1
                                                //7:" fox" if the comparison failed here, 'count' is 8; 'Gulliver' is cbPattern-(4-1)
                                                        count = cbPattern-4+1;
                                                        // Below comparison is UNIdirectional:
                                                        while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                                count = count-4;
// count = cbPattern-4+1 = 23-4+1 = 20
// boomshakalakaZZZZZZ[ZZZZ] 20
// boomshakalakaZZ[ZZZZ]ZZZZ 20-4
// boomshakala[kaZZ]ZZZZZZZZ 20-8 = 12
// boomsha[kala]kaZZZZZZZZZZ 20-12 = 8
// boo[msha]kalakaZZZZZZZZZZ 20-16 = 4

// If we miss to hit then no need to compare the original: Needle
if ( count <= 0 ) {
// I have to add out-of-range checks...
// i-(PRIMALposition-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4

// FIX from 2014-Apr-27:
// Because (count-1) is negative, above fours are reduced to next twos:
// i-(PRIMALposition-1)+(count-1) >= 0
// &pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4
// The line below is BUGGY:
//if ( (i-(PRIMALposition-1) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) && (&pbTarget[i-(PRIMALposition-1)+(count-1)] <= pbTargetMax - 4) ) {
// The line below is OKAY:
if ( ((signed int)(i-(PRIMALposition-1)+(count-1)) >= 0) && (&pbTarget[i-(PRIMALposition-1)] <= pbTargetMax - 4) ) {

      if ( *(uint32_t *)&pbTarget[i-(PRIMALposition-1)] == *(uint32_t *)(pbPattern-(PRIMALposition-1)) ) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                count = PRIMALlengthCANDIDATE-4+1;
                while ( count > 0 && *(uint32_t *)(pbPattern-(PRIMALposition-1)+count-1) == *(uint32_t *)(&pbTarget[i-(PRIMALposition-1)]+(count-1)) )
                        count = count-4;
                if ( count <= 0 ) return(pbTarget+i-(PRIMALposition-1));
      }
}
}

                                                // In order to avoid only-left or only-right WCS the memcmp should be done as left-to-right and right-to-left AT THE SAME TIME.
                                                // Below comparison is BIdirectional. It pays off when needle is 8+++ long:
                                                        for (count = cbPattern-4+1; count > 0; count = count-4) {
//                                                              if ( *(uint32_t *)(pbPattern+count-1) != *(uint32_t *)(&pbTarget[i]+(count-1)) ) {break;};
//                                                              if ( *(uint32_t *)(pbPattern+(cbPattern-4+1)-count) != *(uint32_t *)(&pbTarget[i]+(cbPattern-4+1)-count) ) {count = (cbPattern-4+1)-count +(1); break;}
```

```
   // +(1) because two lookups are implemented as one, also no danger of 'count' being 0 because of the fast check outwith the 'while': if ( *(uint32_t *)&pbTarget[i] == ulHashPattern)
418 //                                                                      }
419 //                                                                      if ( count <= 0 ) return(pbTarget+i);
420                                                                         // Checking the order 2 pairs in mismatched DWORD, all the 3:
421                                                                         //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] == 0 ) Gulliver = count; // 1 or bigger, as it should
422                                                                         //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] == 0 ) Gulliver = count+1; // 1 or bigger, as it should
423                                                                         //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1+1]] == 0 ) Gulliver = count+1+1; // 1 or bigger, as it should
424                                                                 //      if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] + bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] + bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1+1]] < 3 ) Gulliver = count; // 1 or bigger, as it should, THE MIN(count,count+1,count+1+1)
425                                                                         // Above compound 'if' guarantees not that Gulliver > 1, an example:
426                                                                         // Needle:    fastest tax
427                                                                         // Window: ...fastast tax...
428                                                                         // After matching ' tax' vs ' tax' and 'fast' vs 'fast' the mismathced DWORD is 'test' vs 'tast':
429                                                                         // 'tast' when factorized down to order 2 yields: 'ta','as','st' - all the three when summed give 1+1+1=3 i.e. Gulliver remains 1.
430                                                                         // Roughly speaking, this attempt maybe has its place in worst-case scenarios but not in English text and even not in ACGT data, that's why I commented it in original 'Shockeroo'.
431                                                                         //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>16)+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1 or bigger, as it should
432                                                                         // Above line is replaced by next one with better hashing:
433 //                                                                              if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>(16-1))+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1 or bigger, as it should
434                                                                 // Order 4 ]
435                                                         }
436                                                 }
437                                 } else Gulliver = cbPattern-(2-1)-2; // -2 because we check the 4 rightmost bytes not 2.
438                                 i = i + Gulliver;
439                                 //GlobalI++; // Comment it, it is only for stats.
440                         }
441                         return(NULL);
442
443               } // if ( cbPattern<=NeedleThreshold2vs4swampLITE )
444               } // if ( cbPattern<=NeedleThreshold2vs4swampLITE )
445 } //if ( cbPattern<4 )
446 }
447
448 // Fixed version from 2012-Feb-27.
449 // Caution: For better speed the case 'if (cbPattern==1)' was removed, so Pattern must be longer than 1 char.
450 char * Railgun_Doublet (char * pbTarget, char * pbPattern, uint32_t cbTarget, uint32_t cbPattern)
451 {
452 char * pbTargetMax = pbTarget + cbTarget;
453 register uint32_t ulHashPattern;
454 uint32_t ulHashTarget, count, countSTATIC;
455
456 if (cbPattern > cbTarget) return(NULL);
457
458 countSTATIC = cbPattern-2;
459
460 pbTarget = pbTarget+cbPattern;
461 ulHashPattern = (*(uint16_t *)(pbPattern));
462
463 for ( ;; ) {
464       if ( ulHashPattern == (*(uint16_t *)(pbTarget-cbPattern)) ) {
465                 count = countSTATIC;
466                 while ( count && *(char *)(pbPattern+2+(countSTATIC-count)) == *(char *)(pbTarget-cbPattern+2+(countSTATIC-count)) ) {
467                           count--;
468                 }
469                 if ( count == 0 ) return((pbTarget-cbPattern));
470       }
471       pbTarget++;
472       if (pbTarget > pbTargetMax) return(NULL);
473 }
```

```
474 }
475
476 // Last change: 2014-May-07
477 // If you want to help me to improve it, email me at: sanmayce@sanmayce.com
478 // Enfun!


// ### Mix(2in1) of Karp-Rabin & Boyer-Moore-Horspool algorithm [
// Caution: For better speed the case 'if (cbPattern==1)' was removed, so Pattern must be longer than 1 char.
char * Railgun_Quadruplet (char * pbTarget,
     char * pbPattern,
     unsigned long cbTarget,
     unsigned long cbPattern)
{
    char * pbTargetMax = pbTarget + cbTarget;
    register unsigned long  ulHashPattern;
    unsigned long ulHashTarget;
    unsigned long count;
    unsigned long countSTATIC;
//  unsigned long countRemainder;

/*
    const unsigned char SINGLET = *(char *)(pbPattern);
    const unsigned long Quadruplet2nd = SINGLET<<8;
    const unsigned long Quadruplet3rd = SINGLET<<16;
    const unsigned long Quadruplet4th = SINGLET<<24;
*/
    unsigned char SINGLET;
    unsigned long Quadruplet2nd;
    unsigned long Quadruplet3rd;
    unsigned long Quadruplet4th;

    unsigned long  AdvanceHopperGrass;

    long i; //BMH needed
    int a, j, bm_bc[ASIZE]; //BMH needed
    unsigned char ch; //BMH needed
//    unsigned char lastch, firstch; //BMH needed

    if (cbPattern > cbTarget)
        return(NULL);

// Doesn't work when cbPattern = 1
// The next IF-fragment works very well with cbPattern>1, OBVIOUSLY IT MUST BE UNROLLED(but crippled with less functionality) SINCE either cbPattern=2 or cbPattern=3!
if ( cbPattern<4) { // This IF makes me unhappy: it slows down from 390KB/clock to 367KB/clock for 'fast' pattern. This fragment(for 2..3 pattern lengths) is needed because I need a function different than strchr but sticking to strstr
    i.e. lengths above 1 are to be handled.
        pbTarget = pbTarget+cbPattern;
        ulHashPattern = ( (*(char *)(pbPattern))<<8 ) + *(pbPattern+(cbPattern-1));
//        countSTATIC = cbPattern-2;

if ( cbPattern==3) {
    for ( ;; )
    {
        if ( ulHashPattern == ( (*(char *)(pbTarget-3))<<8 ) + *(pbTarget-1) ) {
         if ( *(char *)(pbPattern+1) == *(char *)(pbTarget-2) ) return((pbTarget-3));
        }
        if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) pbTarget++;
        pbTarget++;
        if (pbTarget > pbTargetMax)
            return(NULL);
    }
```

```
} else {
}
    for ( ;; )
    {
        // The line below gives for 'cbPattern'>=1:
        // Karp_Rabin_Kaze_4_OCTETS_hits/Karp_Rabin_Kaze_4_OCTETS_clocks: 4/543
        // Karp_Rabin_Kaze_4_OCTETS performance: 372KB/clock
/*
        if ( (ulHashPattern == ( (*(char *)(pbTarget-cbPattern))<<8 ) + *(pbTarget-1)) && !memcmp(pbPattern, pbTarget-cbPattern, (unsigned int)cbPattern) )
            return((long)(pbTarget-cbPattern));
*/

        // The fragment below gives for 'cbPattern'>=2:
        // Karp_Rabin_Kaze_4_OCTETS_hits/Karp_Rabin_Kaze_4_OCTETS_clocks: 4/546
        // Karp_Rabin_Kaze_4_OCTETS performance: 370KB/clock

/*
//For 2 and 3 [
        if ( ulHashPattern == ( (*(char *)(pbTarget-cbPattern))<<8 ) + *(pbTarget-1) ) {
//          count = countSTATIC;
         count = cbPattern-2;
//          while ( count && *(char *)(pbPattern+1+(countSTATIC-count)) == *(char *)(pbTarget-cbPattern+1+(countSTATIC-count)) ) {
         while ( count && *(char *)(pbPattern+1) == *(char *)(pbTarget-2) ) { // Crippling i.e. only 2 and 3 chars are allowed!
              count--;
         }
         if ( count == 0) return((pbTarget-cbPattern));
        }
        if ( (char)(ulHashPattern>>8) != *(pbTarget-cbPattern+1) ) pbTarget++;
//For 2 and 3 ]
*/


        if ( ulHashPattern == ( (*(char *)(pbTarget-2))<<8 ) + *(pbTarget-1) )
            return((pbTarget-2));
        if ( (char)(ulHashPattern>>8) != *(pbTarget-1) ) pbTarget++;


        // The fragment below gives for 'cbPattern'>=2:
    // Karp_Rabin_Kaze_4_OCTETS_hits/Karp_Rabin_Kaze_4_OCTETS_clocks: 4/554
    // Karp_Rabin_Kaze_4_OCTETS performance: 364KB/clock
/*
        if ( ulHashPattern == ( (*(char *)(pbTarget-cbPattern))<<8 ) + *(pbTarget-1) ) {
         count = countSTATIC>>2;
         countRemainder = countSTATIC % 4;

         while ( count && *(unsigned long *)(pbPattern+1+((count-1)<<2)) == *(unsigned long *)(pbTarget-cbPattern+1+((count-1)<<2)) ) {
              count--;
         }
     //if (count == 0) {  // Disastrous degradation only from this line(317KB/clock when 1+2x4+2+1 bytes pattern: 'skillessness'; 312KB/clock when 1+1x4+2+1 bytes pattern: 'underdog'), otherwise 368KB/clock.
         while ( countRemainder && *(char *)(pbPattern+1+(countSTATIC-countRemainder)) == *(char *)(pbTarget-cbPattern+1+(countSTATIC-countRemainder)) ) {
              countRemainder--;
         }
         //if ( countRemainder == 0) return((long)(pbTarget-cbPattern));
         if ( count+countRemainder == 0) return((long)(pbTarget-cbPattern));
         //}
        }
*/

        pbTarget++;
        if (pbTarget > pbTargetMax)
            return(NULL);
```

```
    }
} else { //if ( cbPattern<4)
if (cbTarget<961) // This value is arbitrary(don't know how exactly), it ensures(at least must) better performance than 'Boyer_Moore_Horspool'.
{
        pbTarget = pbTarget+cbPattern;
        ulHashPattern = *(unsigned long *)(pbPattern);
//        countSTATIC = cbPattern-1;

    //SINGLET = *(char *)(pbPattern);
    SINGLET = ulHashPattern & 0xFF;
    Quadruplet2nd = SINGLET<<8;
    Quadruplet3rd = SINGLET<<16;
    Quadruplet4th = SINGLET<<24;

    for ( ;; )
    {
    AdvanceHopperGrass = 0;
    ulHashTarget = *(unsigned long *)(pbTarget-cbPattern);

        if ( ulHashPattern == ulHashTarget ) { // Three unnecessary comparisons here, but 'AdvanceHopperGrass' must be calculated - it has a higher priority.
//          count = countSTATIC;
//          while ( count && *(char *)(pbPattern+1+(countSTATIC-count)) == *(char *)(pbTarget-cbPattern+1+(countSTATIC-count)) ) {
//          if ( countSTATIC==AdvanceHopperGrass+count && SINGLET != *(char *)(pbTarget-cbPattern+1+(countSTATIC-count)) ) AdvanceHopperGrass++;
//                count--;
//          }
         count = cbPattern-1;
         while ( count && *(char *)(pbPattern+(cbPattern-count)) == *(char *)(pbTarget-count) ) {
           if ( cbPattern-1==AdvanceHopperGrass+count && SINGLET != *(char *)(pbTarget-count) ) AdvanceHopperGrass++;
               count--;
         }
         if ( count == 0) return((pbTarget-cbPattern));
        } else { // The goal here: to avoid memory accesses by stressing the registers.
    if ( Quadruplet2nd != (ulHashTarget & 0x0000FF00) ) {
         AdvanceHopperGrass++;
         if ( Quadruplet3rd != (ulHashTarget & 0x00FF0000) ) {
              AdvanceHopperGrass++;
              if ( Quadruplet4th != (ulHashTarget & 0xFF000000) ) AdvanceHopperGrass++;
         }
    }
    }

    AdvanceHopperGrass++;

    pbTarget = pbTarget + AdvanceHopperGrass;
        if (pbTarget > pbTargetMax)
            return(NULL);
    }
} else { //if (cbTarget<961)
        countSTATIC = cbPattern-2;
    /* Preprocessing */
    for (a=0; a < ASIZE; a++) bm_bc[a]=cbPattern;
    for (j=0; j < cbPattern-1; j++) bm_bc[pbPattern[j]]=cbPattern-j-1;

    /* Searching */
    //lastch=pbPattern[cbPattern-1];
    //firstch=pbPattern[0];
    i=0;
    while (i <= cbTarget-cbPattern) {
       ch=pbTarget[i+cbPattern-1];
       //if (ch ==lastch)
          //if (memcmp(&pbTarget[i],pbPattern,cbPattern-1) == 0) OUTPUT(i);
```

```
          //if (ch == lastch && pbTarget[i] == firstch && memcmp(&pbTarget[i],pbPattern,cbPattern-1) == 0) return(i);  // Kaze: The idea(to prevent execution of slower 'memcmp') is borrowed from Karp-Rabin i.e. to perform a slower check
    only when the target "looks like".
          if (ch == pbPattern[cbPattern-1] && pbTarget[i] == pbPattern[0])
             {
         count = countSTATIC;
         while ( count && *(char *)(pbPattern+1+(countSTATIC-count)) == *(char *)(&pbTarget[i]+1+(countSTATIC-count)) ) {
              count--;
         }
         if ( count == 0) return(pbTarget+i);
         }
       i+=bm_bc[ch];
    }
    return(NULL);
} //if (cbTarget<961)
} //if ( cbPattern<4)
}
// ### Mix(2in1) of Karp-Rabin & Boyer-Moore-Horspool algorithm ]


/*
// DAWG "Directed Acyclic Word Graph" Definition (1): A directed acyclic graph representing the suffixes of a given string in which each edge is labeled with a character. The characters along a path from the root to a node are the
     substring which the node represents.
// DAWG "Directed Acyclic Word Graph" Definition (2): A finite state machine that recognizes a set of words.
// BNDM ("Backward Nondeterministic DAWG Matching")
// Mischa Sandberg
char *bndm(char *target, int tgtlen, char *pattern, int patlen)
{
    uint8_t    *tgt = (uint8_t*)target, *pat = (uint8_t*)pattern;
    int        i, j;
    uint32_t   mask, maskv[256] = {0};

    for (i = 0; i < patlen; ++i)
        maskv[pat[i]] |= 1 << (patlen - 1 - i);

    for (i = 0; i < tgtlen - patlen; i += j) {
        mask = maskv[tgt[patlen - 1 + i]];
        for (j = patlen; mask;) {
            if (!--j) return target + i;
            mask = (mask << 1) & maskv[tgt[i + j - 1]];
        }
    }
    return NULL;
}
*/
char * BNDM_32(char * pbTarget, char * pbPattern, unsigned long cbTarget, unsigned long cbPattern)
{
    uint8_t    *tgt = (uint8_t*)pbTarget, *pat = (uint8_t*)pbPattern;
    int        i, j;
    uint32_t   mask, maskv[256] = {0};

    if (cbPattern > cbTarget)
        return(NULL);

    for (i = 0; i < cbPattern; ++i)
        maskv[pat[i]] |= 1 << (cbPattern - 1 - i);

    i=0;
    while (i <= cbTarget-cbPattern) {
         mask = maskv[tgt[cbPattern - 1 + i]];
         j = cbPattern;
         while (mask) {
```

```
                if (!--j) return pbTarget + i;
                mask = (mask << 1) & maskv[tgt[i + j - 1]];
            }
            i= i + j;
        }
//    for (i = 0; i <= cbTarget - cbPattern; i += j) {
//        mask = maskv[tgt[cbPattern - 1 + i]];
//        for (j = cbPattern; mask;) {
//            if (!--j) return pbTarget + i;
//            mask = (mask << 1) & maskv[tgt[i + j - 1]];
//        }
//    }
    return NULL;
}

// Caution: 1 <= cbPattern <= 32
char * BNDM_32_count_hits(char * pbTarget, char * pbPattern, unsigned long cbTarget, unsigned long cbPattern)
{
    uint8_t    *tgt = (uint8_t*)pbTarget, *pat = (uint8_t*)pbPattern;
    int        i, j;
    uint32_t   mask, maskv[256] = {0};

    unsigned long AdvanceHopperGrass = 0;

    if (cbPattern > cbTarget)
        return(NULL);

    for (i = 0; i < cbPattern; ++i) {
        maskv[pat[i]] |= 1 << (cbPattern - 1 - i);
//printf("i=%lu, %c, %lu\n", i, pat[i], maskv[pat[i]]);

//printf("%s ", _ui64toaKAZEzerocomma(i, llTOaDigits2, 10)+(60-2) );
//printf("%c ",  pat[i]);
//printf("%s\n", _ui64toaKAZEzerocomma(maskv[pat[i]], llTOaDigits2, 2)+(60-39) );
/*
Pattern: alfalfa
Doing Search for Pattern(7bytes) into String(206908949bytes) as-one-line ...

                                  alf,alfa
00 a 0000,0000,0000,0000,0000,0000,0100,0000
01 l 0000,0000,0000,0000,0000,0000,0010,0000
02 f 0000,0000,0000,0000,0000,0000,0001,0000
03 a 0000,0000,0000,0000,0000,0000,0100,1000
04 l 0000,0000,0000,0000,0000,0000,0010,0100
05 f 0000,0000,0000,0000,0000,0000,0001,0010
06 a 0000,0000,0000,0000,0000,0000,0100,1001

BNDM_32:                        1870KB/clock / 0691%, 29,938392 iterations
Railgun_Quadruplet_7sun:        1642KB/clock / 0748%, 27,654192 iterations
Railgun_Quadruplet_7:           1642KB/clock / 0662%, 31,226521 iterations
Railgun_Quadruplet_7sunhorse:   1202KB/clock / 0786%, 26,295205 iterations
Railgun_Quadruplet_7Elsiane:    1063KB/clock / 0871%, 23,744531 iterations
Railgun_Quadruplet_7Gulliver:   1496KB/clock / 0598%, 34,583650 iterations
Railgun_Quadruplet_7Hasherezade: 1507KB/clock / 0598%, 34,583650 iterations

Pattern: underrated
Doing Search for Pattern(10bytes) into String(206908949bytes) as-one-line ...

                               un,derr,ated
00 u 0000,0000,0000,0000,0000,0010,0000,0000
01 n 0000,0000,0000,0000,0000,0001,0000,0000
```

```
02 d 0000,0000,0000,0000,0000,0000,1000,0000
03 e 0000,0000,0000,0000,0000,0000,0100,0000
04 r 0000,0000,0000,0000,0000,0000,0010,0000
05 r 0000,0000,0000,0000,0000,0000,0011,0000
06 a 0000,0000,0000,0000,0000,0000,0000,1000
07 t 0000,0000,0000,0000,0000,0000,0000,0100
08 e 0000,0000,0000,0000,0000,0000,0100,0010
09 d 0000,0000,0000,0000,0000,0000,1000,0001

BNDM_32:                        1287KB/clock / 0965%, 21,438744 iterations
Railgun_Quadruplet_7sun:        1942KB/clock / 0894%, 23,127481 iterations
Railgun_Quadruplet_7:           1942KB/clock / 0804%, 25,703793 iterations
Railgun_Quadruplet_7sunhorse:   1141KB/clock / 1025%, 20,167543 iterations
Railgun_Quadruplet_7Elsiane:    0768KB/clock / 1072%, 19,285870 iterations
Railgun_Quadruplet_7Gulliver:   1906KB/clock / 0885%, 23,374120 iterations
Railgun_Quadruplet_7Hasherezade: 1870KB/clock / 0885%, 23,374120 iterations

Pattern: bye-bye-bye-bye-bye-
Doing Search for Pattern(20bytes) into String(206908949bytes) as-one-line ...

                   bye-,bye-,bye-,bye-,bye-
00 b 0000,0000,0000,1000,0000,0000,0000,0000
01 y 0000,0000,0000,0100,0000,0000,0000,0000
02 e 0000,0000,0000,0010,0000,0000,0000,0000
03 - 0000,0000,0000,0001,0000,0000,0000,0000
04 b 0000,0000,0000,1000,1000,0000,0000,0000
05 y 0000,0000,0000,0100,0100,0000,0000,0000
06 e 0000,0000,0000,0010,0010,0000,0000,0000
07 - 0000,0000,0000,0001,0001,0000,0000,0000
08 b 0000,0000,0000,1000,1000,1000,0000,0000
09 y 0000,0000,0000,0100,0100,0100,0000,0000
10 e 0000,0000,0000,0010,0010,0010,0000,0000
11 - 0000,0000,0000,0001,0001,0001,0000,0000
12 b 0000,0000,0000,1000,1000,1000,1000,0000
13 y 0000,0000,0000,0100,0100,0100,0100,0000
14 e 0000,0000,0000,0010,0010,0010,0010,0000
15 - 0000,0000,0000,0001,0001,0001,0001,0000
16 b 0000,0000,0000,1000,1000,1000,1000,1000
17 y 0000,0000,0000,0100,0100,0100,0100,0100
18 e 0000,0000,0000,0010,0010,0010,0010,0010
19 - 0000,0000,0000,0001,0001,0001,0001,0001

BNDM_32:                        3483KB/clock / 1988%, 10,402957 iterations
Railgun_Quadruplet_7sun:        3207KB/clock / 1891%, 10,940613 iterations
Railgun_Quadruplet_7:           3259KB/clock / 1806%, 11,450773 iterations
Railgun_Quadruplet_7sunhorse:   2557KB/clock / 2069%, 09,995656 iterations
Railgun_Quadruplet_7Elsiane:    2104KB/clock / 2149%, 09,627265 iterations
Railgun_Quadruplet_7Gulliver:   3259KB/clock / 1898%, 10,898305 iterations
Railgun_Quadruplet_7Hasherezade: 3157KB/clock / 1899%, 10,892762 iterations

Pattern: fastest fox with biggest strides
Doing Search for Pattern(32bytes) into String(206908949bytes) as-one-line ...

     fast,est ,fox ,with, big,gest, str,ides
00 f 1000,0000,0000,0000,0000,0000,0000,0000
01 a 0100,0000,0000,0000,0000,0000,0000,0000
02 s 0010,0000,0000,0000,0000,0000,0000,0000
03 t 0001,0000,0000,0000,0000,0000,0000,0000
04 e 0000,1000,0000,0000,0000,0000,0000,0000
05 s 0010,0100,0000,0000,0000,0000,0000,0000
06 t 0001,0010,0000,0000,0000,0000,0000,0000
```

```
07   0000,0001,0000,0000,0000,0000,0000,0000
08 f 1000,0000,1000,0000,0000,0000,0000,0000
09 o 0000,0000,0100,0000,0000,0000,0000,0000
10 x 0000,0000,0010,0000,0000,0000,0000,0000
11   0000,0001,0001,0000,0000,0000,0000,0000
12 w 0000,0000,0000,1000,0000,0000,0000,0000
13 i 0000,0000,0000,0100,0000,0000,0000,0000
14 t 0001,0010,0000,0010,0000,0000,0000,0000
15 h 0000,0000,0000,0001,0000,0000,0000,0000
16   0000,0001,0001,0000,1000,0000,0000,0000
17 b 0000,0000,0000,0000,0100,0000,0000,0000
18 i 0000,0000,0000,0100,0010,0000,0000,0000
19 g 0000,0000,0000,0000,0001,0000,0000,0000
20 g 0000,0000,0000,0000,0001,1000,0000,0000
21 e 0000,1000,0000,0000,0000,0100,0000,0000
22 s 0010,0100,0000,0000,0000,0010,0000,0000
23 t 0001,0010,0000,0010,0000,0001,0000,0000
24   0000,0001,0001,0000,1000,0000,1000,0000
25 s 0010,0100,0000,0000,0000,0010,0100,0000
26 t 0001,0010,0000,0010,0000,0001,0010,0000
27 r 0000,0000,0000,0000,0000,0000,0001,0000
28 i 0000,0000,0000,0100,0010,0000,0000,1000
29 d 0000,0000,0000,0000,0000,0000,0000,0100
30 e 0000,1000,0000,0000,0000,0100,0000,0010
31 s 0010,0100,0000,0000,0000,0010,0100,0001

BNDM_32:                        2886KB/clock / 3113%, 06,644708 iterations
Railgun_Quadruplet_7sun:        2694KB/clock / 1584%, 13,060463 iterations
Railgun_Quadruplet_7:           2767KB/clock / 1511%, 13,689243 iterations
Railgun_Quadruplet_7sunhorse:   1820KB/clock / 2138%, 09,677267 iterations
Railgun_Quadruplet_7Elsiane:    1566KB/clock / 2143%, 09,652548 iterations
Railgun_Quadruplet_7Gulliver:   3108KB/clock / 2924%, 07,074287 iterations
Railgun_Quadruplet_7Hasherezade: 2971KB/clock / 3041%, 06,801754 iterations
*/
}

    i=0;
    while (i <= cbTarget-cbPattern) {
          mask = maskv[tgt[cbPattern - 1 + i]];
          j = cbPattern;
          while (mask) {
                if (j==1) {Railgunhits++; mask = 0;} //return pbTarget + i;
                else {j--; mask = (mask << 1) & maskv[tgt[i + j - 1]];}
          }
          i= i + j;

//    for (i = 0; i <= cbTarget - cbPattern; i += j) {
//        mask = maskv[tgt[cbPattern - 1 + i]];
//        for (j = cbPattern; mask;) {
//            if (!--j) {Railgunhits++; j++; mask = 0;} //return pbTarget + i;
//            else mask = (mask << 1) & maskv[tgt[i + j - 1]];
//        }
    AdvanceHopperGrass++;
    }

    GlobalSP += (int)((double)cbTarget/AdvanceHopperGrass*100);
    GlobalI += AdvanceHopperGrass;
    printf("Skip-Performance(bigger-the-better): %d%%, %d skips/iterations\n",(int)((double)cbTarget/AdvanceHopperGrass*100), AdvanceHopperGrass);

    return NULL;
}
```

```
#define ASIZE 256
// ### Boyer-Moore-Horspool algorithm [
long HORSPOOL(y, x, n, m)
    char *y;
    char *x;
    long n;
    int m;
   {
    long i;
    int a, j, bm_bc[ASIZE];
    unsigned char ch;
    unsigned char lastch;

    /* Preprocessing */
    for (a=0; a < ASIZE; a++) bm_bc[a]=m;
    for (j=0; j < m-1; j++) {
    bm_bc[*(unsigned char *)&x[j]]=m-j-1;
    }

    /* Searching */
    lastch=*(unsigned char *)&x[m-1];
    i=0;
    while (i <= n-m) {
       ch=*(unsigned char *)&y[i+m-1];
       if (ch ==lastch)
          //if (memcmp(&y[i],x,m-1) == 0) OUTPUT(i);
          if (memcmp(&y[i],x,m-1) == 0) return(i);
       i+=bm_bc[ch];
    }
    return(-1);
   }


// Railgun_Ennearch, copyleft 2014-Jan-15, Kaze.
// heptarch, noun.
// L17.
// [from HEPTA- + -ARCH, after TETRARCH noun.]
// † 1. A seventh king (see Revelation 17:9-11). Only in L17.
// 2. A ruler of one of seven divisions of a country; esp. any of the rulers of the Anglo-Saxon Heptarchy. E19.
// /SOED/
// decarch, noun.
// Also dekarch. M17.
// [Greek dekarkhUs, -os decurion, from deka ten + arkhos leader.]
// Greek History. Each of a ruling body of ten.
// /SOED/
// diarchy, noun.
// Also dyarchy. M19.
// [from DI-2 after monarchy.]
// A mode of joint government by two; government by two independent authorities; spec. the system of provincial government in India from 1921 to 1937.
// /SOED/
// monadic -> mona+ARCH
// duadic/dyadic -> dyarchy
// triadic ->
// quadic/tetradic ->
// pentadic -> pentarch
// hexadic ->
// heptadic/hebdomadic (for 7) -> hepta+ARCH
// octadic ->  OCTA+ARCHY
// enneadic (for 9) ->
```

```
// decadic (for 10) ->
// Caution: For better speed the case 'if (cbPattern==1)' was removed, so Pattern must be longer than 1 char.
#define NeedleThreshold2vs4Nonus 9+10 // Should be bigger than 9. BMH2 works up to this value (inclusive), if bigger then BMH4 takes over.
char * Railgun_Ennearch (char * pbTarget, char * pbPattern, uint32_t cbTarget, uint32_t cbPattern)
{
    char * pbTargetMax = pbTarget + cbTarget;
    register uint32_t ulHashPattern;
    signed long count;

    unsigned char bm_Horspool_Order2[256*256]; // Bitwise soon...
    uint32_t i, Gulliver;

    if (cbPattern > cbTarget) return(NULL);

    if ( cbPattern<4 ) {
          // SSE2 i.e. 128bit Assembly rules here:
          // ...
          pbTarget = pbTarget+cbPattern;
          ulHashPattern = ( (*(char *)(pbPattern))<<8 ) + *(pbPattern+(cbPattern-1));
          if ( cbPattern==3 ) {
                    for ( ;; ) {
                              if ( ulHashPattern == ( (*(char *)(pbTarget-3))<<8 ) + *(pbTarget-1) ) {
                                        if ( *(char *)(pbPattern+1) == *(char *)(pbTarget-2) ) return((pbTarget-3));
                              }
                              if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) {
                                        pbTarget++;
                                        if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) pbTarget++;
                              }
                              pbTarget++;
                              if (pbTarget > pbTargetMax) return(NULL);
                    }
          } else {
          }
          for ( ;; ) {
                    if ( ulHashPattern == ( (*(char *)(pbTarget-2))<<8 ) + *(pbTarget-1) ) return((pbTarget-2));
                    if ( (char)(ulHashPattern>>8) != *(pbTarget-1) ) pbTarget++;
                    pbTarget++;
                    if (pbTarget > pbTargetMax) return(NULL);
          }
    } else { //if ( cbPattern<4 )
          if ( cbPattern<=NeedleThreshold2vs4Nonus ) {
                    // BMH order 2, needle should be >=4:
                    ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
                    for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
                    for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=1;
                    i=0;
                    while (i <= cbTarget-cbPattern) {
                              Gulliver = 1; // 'Gulliver' is the skip
                              if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]] != 0 ) {
                                        if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1-2]] == 0 ) Gulliver = cbPattern-(2-1)-2; else {
                                                  if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
                                                            count = cbPattern-4+1;
                                                            while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                                      count = count-4;
                                                            if ( count <= 0 ) return(pbTarget+i);
                                                  }
                                        }
                              } else Gulliver = cbPattern-(2-1);
                              i = i + Gulliver;
                              //GlobalI++; // Comment it, it is only for stats.
                    }
```

```
                return(NULL);
        } else { // if ( cbPattern<=NeedleThreshold2vs4Nonus )
                // BMH pseudo-order 4, needle should be >=8+2:
                ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
                for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
                // In line below we "hash" 4bytes to 2bytes i.e. 16bit table, how to compute TOTAL number of BBs, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest
    fox' and Order=4 we have BBs = 11-4+1=8:
                //"fast"
                //"aste"
                //"stes"
                //"test"
                //"est "
                //"st f"
                //"t fo"
                //" fox"
                //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( *(unsigned short *)(pbPattern+i+0) + *(unsigned short *)(pbPattern+i+2) ) & ( (1<<16)-1 )]=1;
                //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>16)+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
                // Above line is replaced by next one with better hashing:
                for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>(16-1))+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
                i=0;
                while (i <= cbTarget-cbPattern) {
                        Gulliver = 1;
                        //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                        // Above line is replaced by next one with better hashing:
                        if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                                //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = cbPattern-(2-1)-2-4; else {
                                // Above line is replaced in order to strengthen the skip by checking the middle DWORD,if the two DWORDs are 'ab' and 'cd' i.e. [2x][2a][2b][2c][2d] then the middle DWORD is 'bc'.
                                // The respective offsets (backwards) are: -10/-8/-6/-4 for 'xa'/'ab'/'bc'/'cd'.
                                //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t
    *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-
    2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                // Above line is replaced by next one with better hashing:
                                // When using (16-1) right shifting instead of 16 we will have two different pairs (if they are equal), the highest bit being lost do the job especialy for ASCII texts with no symbols in range
    128-255.
                                // Example for genomesque pair TT+TT being shifted by (16-1):
                                // T           = 01010100
                                // TT          = 01010100 01010100
                                // TTTT        = 01010100 01010100 01010100 01010100
                                // TTTT>>16    = 00000000 00000000 01010100 01010100
                                // TTTT>>(16-1) = 00000000 00000000 10101000 10101000 <--- Due to the left shift by 1, the 8th bits of 1st and 2nd bytes are populated - usually they are 0 for English texts & 'ACGT' data.
                                //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t
    *)&pbTarget[i+cbPattern-1-1-2-4]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-
    1-2-2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                // 'Maximus' uses branched 'if', again.
                                if ( \
                                ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6 +1]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6 +1]&0xFFFF) ) & ( (1<<16)-1 )] ) == 0 \
                                || ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4 +1]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4 +1]&0xFFFF) ) & ( (1<<16)-1 )] ) == 0 \
                                ) Gulliver = cbPattern-(2-1)-2-4-2 +1; else {
                                // Above line is not optimized (several a SHR are used), we have 5 non-overlapping WORDs, or 3 overlapping WORDs, within 4 overlapping DWORDs so:
// [2x][2a][2b][2c][2d]
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16) =     !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                      |
//     DWORD #3                                                                    |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16) =     !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//          DWORD #2                                                                   |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16) =     !SHR to be avoided!     |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//             DWORD #1                                                                   |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                               |
```

```
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
//
// So in order to remove 3 SHR instructions the equal extractions are:
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) =  !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                       |
//     DWORD #3                                                                    |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) =  !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//         DWORD #2                                                                    |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) =  !SHR to be avoided!      |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//             DWORD #1                                                                  |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                              |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
                                        //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-6; else {
// Since the above Decumanus mumbo-jumbo (3 overlapping lookups vs 2 non-overlapping lookups) is not fast enough we go DuoDecumanus or 3x4:
// [2y][2x][2a][2b][2c][2d]
// DWORD #3
//         DWORD #2
//                 DWORD #1
                                      //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]&0xFFFF) ) & ( (1<<16)-1 )] ) < 2 ) Gulliver = cbPattern-(2-1)-2-8; else {
                                              if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) {
                                                        // Order 4 [
                                              // Let's try something "outrageous" like comparing with[out] overlap BBs 4bytes long instead of 1 byte back-to-back:
                                              // Inhere we are using order 4, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest fox' and Order=4 we have BBs = 11-4+1=8:
                                              //0:"fast" if the comparison failed here, 'count' is 1; 'Gulliver' is cbPattern-(4-1)-7
                                              //1:"aste" if the comparison failed here, 'count' is 2; 'Gulliver' is cbPattern-(4-1)-6
                                              //2:"stes" if the comparison failed here, 'count' is 3; 'Gulliver' is cbPattern-(4-1)-5
                                              //3:"test" if the comparison failed here, 'count' is 4; 'Gulliver' is cbPattern-(4-1)-4
                                              //4:"est " if the comparison failed here, 'count' is 5; 'Gulliver' is cbPattern-(4-1)-3
                                              //5:"st f" if the comparison failed here, 'count' is 6; 'Gulliver' is cbPattern-(4-1)-2
                                              //6:"t fo" if the comparison failed here, 'count' is 7; 'Gulliver' is cbPattern-(4-1)-1
                                              //7:" fox" if the comparison failed here, 'count' is 8; 'Gulliver' is cbPattern-(4-1)
                                                        //count = cbPattern-4+1;
                                                        // Below comparison is UNIdirectional:
                                                        //while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                        //        count = count-4;
// count = cbPattern-4+1 = 23-4+1 = 20
// boomshakalakaZZZZZZ[ZZZZ] 20
// boomshakalakaZZ[ZZZZ]ZZZZ 20-4
// boomshakala[kaZZ]ZZZZZZZZ 20-8 = 12
// boomsha[kala]kaZZZZZZZZZZ 20-12 = 8
// boo[msha]kalakaZZZZZZZZZZ 20-16 = 4
                                              // In order to avoid only-left or only-right WCS the memcmp should be done as left-to-right and right-to-left AT THE SAME TIME.
                                              // Below comparison is BIdirectional. It pays off when needle is 8+++ long:
                                              for (count = cbPattern-4+1; count > 0; count = count-4) {
                                                        if ( *(uint32_t *)(pbPattern+count-1) != *(uint32_t *)(&pbTarget[i]+(count-1)) ) {break;};
                                                        if ( *(uint32_t *)(pbPattern+(cbPattern-4+1)-count) != *(uint32_t *)(&pbTarget[i]+(cbPattern-4+1)-count) ) {count = (cbPattern-4+1)-count +(1); break;} // +(1) because two lookups are implemented as one, also no danger of 'count' being 0 because of the fast check outwith the 'while': if ( *(uint32_t *)&pbTarget[i] == ulHashPattern)
                                              }
                                              if ( count <= 0 ) return(pbTarget+i);
//                                            else {
                                                        // Checking the order 2 pairs in mismatched DWORD, all the 3:
                                                        //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] == 0 ) Gulliver = count; // 1 or bigger, as it should
                                                        //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] == 0 ) Gulliver = count+1; // 1 or bigger, as it should
                                                        //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1+1]] == 0 ) Gulliver = count+1+1; // 1 or bigger, as it should
```

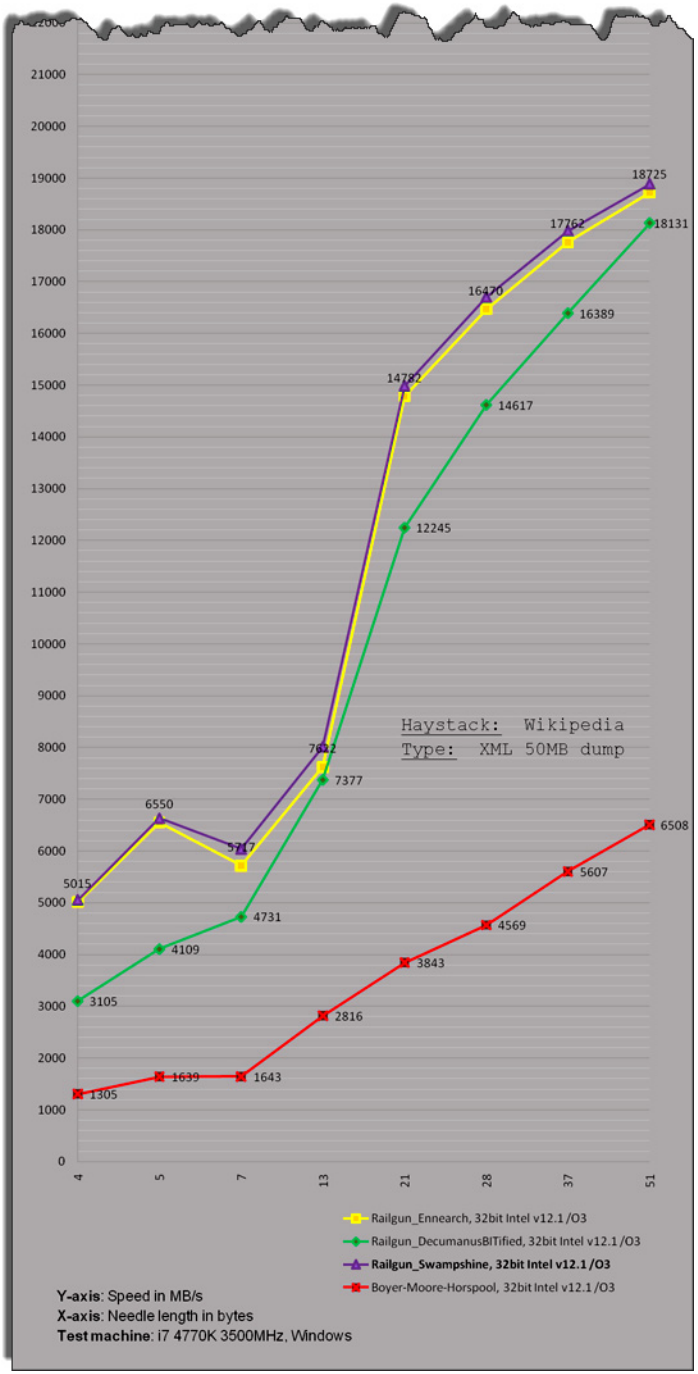
```
                                                                //        if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] + bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] + bm_Horspool_Order2[*(unsigned
    short *)&pbTarget[i+count-1+1+1]] < 3 ) Gulliver = count; // 1 or bigger, as it should, THE MIN(count,count+1,count+1+1)
                                                                // Above compound 'if' guarantees not that Gulliver > 1, an example:
                                                                // Needle:    fastest tax
                                                                // Window: ...fastast tax...
                                                                // After matching ' tax' vs ' tax' and 'fast' vs 'fast' the mismathced DWORD is 'test' vs 'tast':
                                                                // 'tast' when factorized down to order 2 yields: 'ta','as','st' - all the three when summed give 1+1+1=3 i.e. Gulliver remains 1.
                                                                // Roughly speaking, this attempt maybe has its place in worst-case scenarios but not in English text and even not in ACGT data, that's why I commented it in
    original 'Shockeroo'.
                                                                //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>16)+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1 or
    bigger, as it should
                                                                // Above line is replaced by next one with better hashing:
                                                                if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>(16-1))+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1
    or bigger, as it should
//                                                          }
                                                            // Order 4 ]
                                                    }
                                                }
                                    } else Gulliver = cbPattern-(2-1)-2; // -2 because we check the 4 rightmost bytes not 2.
                                    i = i + Gulliver;
                                    //GlobalI++; // Comment it, it is only for stats.
                                }
                                return(NULL);
                        } // if ( cbPattern<=NeedleThreshold2vs4Nonus )
    } //if ( cbPattern<4 )
}


// Railgun_DecumanusBITified, copyleft 2014-Jan-14, Kaze.
// [Latin decumanus var. of decimanus of or belonging to the tenth part or tenth cohort, (by metonymy) large, from decimus: see DECIMAL.]
// 1. Esp. of a wave: very large, immense. M17.
// 2. Roman History. Of or belonging to the tenth cohort. E19.
// decuman gate ~ the main gate of the camp where the tenth cohort was quartered.
// /SOED/
// In Roman city planning, a decumanus was an east-west-oriented road in a Roman city, castra (military camp), or colonia. The main decumanus was the Decumanus Maximus, which normally connected the Porta Praetoria (in a military camp,
    closest to the enemy) to the Porta Decumana (away from the enemy).
// This name comes from the fact that the via decumana or decimana (the tenth) separated the Tenth Cohort from the Ninth in the legionary encampment, in the same way as the via quintana separated the Fifth Cohort from the Sixth.
// In the middle, or groma, the Decumanus Maximus crosses the perpendicular Cardo Maximus, the primary north-south road that was the usual main street. The Forum is normally located close to this intersection of the Decumanus Maximus and
    the Cardo Maximus.
// /Wikipedia/
// duodecimal
// adj.
// 1. Of, relating to, or based on the number 12: the duodecimal number system.
// 2. Of or relating to twelfths.
// n.
// A twelfth.
// [From Latin duodecimus, twelfth, from duodecim, twelve : duo, two; see dwo- in Indo-European roots + decem, ten; see dek' in Indo-European roots.]
// /Heritage/
// Caution: For better speed the case 'if (cbPattern==1)' was removed, so Pattern must be longer than 1 char.
#define NeedleThreshold2vs4Decumanus 10+9 // Should be bigger than 10 (decimanus, got it). BMH2 works up to this value (inclusive), if bigger then BMH4 takes over.
#define _rotl_KAZE(x, n) (((x) << (n)) | ((x) >> (32-(n))))
#define _HASH_Order4_KAZE(x) (( (x>>(16-1))+(x&0xFFFF) ) & ( (1<<16)-1 ))
char * Railgun_DecumanusBITified (char * pbTarget, char * pbPattern, uint32_t cbTarget, uint32_t cbPattern)
{
    char * pbTargetMax = pbTarget + cbTarget;
    register uint32_t ulHashPattern;
    signed long count;

    //unsigned char bm_Horspool_Order2[256*256]; // Bitwise soon...
    unsigned char bm_Horspool_Order2[(256*256)>>3];
```

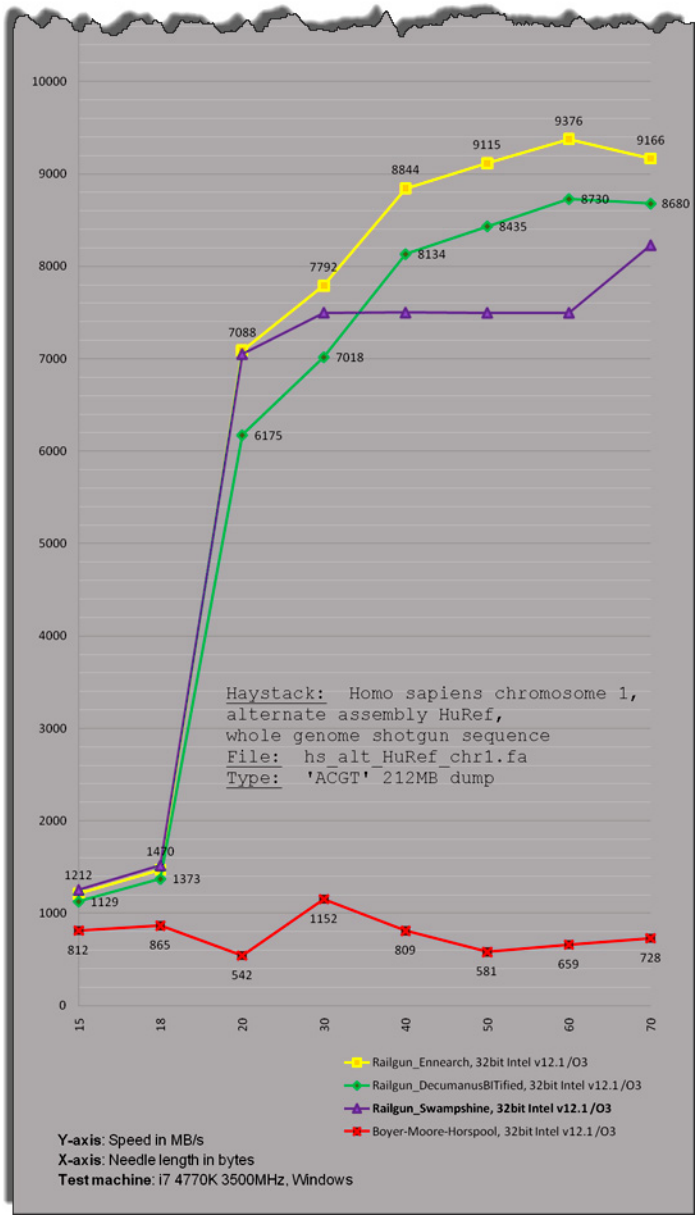
```
	uint32_t i, Gulliver;

	if (cbPattern > cbTarget) return(NULL);

	if ( cbPattern<4 ) {
		// SSE2 i.e. 128bit Assembly rules here:
		// ...
		pbTarget = pbTarget+cbPattern;
		ulHashPattern = ( (*(char *)(pbPattern))<<8 ) + *(pbPattern+(cbPattern-1));
		if ( cbPattern==3 ) {
			for ( ;; ) {
				if ( ulHashPattern == ( (*(char *)(pbTarget-3))<<8 ) + *(pbTarget-1) ) {
					if ( *(char *)(pbPattern+1) == *(char *)(pbTarget-2) ) return((pbTarget-3));
				}
				if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) {
					pbTarget++;
					if ( (char)(ulHashPattern>>8) != *(pbTarget-2) ) pbTarget++;
				}
				pbTarget++;
				if (pbTarget > pbTargetMax) return(NULL);
			}
		} else {
		}
		for ( ;; ) {
			if ( ulHashPattern == ( (*(char *)(pbTarget-2))<<8 ) + *(pbTarget-1) ) return((pbTarget-2));
			if ( (char)(ulHashPattern>>8) != *(pbTarget-1) ) pbTarget++;
			pbTarget++;
			if (pbTarget > pbTargetMax) return(NULL);
		}
	} else { //if ( cbPattern<4 )
		if ( cbPattern<=NeedleThreshold2vs4Decumanus ) {
			// BMH order 2, needle should be >=4:
			ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
			//for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
			for (i=0; i < (256*256)>>3; i++) {bm_Horspool_Order2[i]=0;}
			//for (i=0; i < cbPattern-1; i++) bm_Horspool_Order2[*(unsigned short *)(pbPattern+i)]=1;
			for (i=0; i < cbPattern-2+1; i++) bm_Horspool_Order2[(*(unsigned short *)(pbPattern+i))>>3]= bm_Horspool_Order2[(*(unsigned short *)(pbPattern+i))>>3] | (1<<((*(unsigned short *)(pbPattern+i))&0x7));
			i=0;
			while (i <= cbTarget-cbPattern) {
				Gulliver = 1; // 'Gulliver' is the skip
				//if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1]] != 0 ) {
				if ( ( bm_Horspool_Order2[(*(unsigned short *)&pbTarget[i+cbPattern-1-1])>>3] & (1<<((*(unsigned short *)&pbTarget[i+cbPattern-1-1])&0x7)) ) != 0 ) {
					//if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+cbPattern-1-1-2]] == 0 ) Gulliver = cbPattern-(2-1)-2; else {
					if ( ( bm_Horspool_Order2[(*(unsigned short *)&pbTarget[i+cbPattern-1-1-2])>>3] & (1<<((*(unsigned short *)&pbTarget[i+cbPattern-1-1-2])&0x7)) ) == 0 ) Gulliver = cbPattern-(2-1)-2; else {
						if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) { // This fast check ensures not missing a match (for remainder) when going under 0 in loop below:
							count = cbPattern-4+1;
							while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
								count = count-4;
							if ( count <= 0 ) return(pbTarget+i);
						}
					}
				} else Gulliver = cbPattern-(2-1);
				i = i + Gulliver;
				//GlobalI++; // Comment it, it is only for stats.
			}
			return(NULL);
		} else { // if ( cbPattern<=NeedleThreshold2vs4Decumanus )
			// BMH pseudo-order 4, needle should be >=8+2:
			ulHashPattern = *(uint32_t *)(pbPattern); // First four bytes
			//for (i=0; i < 256*256; i++) {bm_Horspool_Order2[i]=0;}
			for (i=0; i < (256*256)>>3; i++) {bm_Horspool_Order2[i]=0;}
```

```
                    // In line below we "hash" 4bytes to 2bytes i.e. 16bit table, how to compute TOTAL number of BBs, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest
    fox' and Order=4 we have BBs = 11-4+1=8:
                    //"fast"
                    //"aste"
                    //"stes"
                    //"test"
                    //"est "
                    //"st f"
                    //"t fo"
                    //" fox"
                    //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( *(unsigned short *)(pbPattern+i+0) + *(unsigned short *)(pbPattern+i+2) ) & ( (1<<16)-1 )]=1;
                    //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>16)+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
                    // Above line is replaced by next one with better hashing:
                    //for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[( (*(uint32_t *)(pbPattern+i+0)>>(16-1))+(*(uint32_t *)(pbPattern+i+0)&0xFFFF) ) & ( (1<<16)-1 )]=1;
                    for (i=0; i < cbPattern-4+1; i++) bm_Horspool_Order2[_HASH_Order4_KAZE(*(uint32_t *)(pbPattern+i+0))>>3]= bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)(pbPattern+i+0)))>>3] | (1<<((_HASH_Order4_KAZE(*(uint32_t
    *)(pbPattern+i+0)))&0x7));
                    i=0;
                    while (i <= cbTarget-cbPattern) {
                              Gulliver = 1;
                              //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                              // Above line is replaced by next one with better hashing:
                              //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]&0xFFFF) ) & ( (1<<16)-1 )] != 0 ) { // DWORD #1
                              if ( ( bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]))>>3] & (1<<((_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2]))&0x7)) ) != 0 ) { // DWORD #1
                                        //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = cbPattern-(2-1)-2-4; else {
                                        // Above line is replaced in order to strengthen the skip by checking the middle DWORD,if the two DWORDs are 'ab' and 'cd' i.e. [2x][2a][2b][2c][2d] then the middle DWORD is 'bc'.
                                        // The respective offsets (backwards) are: -10/-8/-6/-4 for 'xa'/'ab'/'bc'/'cd'.
                                        //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t
    *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-
    2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                        // Above line is replaced by next one with better hashing:
                                        // When using (16-1) right shifting instead of 16 we will have two different pairs (if they are equal), the highest bit being lost do the job especialy for ASCII texts with no symbols in range
    128-255.
                                        // Example for genomesque pair TT+TT being shifted by (16-1):
                                        // T           = 01010100
                                        // TT          = 01010100 01010100
                                        // TTTT        = 01010100 01010100 01010100 01010100
                                        // TTTT>>16    = 00000000 00000000 01010100 01010100
                                        // TTTT>>(16-1) = 00000000 00000000 10101000 10101000 <--- Due to the left shift by 1, the 8th bits of 1st and 2nd bytes are populated - usually they are 0 for English texts & 'ACGT' data.
                                        //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t
    *)&pbTarget[i+cbPattern-1-1-2-4]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>(16-1))+(*(uint32_t *)&pbTarget[i+cbPattern-1-
    1-2-2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-4-2; else {
                                        if ( \
                                        ( bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]))>>3] & (1<<((_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]))&0x7)) ) == 0 || \
                                        ( bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]))>>3] & (1<<((_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]))&0x7)) ) == 0 || \
                                        ( bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]))>>3] & (1<<((_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]))&0x7)) ) == 0  ) Gulliver =
    cbPattern-(2-1)-2-4-2; else { // Note: Not unbranched as in BYTEwise (above) line.
                                        // Above line is not optimized (several a SHR are used), we have 5 non-overlapping WORDs, or 3 overlapping WORDs, within 4 overlapping DWORDs so:
// [2x][2a][2b][2c][2d]
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]>>16) =     !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                       |
//     DWORD #3                                                                    |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16) =     !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//         DWORD #2                                                                    |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]>>16) =     !SHR to be avoided!      |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//             DWORD #1                                                                    |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                                |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
//
```

```
// So in order to remove 3 SHR instructions the equal extractions are:
// DWORD #4
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) =  !SHR to be avoided! <--
// [2x] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) =                       |
//     DWORD #3                                                                    |
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) =  !SHR to be avoided!   |<--
// [2a] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) = ------------------------  |
//          DWORD #2                                                                   |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) =  !SHR to be avoided!     |<--
// [2b] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) = --------------------------  |
//            DWORD #1                                                                    |
// [2d] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]>>16) =                               |
// [2c] (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF) = ------------------------------
                                        //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-6]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-0]&0xFFFF)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-2]&0xFFFF) ) & ( (1<<16)-1 )] ) < 3 ) Gulliver = cbPattern-(2-1)-2-6; else {
// Since the above Decumanus mumbo-jumbo (3 overlapping lookups vs 2 non-overlapping lookups) is not fast enough we go DuoDecumanus or 3x4:
// [2y][2x][2a][2b][2c][2d]
// DWORD #3
//         DWORD #2
//                 DWORD #1
                                      //if ( ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-4]&0xFFFF) ) & ( (1<<16)-1 )] ) + ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]>>16)+(*(uint32_t *)&pbTarget[i+cbPattern-1-1-2-8]&0xFFFF) ) & ( (1<<16)-1 )] ) < 2 ) Gulliver = cbPattern-(2-1)-2-8; else {
                                              if ( *(uint32_t *)&pbTarget[i] == ulHashPattern) {
                                                      // Order 4 [
                                              // Let's try something "outrageous" like comparing with[out] overlap BBs 4bytes long instead of 1 byte back-to-back:
                                              // Inhere we are using order 4, 'cbPattern - Order + 1' is the number of BBs for text 'cbPattern' bytes long, for example, for cbPattern=11 'fastest fox' and Order=4 we have BBs = 11-4+1=8:
                                              //0:"fast" if the comparison failed here, 'count' is 1; 'Gulliver' is cbPattern-(4-1)-7
                                              //1:"aste" if the comparison failed here, 'count' is 2; 'Gulliver' is cbPattern-(4-1)-6
                                              //2:"stes" if the comparison failed here, 'count' is 3; 'Gulliver' is cbPattern-(4-1)-5
                                              //3:"test" if the comparison failed here, 'count' is 4; 'Gulliver' is cbPattern-(4-1)-4
                                              //4:"est " if the comparison failed here, 'count' is 5; 'Gulliver' is cbPattern-(4-1)-3
                                              //5:"st f" if the comparison failed here, 'count' is 6; 'Gulliver' is cbPattern-(4-1)-2
                                              //6:"t fo" if the comparison failed here, 'count' is 7; 'Gulliver' is cbPattern-(4-1)-1
                                              //7:" fox" if the comparison failed here, 'count' is 8; 'Gulliver' is cbPattern-(4-1)
                                                      count = cbPattern-4+1;
                                                      while ( count > 0 && *(uint32_t *)(pbPattern+count-1) == *(uint32_t *)(&pbTarget[i]+(count-1)) )
                                                              count = count-4;
                                                      if ( count <= 0 ) return(pbTarget+i);
                                                      else {
                                                              // Checking the order 2 pairs in mismatched DWORD, all the 3:
                                                              //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] == 0 ) Gulliver = count; // 1 or bigger, as it should
                                                              //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] == 0 ) Gulliver = count+1; // 1 or bigger, as it should
                                                              //if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1+1]] == 0 ) Gulliver = count+1+1; // 1 or bigger, as it should
                                                      //        if ( bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1]] + bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1]] + bm_Horspool_Order2[*(unsigned short *)&pbTarget[i+count-1+1+1]] < 3 ) Gulliver = count; // 1 or bigger, as it should, THE_MIN(count,count+1,count+1+1)
                                                              // Above compound 'if' guarantees not that Gulliver > 1, an example:
                                                              // Needle:    fastest tax
                                                              // Window: ...fastast tax...
                                                              // After matching ' tax' vs ' tax' and 'fast' vs 'fast' the mismathced DWORD is 'test' vs 'tast':
                                                              // 'tast' when factorized down to order 2 yields: 'ta','as','st' - all the three when summed give 1+1+1=3 i.e. Gulliver remains 1.
                                                              // Roughly speaking, this attempt maybe has its place in worst-case scenarios but not in English text and even not in ACGT data, that's why I commented it in original 'Shockeroo'.
                                                              //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>16)+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1 or bigger, as it should
                                                              // Above line is replaced by next one with better hashing:
                                                              //if ( bm_Horspool_Order2[( (*(uint32_t *)&pbTarget[i+count-1]>>(16-1))+(*(uint32_t *)&pbTarget[i+count-1]&0xFFFF) ) & ( (1<<16)-1 )] == 0 ) Gulliver = count; // 1 or bigger, as it should
                                                              if ( ( bm_Horspool_Order2[(_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+count-1]))>>3] & (1<<((_HASH_Order4_KAZE(*(uint32_t *)&pbTarget[i+count-1]))&0x7)) ) == 0 ) Gulliver = count; // 1 or bigger, as it should
```

```
                                                    }
                                                    // Order 4 ]
                                                }
                                            }
                                        } else Gulliver = cbPattern-(2-1)-2; // -2 because we check the 4 rightmost bytes not 2.
                                        i = i + Gulliver;
                                        //GlobalI++; // Comment it, it is only for stats.
                                    }
                                    return(NULL);
                                } // if ( cbPattern<=NeedleThreshold2vs4Decumanus )
                            } //if ( cbPattern<4 )
                        }
```


Haystack: Wikipedia
Type: XML 50MB dump
Railgun_Ennearch, 32bit Intel v12.1 /O3
Railgun_DecumanusBITified, 32bit Intel v12.1 /O3
Railgun_Swampshine, 32bit Intel v12.1 /O3
Boyer-Moore-Horspool, 32bit Intel v12.1 /O3
Y-axis: Speed in MB/s
X-axis: Needle length in bytes
Test machine: i7 4770K 3500MHz, Windows


Haystack: Homo sapiens chromosome 1, alternate assembly HuRef, whole genome shotgun sequence
File: hs_alt_HuRef_chr1.fa
Type: 'ACGT' 212MB dump
Railgun_Ennearch, 32bit Intel v12.1 /O3
Railgun_DecumanusBITified, 32bit Intel v12.1 /O3
Railgun_Swampshine, 32bit Intel v12.1 /O3
Boyer-Moore-Horspool, 32bit Intel v12.1 /O3
Y-axis: Speed in MB/s
X-axis: Needle length in bytes
Test machine: i7 4770K 3500MHz, Windows

RUNME_the-three-tests.BAT:

```
echo Dumping (in APPEND mode) results to 'Results.txt'...
echo. >> Results.txt
echo 32bit_IntelV12: >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_PREFIXEDandPOSTFIXED_609chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_PREFIXED_309chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_POSTFIXED_309chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_PREFIXEDandPOSTFIXED_17chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_PREFIXED_13chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WCS_32bit_IntelV12.exe<WCS_POSTFIXED_13chars.txt >> Results.txt
echo. >> Results.txt
echo Dumping (in APPEND mode) results to 'Results.txt'...
echo. >> Results.txt
echo 32bit_IntelV12: >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_004chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_005chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_007chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_013chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_021chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_028chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_037chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_051chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_065chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_083chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_098chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_113chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_130chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_140chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_151chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_162chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_201chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_230chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_255chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_589chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_1080chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_1755chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_2686chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_WIKI_32bit_IntelV12.exe<WIKI_3867chars.txt >> Results.txt
echo. >> Results.txt
echo Dumping (in APPEND mode) results to 'Results.txt'...
echo. >> Results.txt
echo 32bit_IntelV12: >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_015chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_018chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_020chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_030chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_040chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_050chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_060chars.txt >> Results.txt
strstr_SHORT-SHOWDOWN_LV_ACGT_32bit_IntelV12.exe<ACGT_070chars.txt >> Results.txt
echo. >> Results.txt
Results.txt
```

Results_i7-4770k.txt:

```
32bit_IntelV12:
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
```

Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
If you have read through the preceding chapters you should have a pretty good idea on how to make
use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
electronic repository of understanding and knowledge on meditation and its techniques. It is also
much more, a complete, world view of the New Man and a new way of life. The purpose of this
CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(609bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/212
Railgun_Swampshine performance: 921KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 422clocks
Railgun_Swampshine precise performance: 902MB/s

Doing Search for Pattern(609bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/19182
Railgun_Ennearch performance: 10KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 38362clocks
Railgun_Ennearch precise performance: 9MB/s

Doing Search for Pattern(609bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/13160
Railgun_DecumanusBITified performance: 14KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 26318clocks
Railgun_DecumanusBITified precise performance: 14MB/s

Doing Search for Pattern(609bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 200000000
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/8823
Boyer_Moore-Horspool performance: 22KB/clock

```
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 17645clocks
Boyer_Moore-Horspool precise performance: 21MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
           If you have read through the preceding chapters you should have a pretty good idea on how to make
           use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
           electronic repository of understanding and knowledge on meditation and its techniques. It is also
           much more, a complete, world view of the New Man and a new way of life. The purpose of this
           CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
           many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/578
Railgun_Swampshine performance: 337KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 1155clocks
Railgun_Swampshine precise performance: 329MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1226
Railgun_Ennearch performance: 159KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 2450clocks
Railgun_Ennearch precise performance: 155MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000000 within next line:
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZSHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1866
Railgun_DecumanusBITified performance: 104KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
```

```
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 3730clocks
Railgun_DecumanusBITified precise performance: 102MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 200000000
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/63
Boyer_Moore-Horspool performance: 3100KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 125clocks
Boyer_Moore-Horspool precise performance: 3048MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
           If you have read through the preceding chapters you should have a pretty good idea on how to make
           use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
           electronic repository of understanding and knowledge on meditation and its techniques. It is also
           much more, a complete, world view of the New Man and a new way of life. The purpose of this
           CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
           many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000300 within next line:
SHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/212
Railgun_Swampshine performance: 921KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 422clocks
Railgun_Swampshine precise performance: 902MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000300 within next line:
SHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/422
Railgun_Ennearch performance: 462KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 843clocks
Railgun_Ennearch precise performance: 451MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 200000300 within next line:
SHAKALAKAZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/844
Railgun_DecumanusBITified performance: 231KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 1686clocks
Railgun_DecumanusBITified precise performance: 225MB/s

Doing Search for Pattern(309bytes) into String(200000609bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 200000300
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/851
Boyer_Moore-Horspool performance: 229KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 1701clocks
Boyer_Moore-Horspool precise performance: 223MB/s


32bit_IntelV12:
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
           If you have read through the preceding chapters you should have a pretty good idea on how to make
           use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
           electronic repository of understanding and knowledge on meditation and its techniques. It is also
           much more, a complete, world view of the New Man and a new way of life. The purpose of this
           CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
           many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(4bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397660 within next line:
}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer
    plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do, East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the
    king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen
    with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway,
    this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes
    described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/10
Railgun_Swampshine performance: 5120KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 40452clocks
Railgun_Swampshine precise performance: 5059MB/s

Doing Search for Pattern(4bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397660 within next line:
}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer
```

```
    plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do, East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the
    king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen
    with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway,
    this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes
    described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/10
Railgun_Ennearch performance: 5120KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 40811clocks
Railgun_Ennearch precise performance: 5015MB/s

Doing Search for Pattern(4bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397660 within next line:
}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer
    plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do, East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the
    king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen
    with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway,
    this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes
    described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/17
Railgun_DecumanusBITified performance: 3011KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 65911clocks
Railgun_DecumanusBITified precise performance: 3105MB/s

Doing Search for Pattern(4bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52397660
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/39
Boyer_Moore-Horspool performance: 1312KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 156812clocks
Boyer_Moore-Horspool precise performance: 1305MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(5bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52398302 within next line:
```

```
bs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a
    low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry''
    back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand
    out to prove they are all winners.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/8
Railgun_Swampshine performance: 6400KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TOtalRoughSearchTime (lesser-the-better): 30858clocks
Railgun_Swampshine precise performance: 6632MB/s

Doing Search for Pattern(5bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52398302 within next line:
bs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a
    low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry''
    back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand
    out to prove they are all winners.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/8
Railgun_Ennearch performance: 6400KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 31248clocks
Railgun_Ennearch precise performance: 6550MB/s

Doing Search for Pattern(5bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52398302 within next line:
bs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a
    low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry''
    back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by showing his or her hand, as it now contains only high trumps and there's no need to play the hand
    out to prove they are all winners.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/13
Railgun_DecumanusBITified performance: 3938KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 49812clocks
Railgun_DecumanusBITified precise performance: 4109MB/s

Doing Search for Pattern(5bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52398302
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/31
Boyer_Moore-Horspool performance: 1651KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 124832clocks
Boyer_Moore-Horspool precise performance: 1639MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
```

```
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(7bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/9
Railgun_Swampshine performance: 5688KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 33151clocks
Railgun_Swampshine precise performance: 6040MB/s

Doing Search for Pattern(7bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/9
Railgun_Ennearch performance: 5688KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 35023clocks
Railgun_Ennearch precise performance: 5717MB/s

Doing Search for Pattern(7bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/11
Railgun_DecumanusBITified performance: 4654KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 42324clocks
Railgun_DecumanusBITified precise performance: 4731MB/s

Doing Search for Pattern(7bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 51265299
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/30
Boyer_Moore-Horspool performance: 1706KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 121850clocks
```

```
Boyer_Moore-Horspool precise performance: 1643MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(13bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/7
Railgun_Swampshine performance: 7314KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 24976clocks
Railgun_Swampshine precise performance: 8017MB/s

Doing Search for Pattern(13bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/7
Railgun_Ennearch performance: 7314KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 26272clocks
Railgun_Ennearch precise performance: 7622MB/s

Doing Search for Pattern(13bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 51265299 within next line:
In [[vertebrate]]s, vigorously contracting [[skeletal muscle]]s (during weightlifting or sprinting, for example) do not receive enough oxygen to meet the energy demand, and so they shift to [[Fermentation (biochemistry)|anaerobic
    metabolism]], converting glucose to lactate. The [[liver]] regenerates the glucose, using a process called [[gluconeogenesis]]. This process is not quite the opposite of glycolysis, and actually requires three times the amount of
    energy gained from glycolysis (six molecules of ATP are used, compared to the two gained in glycolysis). Analogous to the above reactions, the glucose produced can then undergo glycolysis in tissues that need energy, be stored as
    glycogen (or starch in plants), or be converted to other monosaccharides or joined into di- or oligosaccharides. The combined pathways of glycolysis during exercise, lactate's crossing via the bloodstream to the liver, subsequent
    gluconeogenesis and release of glucose into the bloodstream is called the [[Cori cycle]].&lt;ref&gt;Fromm and Hargrove (2012), pp. 183–194.&lt;/ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/7
```

```
Railgun_DecumanusBITified performance: 7314KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 27145clocks
Railgun_DecumanusBITified precise performance: 7377MB/s

Doing Search for Pattern(13bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 51265299
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/18
Boyer_Moore-Horspool performance: 2844KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 71106clocks
Boyer_Moore-Horspool precise performance: 2816MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(21bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/4
Railgun_Swampshine performance: 12800KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 13666clocks
Railgun_Swampshine precise performance: 14985MB/s

Doing Search for Pattern(21bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/4
Railgun_Ennearch performance: 12800KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 13854clocks
Railgun_Ennearch precise performance: 14782MB/s

Doing Search for Pattern(21bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
```

```
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/5
Railgun_DecumanusBITified performance: 10240KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 16724clocks
Railgun_DecumanusBITified precise performance: 12245MB/s

Doing Search for Pattern(21bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/14
Boyer_Moore-Horspool performance: 3657KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 53291clocks
Boyer_Moore-Horspool precise performance: 3843MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(28bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 12263clocks
Railgun_Swampshine precise performance: 16700MB/s

Doing Search for Pattern(28bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/4
Railgun_Ennearch performance: 12800KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 12434clocks
Railgun_Ennearch precise performance: 16470MB/s

Doing Search for Pattern(28bytes) into String(52428800bytes) as-one-line ...
```

```
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/4
Railgun_DecumanusBITified performance: 12800KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 14010clocks
Railgun_DecumanusBITified precise performance: 14617MB/s

Doing Search for Pattern(28bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/11
Boyer_Moore-Horspool performance: 4654KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 44820clocks
Boyer_Moore-Horspool precise performance: 4569MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(37bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11389clocks
Railgun_Swampshine precise performance: 17982MB/s

Doing Search for Pattern(37bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 11530clocks
Railgun_Ennearch precise performance: 17762MB/s
```

```
Doing Search for Pattern(37bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/4
Railgun_DecumanusBITified performance: 12800KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 12496clocks
Railgun_DecumanusBITified precise performance: 16389MB/s

Doing Search for Pattern(37bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/9
Boyer_Moore-Horspool performance: 5688KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 36521clocks
Boyer_Moore-Horspool precise performance: 5607MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(51bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 10843clocks
Railgun_Swampshine precise performance: 18887MB/s

Doing Search for Pattern(51bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
```

```
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 10937clocks
Railgun_Ennearch precise performance: 18725MB/s

Doing Search for Pattern(51bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 11295clocks
Railgun_DecumanusBITified precise performance: 18131MB/s

Doing Search for Pattern(51bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/8
Boyer_Moore-Horspool performance: 6400KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 31466clocks
Boyer_Moore-Horspool precise performance: 6508MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(65bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 10812clocks
Railgun_Swampshine precise performance: 18941MB/s

Doing Search for Pattern(65bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
```

```
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 10859clocks
Railgun_Ennearch precise performance: 18859MB/s

Doing Search for Pattern(65bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10702clocks
Railgun_DecumanusBITified precise performance: 19136MB/s

Doing Search for Pattern(65bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/8
Boyer_Moore-Horspool performance: 6400KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 29428clocks
Boyer_Moore-Horspool precise performance: 6959MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(83bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11548clocks
Railgun_Swampshine precise performance: 17734MB/s

Doing Search for Pattern(83bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
```

```
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 10883clocks
Railgun_Ennearch precise performance: 18818MB/s

Doing Search for Pattern(83bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10812clocks
Railgun_DecumanusBITified precise performance: 18941MB/s

Doing Search for Pattern(83bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/7
Boyer_Moore-Horspool performance: 7314KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 26019clocks
Boyer_Moore-Horspool precise performance: 7871MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(98bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11511clocks
Railgun_Swampshine precise performance: 17791MB/s

Doing Search for Pattern(98bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
```

```
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 10420clocks
Railgun_Ennearch precise performance: 19654MB/s

Doing Search for Pattern(98bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10808clocks
Railgun_DecumanusBITified precise performance: 18948MB/s

Doing Search for Pattern(98bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/7
Boyer_Moore-Horspool performance: 7314KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 25868clocks
Boyer_Moore-Horspool precise performance: 7917MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(113bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11797clocks
Railgun_Swampshine precise performance: 17360MB/s
```

```
Doing Search for Pattern(113bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 9845clocks
Railgun_Ennearch precise performance: 20802MB/s

Doing Search for Pattern(113bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10771clocks
Railgun_DecumanusBITified precise performance: 19013MB/s

Doing Search for Pattern(113bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/7
Boyer_Moore-Horspool performance: 7314KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 25137clocks
Boyer_Moore-Horspool precise performance: 8147MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(130bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11712clocks
```

```
Railgun_Swampshine precise performance: 17486MB/s

Doing Search for Pattern(130bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 8982clocks
Railgun_Ennearch precise performance: 22800MB/s

Doing Search for Pattern(130bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10992clocks
Railgun_DecumanusBITified precise performance: 18631MB/s

Doing Search for Pattern(130bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/6
Boyer_Moore-Horspool performance: 8533KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 22540clocks
Boyer_Moore-Horspool precise performance: 9085MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(140bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
```

```
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TOtalRoughSearchTime (lesser-the-better): 11830clocks
Railgun_Swampshine precise performance: 17311MB/s

Doing Search for Pattern(140bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 8579clocks
Railgun_Ennearch precise performance: 23872MB/s

Doing Search for Pattern(140bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 11424clocks
Railgun_DecumanusBITified precise performance: 17926MB/s

Doing Search for Pattern(140bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/6
Boyer_Moore-Horspool performance: 8533KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 21919clocks
Boyer_Moore-Horspool precise performance: 9343MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(151bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
```

```
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11996clocks
Railgun_Swampshine precise performance: 17072MB/s

Doing Search for Pattern(151bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/3
Railgun_Ennearch performance: 17066KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 8539clocks
Railgun_Ennearch precise performance: 23983MB/s

Doing Search for Pattern(151bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 11420clocks
Railgun_DecumanusBITified precise performance: 17933MB/s

Doing Search for Pattern(151bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/6
Boyer_Moore-Horspool performance: 8533KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 23837clocks
Boyer_Moore-Horspool precise performance: 8591MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(162bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
```

```
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11554clocks
Railgun_Swampshine precise performance: 17725MB/s

Doing Search for Pattern(162bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/2
Railgun_Ennearch performance: 25600KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 7718clocks
Railgun_Ennearch precise performance: 26535MB/s

Doing Search for Pattern(162bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52428369 within next line:
In [[vertebrate]]s, it is composed of [[blood cells]] suspended in [[blood plasma]]. Plasma, which constitutes 55% of blood fluid, is mostly water (92% by volume),&lt;ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 10315clocks
Railgun_DecumanusBITified precise performance: 19854MB/s

Doing Search for Pattern(162bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52428369
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/6
Boyer_Moore-Horspool performance: 8533KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 22573clocks
Boyer_Moore-Horspool precise performance: 9072MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(201bytes) into String(52428800bytes) as-one-line ...
```

```
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52422754 within next line:
Cold moulding is similar to FRP in as much as it involves the use of epoxy or polyester resins, but the structural component is wood instead of fibreglass. In cold moulding very thin strips of wood are laid over a form or mould in layers.
    This layer is then coated with resin and another directionally alternating layer is laid on top. In some processes the subsequent layers are stapled or otherwise mechanically fastened to the previous layers, but in other processes the
    layers are weighted or even vacuum bagged to hold layers together while the resin sets. Layers are built up thus to create the required thickness of hull.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/3
Railgun_Swampshine performance: 17066KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 11405clocks
Railgun_Swampshine precise performance: 17954MB/s

Doing Search for Pattern(201bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52422754 within next line:
Cold moulding is similar to FRP in as much as it involves the use of epoxy or polyester resins, but the structural component is wood instead of fibreglass. In cold moulding very thin strips of wood are laid over a form or mould in layers.
    This layer is then coated with resin and another directionally alternating layer is laid on top. In some processes the subsequent layers are stapled or otherwise mechanically fastened to the previous layers, but in other processes the
    layers are weighted or even vacuum bagged to hold layers together while the resin sets. Layers are built up thus to create the required thickness of hull.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/2
Railgun_Ennearch performance: 25600KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 6458clocks
Railgun_Ennearch precise performance: 31708MB/s

Doing Search for Pattern(201bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52422754 within next line:
Cold moulding is similar to FRP in as much as it involves the use of epoxy or polyester resins, but the structural component is wood instead of fibreglass. In cold moulding very thin strips of wood are laid over a form or mould in layers.
    This layer is then coated with resin and another directionally alternating layer is laid on top. In some processes the subsequent layers are stapled or otherwise mechanically fastened to the previous layers, but in other processes the
    layers are weighted or even vacuum bagged to hold layers together while the resin sets. Layers are built up thus to create the required thickness of hull.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/3
Railgun_DecumanusBITified performance: 17066KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 8754clocks
Railgun_DecumanusBITified precise performance: 23392MB/s

Doing Search for Pattern(201bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52422754
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/8
Boyer_Moore-Horspool performance: 6400KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 29324clocks
Boyer_Moore-Horspool precise performance: 6983MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.
```

```
Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(230bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424290 within next line:
An early uncommon means of boat propulsion was the [[water caterpillar]].  This moved a series of paddles on chains along the bottom of the boat to propel it over the water and preceded the development of [[tracked
    vehicle]]s.&lt;ref&gt;&quot;The Caterpillar Is Now Being Applied to Ships&quot;, ''[[Popular Science]]'' monthly, December 1918, page 68, Scanned by Google Books: http://books.google.com/books?id=EikDAAAAMBAJ&amp;pg=PA68&lt;/ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/4
Railgun_Swampshine performance: 12800KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 13550clocks
Railgun_Swampshine precise performance: 15113MB/s

Doing Search for Pattern(230bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424290 within next line:
An early uncommon means of boat propulsion was the [[water caterpillar]].  This moved a series of paddles on chains along the bottom of the boat to propel it over the water and preceded the development of [[tracked
    vehicle]]s.&lt;ref&gt;&quot;The Caterpillar Is Now Being Applied to Ships&quot;, ''[[Popular Science]]'' monthly, December 1918, page 68, Scanned by Google Books: http://books.google.com/books?id=EikDAAAAMBAJ&amp;pg=PA68&lt;/ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/2
Railgun_Ennearch performance: 25600KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 6187clocks
Railgun_Ennearch precise performance: 33098MB/s

Doing Search for Pattern(230bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424290 within next line:
An early uncommon means of boat propulsion was the [[water caterpillar]].  This moved a series of paddles on chains along the bottom of the boat to propel it over the water and preceded the development of [[tracked
    vehicle]]s.&lt;ref&gt;&quot;The Caterpillar Is Now Being Applied to Ships&quot;, ''[[Popular Science]]'' monthly, December 1918, page 68, Scanned by Google Books: http://books.google.com/books?id=EikDAAAAMBAJ&amp;pg=PA68&lt;/ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/2
Railgun_DecumanusBITified performance: 25600KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 8025clocks
Railgun_DecumanusBITified precise performance: 25518MB/s

Doing Search for Pattern(230bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52424290
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/7
Boyer_Moore-Horspool performance: 7314KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 26946clocks
Boyer_Moore-Horspool precise performance: 7599MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
```

```
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(255bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424775 within next line:
A floating boat [[displacement (fluid)|displaces]] its weight in water. The material of the boat hull may be denser than water, but if this is the case then it forms only the outer layer. If the boat floats, the mass of the boat (plus
    contents) ''as a whole'' divided by the volume ''below the waterline'' is equal to the [[density]] of water (1&amp;nbsp;kg/l). If weight is added to the boat, the volume below the waterline will increase to keep the weight balance
    equal, and so the boat sinks a little to compensate.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/4
Railgun_Swampshine performance: 12800KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 13920clocks
Railgun_Swampshine precise performance: 14711MB/s

Doing Search for Pattern(255bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424775 within next line:
A floating boat [[displacement (fluid)|displaces]] its weight in water. The material of the boat hull may be denser than water, but if this is the case then it forms only the outer layer. If the boat floats, the mass of the boat (plus
    contents) ''as a whole'' divided by the volume ''below the waterline'' is equal to the [[density]] of water (1&amp;nbsp;kg/l). If weight is added to the boat, the volume below the waterline will increase to keep the weight balance
    equal, and so the boat sinks a little to compensate.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/2
Railgun_Ennearch performance: 25600KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 5131clocks
Railgun_Ennearch precise performance: 39911MB/s

Doing Search for Pattern(255bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52424775 within next line:
A floating boat [[displacement (fluid)|displaces]] its weight in water. The material of the boat hull may be denser than water, but if this is the case then it forms only the outer layer. If the boat floats, the mass of the boat (plus
    contents) ''as a whole'' divided by the volume ''below the waterline'' is equal to the [[density]] of water (1&amp;nbsp;kg/l). If weight is added to the boat, the volume below the waterline will increase to keep the weight balance
    equal, and so the boat sinks a little to compensate.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/2
Railgun_DecumanusBITified performance: 25600KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 6957clocks
Railgun_DecumanusBITified precise performance: 29435MB/s

Doing Search for Pattern(255bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52424775
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/7
Boyer_Moore-Horspool performance: 7314KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 26938clocks
Boyer_Moore-Horspool precise performance: 7602MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
```

```
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(589bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/7
Railgun_Swampshine performance: 7314KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 28586clocks
Railgun_Swampshine precise performance: 7163MB/s

Doing Search for Pattern(589bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1
Railgun_Ennearch performance: 51200KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 1657clocks
Railgun_Ennearch precise performance: 123574MB/s

Doing Search for Pattern(589bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
```

```
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1
Railgun_DecumanusBITified performance: 51200KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 2138clocks
Railgun_DecumanusBITified precise performance: 95773MB/s

Doing Search for Pattern(589bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52419431
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/6
Boyer_Moore-Horspool performance: 8533KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 24463clocks
Boyer_Moore-Horspool precise performance: 8370MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(1080bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire
    framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or
    corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden
    boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in
    many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available
    that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/14
Railgun_Swampshine performance: 3657KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 55147clocks
Railgun_Swampshine precise performance: 3713MB/s

Doing Search for Pattern(1080bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire
    framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or
    corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden
    boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in
    many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available
```

    that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1
Railgun_Ennearch performance: 51200KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 1043clocks
Railgun_Ennearch precise performance: 196321MB/s

Doing Search for Pattern(1080bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52419431 within next line:
By the mid-19th century, many boats had been built with iron or steel frames but still planked in wood. In 1855 ferro-cement boat construction was patented by the French as Ferciment. This is a system by which a steel or iron wire framework is built in the shape of a boat's hull and covered ([[trowell]]ed) over with cement. Reinforced with bulkheads and other internal structure, it is strong but heavy, easily repaired, and, if sealed properly, will not leak or corrode. These materials and methods were copied all over the world, and have faded in and out of popularity to the present. As the forests of Britain and Europe continued to be over-harvested to supply the keels of larger wooden boats, and the [[Bessemer process]] ([[patent]]ed in 1855) cheapened the cost of steel, steel ships and boats began to be more common. By the 1930s boats built of all steel from frames to plating were seen replacing wooden boats in many industrial uses, even the fishing fleets. Private recreational boats in steel are uncommon. In the mid 20th century [[aluminium]] gained popularity. Though much more expensive than steel, there are now aluminium alloys available that will not corrode in salt water, and an aluminium boat built to similar load carrying standards could be built lighter than steel.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1
Railgun_DecumanusBITified performance: 51200KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 1150clocks
Railgun_DecumanusBITified precise performance: 178054MB/s

Doing Search for Pattern(1080bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52419431
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/5
Boyer_Moore-Horspool performance: 10240KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 18443clocks
Boyer_Moore-Horspool precise performance: 11102MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(1755bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397285 within next line:
After considering the cards, the declarer directs dummy (North) to play a small spade. East plays ''low'' (small card) and South takes the {{Spades}}A, gaining the ''lead''. (South may also elect to ''[[duck (bridge)|duck]]'', but for the purpose of this example, let us assume South wins the {{Spades}}A at trick 1). South proceeds by ''drawing trump'', leading the {{Hearts}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do, East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by

```
    leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the
    remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff
    (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by
    showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/19
Railgun_Swampshine performance: 2694KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 75161clocks
Railgun_Swampshine precise performance: 2723MB/s

Doing Search for Pattern(1755bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397285 within next line:
After considering the cards, the declarer directs dummy (North) to play a small spade. East plays ''low'' (small card) and South takes the {{Spades}}A, gaining the ''lead''. (South may also elect to ''[[duck (bridge)|duck]]'', but for the
    purpose of this example, let us assume South wins the {{Spades}}A at trick 1). South proceeds by ''drawing trump'', leading the {{Hearts}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and
    then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do,
    East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by
    leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the
    remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff
    (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by
    showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1
Railgun_Ennearch performance: 51200KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 713clocks
Railgun_Ennearch precise performance: 287063MB/s

Doing Search for Pattern(1755bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52397285 within next line:
After considering the cards, the declarer directs dummy (North) to play a small spade. East plays ''low'' (small card) and South takes the {{Spades}}A, gaining the ''lead''. (South may also elect to ''[[duck (bridge)|duck]]'', but for the
    purpose of this example, let us assume South wins the {{Spades}}A at trick 1). South proceeds by ''drawing trump'', leading the {{Hearts}}K. West decides there is no benefit to holding back, and so wins the trick with the ace, and
    then cashes the {{Spades}}Q. For fear of conceding a ''[[ruff and discard]]'', West plays the {{Diams}}2 instead of another spade. Declarer plays low from the table, and East scores the {{Diams}}Q. Not having anything better to do,
    East returns the remaining trump, taken in South's hand. The trumps now accounted for, South can now execute the finesse, perhaps trapping the king as planned. South ''enters'' the dummy (i.e. wins a trick in the dummy's hand) by
    leading a low diamond, using dummy's {{Diams}}A to win the trick, and leads the {{Clubs}}Q from dummy to the next trick. East ''covers'' the queen with the king, and South takes the trick with the Ace, and proceeds by ''cashing'' the
    remaining ''master'' {{Clubs}}J. (If East doesn't play the king, then South will play a low club from South's hand and the queen will win anyway, this being the essence of the finesse). The game is now safe: South ''[[ruff
    (cards)|ruffs]]'' a small club with a dummy's trump, then ruffs a diamond in hand for an ''entry'' back, and ruffs the last club in dummy (sometimes described as a ''[[crossruff]]''). Finally, South ''claims'' the remaining tricks by
    showing his or her hand, as it now contains only high trumps and there's no need to play the hand out to prove they are all winners.
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1
Railgun_DecumanusBITified performance: 51200KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 765clocks
Railgun_DecumanusBITified precise performance: 267550MB/s

Doing Search for Pattern(1755bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52397285
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/4
Boyer_Moore-Horspool performance: 12800KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 14630clocks
Boyer_Moore-Horspool precise performance: 13990MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
```

```
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(2686bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52200088 within next line:
Franklin [[bequest|bequeathed]] £1,000 (about $4,400 at the time, or about $112,000 in 2011 dollars&lt;ref&gt;[http://www.measuringworth.com/uscompare/relativevalue.php Measuring Worth] Select $4,400 and 1790 and 2011 in online
    calculator&lt;/ref&gt;) each to the cities of Boston and Philadelphia, in trust to gather interest for 200 years. The trust began in 1785 when the French mathematician [[Charles-Joseph Mathon de la Cour]], who admired Franklin
    greatly, wrote a friendly [[parody]] of Franklin's &quot;Poor Richard's Almanack&quot; called &quot;Fortunate Richard.&quot; The main character leaves a smallish amount of money in his will, five lots of 100 ''[[French
    livre|livres]]'', to collect interest over one, two, three, four or five full centuries, with the resulting astronomical sums to be spent on impossibly elaborate utopian projects.&lt;ref&gt;Richard Price. ''Observations on the
    Importance of the American Revolution, and the Means of Making it a Benefit to the World. To which is added, a Letter from M. Turgot, late Comptroller-General of the Finances of France: with an Appendix, containing a Translation of
    the Will of M. Fortuné Ricard, lately published in France.'' London: T. Cadell, 1785.&lt;/ref&gt; Franklin, who was 79 years old at the time, wrote thanking him for a great idea and telling him that he had decided to leave a bequest
    of 1,000&amp;nbsp;pounds each to his native Boston and his adopted Philadelphia. By 1990, more than $2,000,000 had accumulated in Franklin's Philadelphia trust, which had loaned the money to local residents. From 1940 to 1990, the
    money was used mostly for mortgage loans. When the trust came due, Philadelphia decided to spend it on scholarships for local high school students. Franklin's Boston trust fund accumulated almost $5,000,000 during that same time; at
    the end of its first 100 years a portion was allocated to help establish a [[Vocational school|trade school]] that became the [[Benjamin Franklin Institute of Technology|Franklin Institute of Boston]] and the whole fund was later
    dedicated to supporting this institute.&lt;ref&gt;{{cite web|url=http://www.mathsci.appstate.edu/~sjg/class/1010/wc/finance/franklin1.html |title=Excerpt from Philadelphia Inquirer article by Clark De Leon
    |publisher=Mathsci.appstate.edu |date=February 7, 1993 |accessdate=September 21, 2009}}&lt;/ref&gt;&lt;ref&gt;{{cite web|url=http://www.bfit.edu/aboutus/history.php |title=History of the Benjamin Franklin Institute of Technology
    |publisher=Bfit.edu |accessdate=September 21, 2009| archiveurl = http://web.archive.org/web/20080731130624/http://www.bfit.edu/aboutus/history.php| archivedate = July 31, 2008}}&lt;/ref&gt;
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/63
Railgun_Swampshine performance: 812KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 254796clocks
Railgun_Swampshine precise performance: 800MB/s

Doing Search for Pattern(2686bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52200088 within next line:
Franklin [[bequest|bequeathed]] £1,000 (about $4,400 at the time, or about $112,000 in 2011 dollars&lt;ref&gt;[http://www.measuringworth.com/uscompare/relativevalue.php Measuring Worth] Select $4,400 and 1790 and 2011 in online
    calculator&lt;/ref&gt;) each to the cities of Boston and Philadelphia, in trust to gather interest for 200 years. The trust began in 1785 when the French mathematician [[Charles-Joseph Mathon de la Cour]], who admired Franklin
    greatly, wrote a friendly [[parody]] of Franklin's &quot;Poor Richard's Almanack&quot; called &quot;Fortunate Richard.&quot; The main character leaves a smallish amount of money in his will, five lots of 100 ''[[French
    livre|livres]]'', to collect interest over one, two, three, four or five full centuries, with the resulting astronomical sums to be spent on impossibly elaborate utopian projects.&lt;ref&gt;Richard Price. ''Observations on the
    Importance of the American Revolution, and the Means of Making it a Benefit to the World. To which is added, a Letter from M. Turgot, late Comptroller-General of the Finances of France: with an Appendix, containing a Translation of
    the Will of M. Fortuné Ricard, lately published in France.'' London: T. Cadell, 1785.&lt;/ref&gt; Franklin, who was 79 years old at the time, wrote thanking him for a great idea and telling him that he had decided to leave a bequest
    of 1,000&amp;nbsp;pounds each to his native Boston and his adopted Philadelphia. By 1990, more than $2,000,000 had accumulated in Franklin's Philadelphia trust, which had loaned the money to local residents. From 1940 to 1990, the
    money was used mostly for mortgage loans. When the trust came due, Philadelphia decided to spend it on scholarships for local high school students. Franklin's Boston trust fund accumulated almost $5,000,000 during that same time; at
    the end of its first 100 years a portion was allocated to help establish a [[Vocational school|trade school]] that became the [[Benjamin Franklin Institute of Technology|Franklin Institute of Boston]] and the whole fund was later
    dedicated to supporting this institute.&lt;ref&gt;{{cite web|url=http://www.mathsci.appstate.edu/~sjg/class/1010/wc/finance/franklin1.html |title=Excerpt from Philadelphia Inquirer article by Clark De Leon
    |publisher=Mathsci.appstate.edu |date=February 7, 1993 |accessdate=September 21, 2009}}&lt;/ref&gt;&lt;ref&gt;{{cite web|url=http://www.bfit.edu/aboutus/history.php |title=History of the Benjamin Franklin Institute of Technology
    |publisher=Bfit.edu |accessdate=September 21, 2009| archiveurl = http://web.archive.org/web/20080731130624/http://www.bfit.edu/aboutus/history.php| archivedate = July 31, 2008}}&lt;/ref&gt;
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1
Railgun_Ennearch performance: 51200KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 895clocks
Railgun_Ennearch precise performance: 227827MB/s

Doing Search for Pattern(2686bytes) into String(52428800bytes) as-one-line ...
```

WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52200088 within next line:
Franklin [[bequest|bequeathed]] £1,000 (about $4,400 at the time, or about $112,000 in 2011 dollars<ref>[http://www.measuringworth.com/uscompare/relativevalue.php Measuring Worth] Select $4,400 and 1790 and 2011 in online calculator</ref>) each to the cities of Boston and Philadelphia, in trust to gather interest for 200 years. The trust began in 1785 when the French mathematician [[Charles-Joseph Mathon de la Cour]], who admired Franklin greatly, wrote a friendly [[parody]] of Franklin's "Poor Richard's Almanack" called "Fortunate Richard." The main character leaves a smallish amount of money in his will, five lots of 100 ''[[French livre|livres]]'', to collect interest over one, two, three, four or five full centuries, with the resulting astronomical sums to be spent on impossibly elaborate utopian projects.<ref>Richard Price. ''Observations on the Importance of the American Revolution, and the Means of Making it a Benefit to the World. To which is added, a Letter from M. Turgot, late Comptroller-General of the Finances of France: with an Appendix, containing a Translation of the Will of M. Fortuné Ricard, lately published in France.'' London: T. Cadell, 1785.</ref> Franklin, who was 79 years old at the time, wrote thanking him for a great idea and telling him that he had decided to leave a bequest of 1,000 pounds each to his native Boston and his adopted Philadelphia. By 1990, more than $2,000,000 had accumulated in Franklin's Philadelphia trust, which had loaned the money to local residents. From 1940 to 1990, the money was used mostly for mortgage loans. When the trust came due, Philadelphia decided to spend it on scholarships for local high school students. Franklin's Boston trust fund accumulated almost $5,000,000 during that same time; at the end of its first 100 years a portion was allocated to help establish a [[Vocational school|trade school]] that became the [[Benjamin Franklin Institute of Technology|Franklin Institute of Boston]] and the whole fund was later dedicated to supporting this institute.<ref>{{cite web|url=http://www.mathsci.appstate.edu/~sjg/class/1010/wc/finance/franklin1.html |title=Excerpt from Philadelphia Inquirer article by Clark De Leon |publisher=Mathsci.appstate.edu |date=February 7, 1993 |accessdate=September 21, 2009}}</ref><ref>{{cite web|url=http://www.bfit.edu/aboutus/history.php |title=History of the Benjamin Franklin Institute of Technology |publisher=Bfit.edu |accessdate=September 21, 2009| archiveurl = http://web.archive.org/web/20080731130624/http://www.bfit.edu/aboutus/history.php| archivedate = July 31, 2008}}</ref>
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1
Railgun_DecumanusBITified performance: 51200KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 849clocks
Railgun_DecumanusBITified precise performance: 240171MB/s

Doing Search for Pattern(2686bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52200088
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/5
Boyer_Moore-Horspool performance: 10240KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 17631clocks
Boyer_Moore-Horspool precise performance: 11565MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(3867bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52180092 within next line:
Although Franklin's parents had intended for him to have a career in the Church,<ref name="autobio" /> Franklin as a young man adopted the Enlightenment religious belief in deism, that God's truths can be found entirely through nature and reason.<ref>Isaacson, 2003, p. 46</ref> "I soon became a thorough Deist."<ref>Franklin, Benjamin. [http://www.usgennet.org/usa/topic/preservation/bios/franklin/chpt4.htm ''Benjamin Franklin's Autobiography''.] Chapter IV. reprinted on USGenNet.org.</ref> As a young man he rejected Christian dogma in a 1725 pamphlet ''[[A Dissertation on Liberty and Necessity, Pleasure and Pain]]'',<ref>{{cite web|url=http://www.historycarper.com/resources/twobf1/m7.htm |title=A Dissertation on Liberty and Necessity, Pleasure and Pain |publisher=Historycarper.com |accessdate=September 21, 2009}}</ref> which he later saw as an embarrassment,<ref name="Isaacson45">{{cite book|url=http://books.google.com/?id=oIW915dDMBwC&pg=PA45&lpg=PA45&dq=%22A+Dissertation+on+Liberty+and+Necessity,+Pleasure+and+Pain%22+%22Benjamin+Franklin%22+embarrassment|title=Isaacson, 2003, p. 45 |publisher=Google Books |date=November 30, 2004 |accessdate=September 21, 2009|isbn=978-0-684-80761-4|author1=Isaacson, Walter}}</ref> while simultaneously asserting that God is "all wise, [[Omnibenevolence|all good]], [[Omnipotence|all

powerful]]."<ref name="Isaacson45" /> He defended his rejection of religious dogma with these words: "I think opinions should be judged by their influences and effects; and if a man holds none that tend to make him less virtuous or more vicious, it may be concluded that he holds none that are dangerous, which I hope is the case with me." After the disillusioning experience of seeing the decay in his own moral standards, and those of two friends in London whom he had converted to Deism, Franklin turned back to a belief in the importance of organized religion, on the pragmatic grounds that without God and organized churches, man will not be good.<ref>Isaacson, 2003, p 46, 486</ref> Moreover, because of his proposal that [[Prayer in Christianity|prayers]] be said in the [[Constitutional Convention of 1787]], many have contended that in his later life, Franklin became a [[pious]] Christian.<ref name="Pious">{{cite book|author=Henry Louis Mencken, George Jean Nathan|url=http://books.google.com/books?id=LqJUonES6m8C&q=benjamin+franklin+identify+christian+religion&dq=benjamin+franklin+identify+christian+religion&hl=en&ei=dMHMTYaOHILeOQHKpYjeBA&sa=X&oi=book_result&ct=result&resnum=2&ved=0CC4Q6AEwATge|title=The American Mercury, Volume 8|publisher=Garber Communications|quote=It is well known that in his youth Benjamin Franklin was a thorough-going Deist, but because he proposed that prayers be said in the Constitution Convention of 1787 many have contended that in later life he became a pious Christian.|date=October 19, 2009}}</ref><ref name="Faith">{{cite book|author=Ralph Frasca|url=http://books.google.com/books?id=CY2UVzcU5l0C&pg=PA40&dq=benjamin+franklin+christian+or+deist&hl=en&ei=wEXLTe7uEY-2twf-rIXyBw&sa=X&oi=book_result&ct=result&resnum=7&ved=0CEcQ6AEwBg#v=onepage&q&f=false|title=Benjamin Franklin's Printing Network: Disseminating Virtue in Early America|publisher=[[University of Missouri Press]]|quote=Despite being raised a Puritan of the Congregationalist stripe by his parents, who "brought me through my Childhood piously in the Dissenting Way," Franklin recalled, he abandoned that denomination, briefly embraced deism, and finally became a non-denominational Protestant Christian. |date=October 19, 2009}}</ref>
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/101
Railgun_Swampshine performance: 506KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 413661clocks
Railgun_Swampshine precise performance: 492MB/s

Doing Search for Pattern(3867bytes) into String(5242880bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52180092 within next line:
Although Franklin's parents had intended for him to have a career in the Church,<ref name="autobio" /> Franklin as a young man adopted the Enlightenment religious belief in deism, that God’s truths can be found entirely through nature and reason.<ref>Isaacson, 2003, p. 46</ref> "I soon became a thorough Deist."<ref>Franklin, Benjamin. [http://www.usgennet.org/usa/topic/preservation/bios/franklin/chpt4.htm ''Benjamin Franklin's Autobiography''.] Chapter IV. reprinted on USGenNet.org.</ref> As a young man he rejected Christian dogma in a 1725 pamphlet ''[[A Dissertation on Liberty and Necessity, Pleasure and Pain]]'',<ref>{{cite web|url=http://www.historycarper.com/resources/twobf1/m7.htm |title=A Dissertation on Liberty and Necessity, Pleasure and Pain |publisher=Historycarper.com |accessdate=September 21, 2009}}</ref> which he later saw as an embarrassment,<ref name="Isaacson45">{{cite book|url=http://books.google.com/?id=oIW915dDMBwC&pg=PA45&lpg=PA45&dq=%22A+Dissertation+on+Liberty+and+Necessity,+Pleasure+and+Pain%22+%22Benjamin+Franklin%22+embarrassment|title=Isaacson, 2003, p. 45 |publisher=Google Books |date=November 30, 2004 |accessdate=September 21, 2009|isbn=978-0-684-80761-4|author1=Isaacson, Walter}}</ref> while simultaneously asserting that God is "all wise, [[Omnibenevolence|all good]], [[Omnipotence|all powerful]]."<ref name="Isaacson45" /> He defended his rejection of religious dogma with these words: "I think opinions should be judged by their influences and effects; and if a man holds none that tend to make him less virtuous or more vicious, it may be concluded that he holds none that are dangerous, which I hope is the case with me." After the disillusioning experience of seeing the decay in his own moral standards, and those of two friends in London whom he had converted to Deism, Franklin turned back to a belief in the importance of organized religion, on the pragmatic grounds that without God and organized churches, man will not be good.<ref>Isaacson, 2003, p 46, 486</ref> Moreover, because of his proposal that [[Prayer in Christianity|prayers]] be said in the [[Constitutional Convention of 1787]], many have contended that in his later life, Franklin became a [[pious]] Christian.<ref name="Pious">{{cite book|author=Henry Louis Mencken, George Jean Nathan|url=http://books.google.com/books?id=LqJUonES6m8C&q=benjamin+franklin+identify+christian+religion&dq=benjamin+franklin+identify+christian+religion&hl=en&ei=dMHMTYaOHILeOQHKpYjeBA&sa=X&oi=book_result&ct=result&resnum=2&ved=0CC4Q6AEwATge|title=The American Mercury, Volume 8|publisher=Garber Communications|quote=It is well known that in his youth Benjamin Franklin was a thorough-going Deist, but because he proposed that prayers be said in the Constitution Convention of 1787 many have contended that in later life he became a pious Christian.|date=October 19, 2009}}</ref><ref name="Faith">{{cite book|author=Ralph Frasca|url=http://books.google.com/books?id=CY2UVzcU5l0C&pg=PA40&dq=benjamin+franklin+christian+or+deist&hl=en&ei=wEXLTe7uEY-2twf-rIXyBw&sa=X&oi=book_result&ct=result&resnum=7&ved=0CEcQ6AEwBg#v=onepage&q&f=false|title=Benjamin Franklin's Printing Network: Disseminating Virtue in Early America|publisher=[[University of Missouri Press]]|quote=Despite being raised a Puritan of the Congregationalist stripe by his parents, who "brought me through my Childhood piously in the Dissenting Way," Franklin recalled, he abandoned that denomination, briefly embraced deism, and finally became a non-denominational Protestant Christian. |date=October 19, 2009}}</ref>
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/1
Railgun_Ennearch performance: 51200KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 761clocks
Railgun_Ennearch precise performance: 267842MB/s

Doing Search for Pattern(3867bytes) into String(5242880bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 52180092 within next line:
Although Franklin's parents had intended for him to have a career in the Church,<ref name="autobio" /> Franklin as a young man adopted the Enlightenment religious belief in deism, that God’s truths can be found entirely through nature and reason.<ref>Isaacson, 2003, p. 46</ref> "I soon became a thorough Deist."<ref>Franklin, Benjamin. [http://www.usgennet.org/usa/topic/preservation/bios/franklin/chpt4.htm ''Benjamin Franklin's Autobiography''.] Chapter IV. reprinted on USGenNet.org.</ref> As a young man he rejected Christian dogma in a 1725 pamphlet ''[[A Dissertation on Liberty and Necessity, Pleasure and Pain]]'',<ref>{{cite web|url=http://www.historycarper.com/resources/twobf1/m7.htm |title=A Dissertation on Liberty and Necessity, Pleasure and Pain |publisher=Historycarper.com |accessdate=September 21, 2009}}</ref> which he later saw as an embarrassment,<ref name="Isaacson45">{{cite book|url=http://books.google.com/?id=oIW915dDMBwC&pg=PA45&lpg=PA45&dq=%22A+Dissertation+on+Liberty+and+Necessity,+Pleasure+and+Pain%22+%22Benjamin+Franklin%22+embarrassment|title=Isaacson, 2003, p. 45 |publisher=Google Books |date=November 30, 2004 |accessdate=September 21, 2009|isbn=978-0-684-80761-4|author1=Isaacson, Walter}}</ref> while simultaneously asserting that God is "all wise, [[Omnibenevolence|all good]], [[Omnipotence|all powerful]]."<ref name="Isaacson45" /> He defended his rejection of religious dogma with these words: "I think opinions should be judged by their influences and effects; and if a man holds none that tend to make him less virtuous or more vicious, it may be concluded that he holds none that are dangerous, which I hope is the case with me." After the disillusioning experience of seeing the decay in his own moral standards, and those of two friends in London whom he had converted to Deism, Franklin turned back to a belief in the importance of organized religion, on the pragmatic grounds that without God and organized churches, man will not be

```
good.&lt;ref&gt;Isaacson, 2003, p 46, 486&lt;/ref&gt; Moreover, because of his proposal that [[Prayer in Christianity|prayers]] be said in the [[Constitutional Convention of 1787]], many have contended that in his later life, Franklin became a [[pious]] Christian.&lt;ref name=&quot;Pious&quot;&gt;{{cite book|author=Henry Louis Mencken, George Jean Nathan|url=http://books.google.com/books?id=LqJUonES6m8C&amp;q=benjamin+franklin+identify+christian+religion&amp;dq=benjamin+franklin+identify+christian+religion&amp;hl=en&amp;ei=dMHMTYaOHILe0QHKpYjeBA&amp;sa=X&amp;oi=book_result&amp;ct=result&amp;resnum=2&amp;ved=0CC4Q6AEwATge|title=The American Mercury, Volume 8|publisher=Garber Communications|quote=It is well known that in his youth Benjamin Franklin was a thorough-going Deist, but because he proposed that prayers be said in the Constitution Convention of 1787 many have contended that in later life he became a pious Christian.|date=October 19, 2009}}&lt;/ref&gt;&lt;ref name=&quot;Faith&quot;&gt;{{cite book|author=Ralph Frasca|url=http://books.google.com/books?id=CY2UVzcU510C&amp;pg=PA40&amp;dq=benjamin+franklin+christian+or+deist&amp;hl=en&amp;ei=wEXLTe7uEY-2twf-rIXyBw&amp;sa=X&amp;oi=book_result&amp;ct=result&amp;resnum=7&amp;ved=0CEcQ6AEwBg#v=onepage&amp;q&amp;f=false|title=Benjamin Franklin's Printing Network: Disseminating Virtue in Early America|publisher=[[University of Missouri Press]]|quote=Despite being raised a Puritan of the Congregationalist stripe by his parents, who &quot;brought me through my Childhood piously in the Dissenting Way,&quot; Franklin recalled, he abandoned that denomination, briefly embraced deism, and finally became a non-denominational Protestant Christian. |date=October 19, 2009}}&lt;/ref&gt;
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/1
Railgun_DecumanusBITified performance: 51200KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 729clocks
Railgun_DecumanusBITified precise performance: 279599MB/s

Doing Search for Pattern(3867bytes) into String(52428800bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 52180092
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/5
Boyer_Moore-Horspool performance: 10240KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 17206clocks
Boyer_Moore-Horspool precise performance: 11846MB/s


32bit_Intelv12:
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(15bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/170
Railgun_Swampshine performance: 1278KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 43355clocks
Railgun_Swampshine precise performance: 1253MB/s

Doing Search for Pattern(15bytes) into String(222610477bytes) as-one-line ...
```

```
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/176
Railgun_Ennearch performance: 1235KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 44811clocks
Railgun_Ennearch precise performance: 1212MB/s

Doing Search for Pattern(15bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/188
Railgun_DecumanusBITified performance: 1156KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 48102clocks
Railgun_DecumanusBITified precise performance: 1129MB/s

Doing Search for Pattern(15bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/262
Boyer_Moore-Horspool performance: 829KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 66863clocks
Boyer_Moore-Horspool precise performance: 812MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(18bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610322 within next line:
TTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/140
Railgun_Swampshine performance: 1552KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 35783clocks
Railgun_Swampshine precise performance: 1518MB/s
```

```
Doing Search for Pattern(18bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610322 within next line:
TTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/145
Railgun_Ennearch performance: 1499KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 36968clocks
Railgun_Ennearch precise performance: 1470MB/s

Doing Search for Pattern(18bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610322 within next line:
TTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/155
Railgun_DecumanusBITified performance: 1402KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 39565clocks
Railgun_DecumanusBITified precise performance: 1373MB/s

Doing Search for Pattern(18bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610322
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/246
Boyer_Moore-Horspool performance: 883KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 62793clocks
Boyer_Moore-Horspool precise performance: 865MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(20bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/31
Railgun_Swampshine performance: 7012KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
```

```
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 7707clocks
Railgun_Swampshine precise performance: 7051MB/s

Doing Search for Pattern(20bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/30
Railgun_Ennearch performance: 7246KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 7667clocks
Railgun_Ennearch precise performance: 7088MB/s

Doing Search for Pattern(20bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/35
Railgun_DecumanusBITified performance: 6211KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 8800clocks
Railgun_DecumanusBITified precise performance: 6175MB/s

Doing Search for Pattern(20bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/392
Boyer_Moore-Horspool performance: 554KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 100230clocks
Boyer_Moore-Horspool precise performance: 542MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(30bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/29
Railgun_Swampshine performance: 7496KB/clock
```

```
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 7247clocks
Railgun_Swampshine precise performance: 7499MB/s

Doing Search for Pattern(30bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/28
Railgun_Ennearch performance: 7764KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 6974clocks
Railgun_Ennearch precise performance: 7792MB/s

Doing Search for Pattern(30bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/31
Railgun_DecumanusBITified performance: 7012KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 7744clocks
Railgun_DecumanusBITified precise performance: 7018MB/s

Doing Search for Pattern(30bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/185
Boyer_Moore-Horspool performance: 1175KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 47175clocks
Boyer_Moore-Horspool precise performance: 1152MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(40bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
```

```
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/29
Railgun_Swampshine performance: 7496KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 7246clocks
Railgun_Swampshine precise performance: 7500MB/s

Doing Search for Pattern(40bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/25
Railgun_Ennearch performance: 8695KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 6145clocks
Railgun_Ennearch precise performance: 8844MB/s

Doing Search for Pattern(40bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/27
Railgun_DecumanusBITified performance: 8051KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 6681clocks
Railgun_DecumanusBITified precise performance: 8134MB/s

Doing Search for Pattern(40bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/263
Boyer_Moore-Horspool performance: 826KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 67161clocks
Boyer_Moore-Horspool precise performance: 809MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(50bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
```

```
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/29
Railgun_Swampshine performance: 7496KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 7248clocks
Railgun_Swampshine precise performance: 7498MB/s

Doing Search for Pattern(50bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/24
Railgun_Ennearch performance: 9058KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 5962clocks
Railgun_Ennearch precise performance: 9115MB/s

Doing Search for Pattern(50bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/26
Railgun_DecumanusBITified performance: 8361KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 6443clocks
Railgun_DecumanusBITified precise performance: 8435MB/s

Doing Search for Pattern(50bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/366
Boyer_Moore-Horspool performance: 593KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 93506clocks
Boyer_Moore-Horspool precise performance: 581MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT

Input Pattern(up to 19+5000 chars):
```

```
Doing Search for Pattern(60bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/29
Railgun_Swampshine performance: 7496KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TOtalRoughSearchTime (lesser-the-better): 7248clocks
Railgun_Swampshine precise performance: 7498MB/s

Doing Search for Pattern(60bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/23
Railgun_Ennearch performance: 9451KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 5796clocks
Railgun_Ennearch precise performance: 9376MB/s

Doing Search for Pattern(60bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/25
Railgun_DecumanusBITified performance: 8695KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 6225clocks
Railgun_DecumanusBITified precise performance: 8730MB/s

Doing Search for Pattern(60bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/322
Boyer_Moore-Horspool performance: 675KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 82363clocks
Boyer_Moore-Horspool precise performance: 659MB/s
strstr_SHORT-SHOWDOWN, revision Doublet_8Triplet_7Trident_7Hasherezade_vs_7Gulliver2_vs_7Elsiane_vs_7sun_vs_7_vs_7sunhorse_vs_7deuce_vs_BMF, written by Kaze.
Full credits to: R.S. Boyer, J.S. Moore, R.N. Horspool, D.M. Sunday.
Usage: strstr_SHORT-SHOWDOWN.exe [anystring] [anystring]
Example1(keyboard test): strstr_SHORT-SHOWDOWN.exe
Example2(DUMBO 8x2 test): strstr_SHORT-SHOWDOWN.exe go
Example3(6x2 and 52 tests): strstr_SHORT-SHOWDOWN.exe go go

Last 6 lines of 'OSHO.TXT' file 197 MB (206,908,949 bytes):
          If you have read through the preceding chapters you should have a pretty good idea on how to make
          use of the Osho Books on CD-ROM. What use you will put it to is up to you. It is the largest ever
          electronic repository of understanding and knowledge on meditation and its techniques. It is also
          much more, a complete, world view of the New Man and a new way of life. The purpose of this
          CD-ROM is to provide access to Osho's words, ideas and vision, and to make them available to as
          many people as possible.

Last 3 lines of 'hs_alt_HuRef_chr1.fa' file 212 MB (222,610,477 bytes):
AGATTTTAAAGATTTTCTTTTTTTTTGACATAGAATCTTATGGAGGCTGAGAAATAATTTTTTTTCTATT
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
GGGGTGCCTGAAGTT
```

```
Input Pattern(up to 19+5000 chars):
Doing Search for Pattern(70bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Swampshine_hits/Railgun_Swampshine_clocks: 0/26
Railgun_Swampshine performance: 8361KB/clock
Railgun_Swampshine Iterations performance (lesser-the-better): 0
Railgun_Swampshine HashSectionExecutions performance (lesser-the-better): 0
Railgun_Swampshine TotalRoughSearchTime (lesser-the-better): 6604clocks
Railgun_Swampshine precise performance: 8229MB/s

Doing Search for Pattern(70bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_Ennearch_hits/Railgun_Ennearch_clocks: 0/24
Railgun_Ennearch performance: 9058KB/clock
Railgun_Ennearch Iterations performance (lesser-the-better): 0
Railgun_Ennearch HashSectionExecutions performance (lesser-the-better): 0
Railgun_Ennearch TotalRoughSearchTime (lesser-the-better): 5929clocks
Railgun_Ennearch precise performance: 9166MB/s

Doing Search for Pattern(70bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position 222610390 within next line:
TTATTCTTCAGCCCCAGGTGTTTGCTTTTGCAGATTCTTGAGCACATTGAGAGCCTCCAAGGCATGGAGT
Railgun_DecumanusBITified_hits/Railgun_DecumanusBITified_clocks: 0/25
Railgun_DecumanusBITified performance: 8695KB/clock
Railgun_DecumanusBITified Iterations performance (lesser-the-better): 0
Railgun_DecumanusBITified HashSectionExecutions performance (lesser-the-better): 0
Railgun_DecumanusBITified TotalRoughSearchTime (lesser-the-better): 6261clocks
Railgun_DecumanusBITified precise performance: 8680MB/s

Doing Search for Pattern(70bytes) into String(222610477bytes) as-one-line ...
WARNING! The next line works with BMH only for pattern 4[+] long, otherwise (for 2 and 3) other searcher takes over!
The first instance occurred at position: 222610390
Boyer_Moore-Horspool_hits/Boyer_Moore-Horspool_clocks: 0/292
Boyer_Moore-Horspool performance: 744KB/clock
Boyer_Moore-Horspool Iterations performance (lesser-the-better): 0
Boyer_Moore-Horspool HashSectionExecutions performance (lesser-the-better): 0
Boyer_Moore-Horspool TotalRoughSearchTime (lesser-the-better): 74637clocks
Boyer_Moore-Horspool precise performance: 728MB/s
```