1 Boyer-Moore Algorithm

1. good-suffix shift (matching shift)

2. bad-character shift (occurence shift)

3. C code

```
void preBmBc(char *x, int m, int bmBc[]) {   /* Bad-character shift 表 */
   int i;

   for (i = 0; i < ASIZE; ++i)               /* すべての文字に m を設定 */
      bmBc[i] = m;
   for (i = 0; i < m - 1; ++i)               /* パターンに出現する文字の処理 */
      bmBc[x[i]] = m - i - 1;
}
                                     /* suffix[i]= k s.t. x[i-k+1, ..., i]=x[m-k, ..., m-1] */
void suffixes(char *x, int m, int *suff) {
   int f, g, i;

   suff[m - 1] = m;
   g = m - 1;
   for (i = m - 2; i >= 0; --i) {
      if (i > g && suff[i + m - 1 - f] < i - g)
         suff[i] = suff[i + m - 1 - f];
      else {
         if (i < g)
            g = i;
         f = i;
         while (g >= 0 && x[g] == x[g + m - 1 - f])
            --g;
         suff[i] = f - g;
      }
   }
}

void preBmGs(char *x, int m, int bmGs[]) {
   int i, j, suff[XSIZE];

   suffixes(x, m, suff);

   for (i = 0; i < m; ++i)
      bmGs[i] = m;
   j = 0;
   for (i = m - 1; i >= -1; --i)
      if (i == -1 || suff[i] == i + 1)
         for (; j < m - 1 - i; ++j)
            if (bmGs[j] == m)
               bmGs[j] = m - 1 - i;
   for (i = 0; i <= m - 2; ++i)
      bmGs[m - 1 - suff[i]] = m - 1 - i;
}

void BM(char *x, int m, char *y, int n) {
   int i, j, bmGs[XSIZE], bmBc[ASIZE];

   /* Preprocessing */
   preBmGs(x, m, bmGs);
   preBmBc(x, m, bmBc);

   /* Searching */
   j = 0;
   while (j <= n - m) {
      for (i = m - 1; i >= 0 && x[i] == y[i + j]; --i);
      if (i < 0) {
         OUTPUT(j);
         j += bmGs[0];
      }
      else
         j += MAX(bmGs[i], bmBc[y[i + j]] - m + 1 + i);
   }
}
```

4. 作成される表

c	A	C	G	T
$bmBc[c]$	1	6	2	8

i	0	1	2	3	4	5	6	7
$x[i]$	G	C	A	G	A	G	A	G
$suff[i]$	1	0	0	2	0	4	0	8
$bmGs[i]$	7	7	7	2	7	4	7	1

5. 実行例 GCATCGCAGAGAGTATACAGTACG で実行される文字比較の回数は？

2 Knuth-Morris-Pratt Algorithm

1. tagged border

2. C code

```
void preKmp(char *x, int m, int kmpNext[]) {
   int i, j;

   i = 0;
   j = kmpNext[0] = -1;
   while (i < m) {
      while (j > -1 && x[i] != x[j])
         j = kmpNext[j];
      i++;
      j++;
      if (x[i] == x[j])
         kmpNext[i] = kmpNext[j];
      else
         kmpNext[i] = j;
   }
}


void KMP(char *x, int m, char *y, int n) {
   int i, j, kmpNext[XSIZE];

   /* Preprocessing */
   preKmp(x, m, kmpNext);

   /* Searching */
   i = j = 0;
   while (j < n) {
      while (i > -1 && x[i] != y[j])
         i = kmpNext[i];
      i++;
      j++;
      if (i >= m) {
         OUTPUT(j - i);
         i = kmpNext[i];
      }
   }
}
```

3. 作成される表

i	0	1	2	3	4	5	6	7	8
$x[i]$	G	C	A	G	A	G	A	G	
$kmpNext[i]$	-1	0	0	-1	1	-1	1	-1	1

4. 実行例 GCATCGCAGAGAGTATACAGTACG で実行される文字比較の回数は？