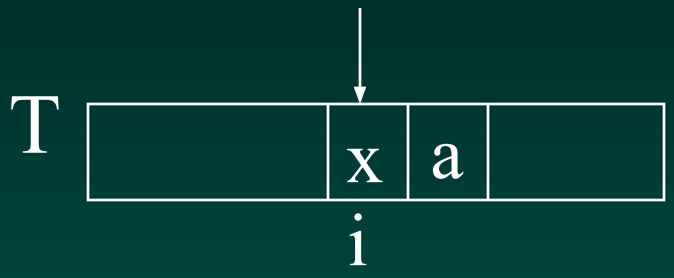
3. *The Boyer-Moore Algorithm*

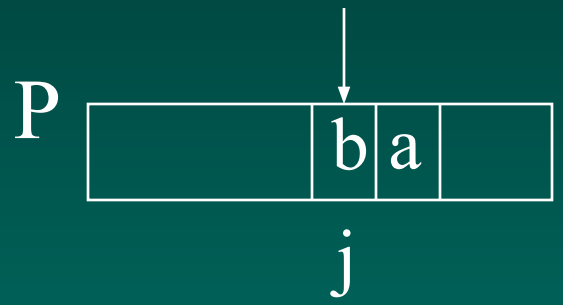
- The Boyer-Moore pattern matching algorithm is based on two Phases.

- 1. The *looking-glass* phase
  - find P in T by moving *backwards* through P, starting at its end

- 2. The *character-jump* phase
  - when a mismatch occurs at T[i] == x
  - the character in pattern P[j] is not the same as T[i]

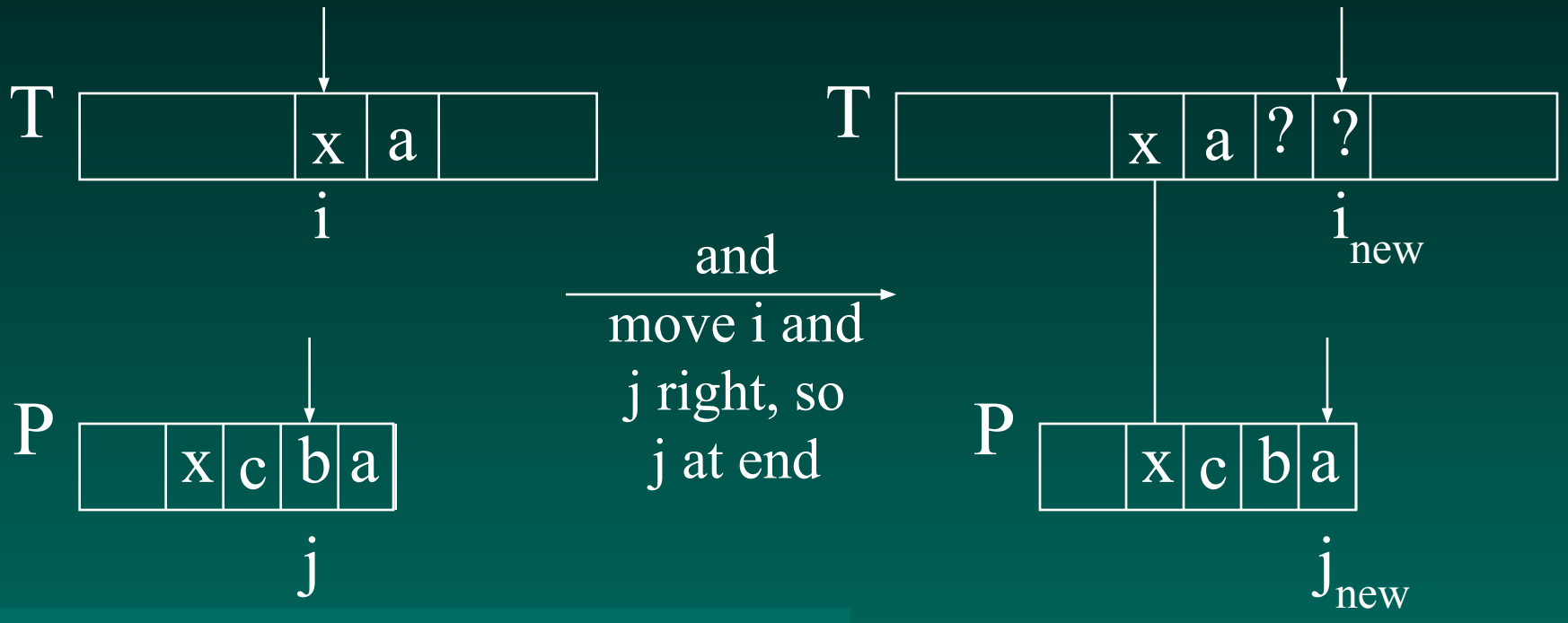


- There are 3 possible cases, tried in order.



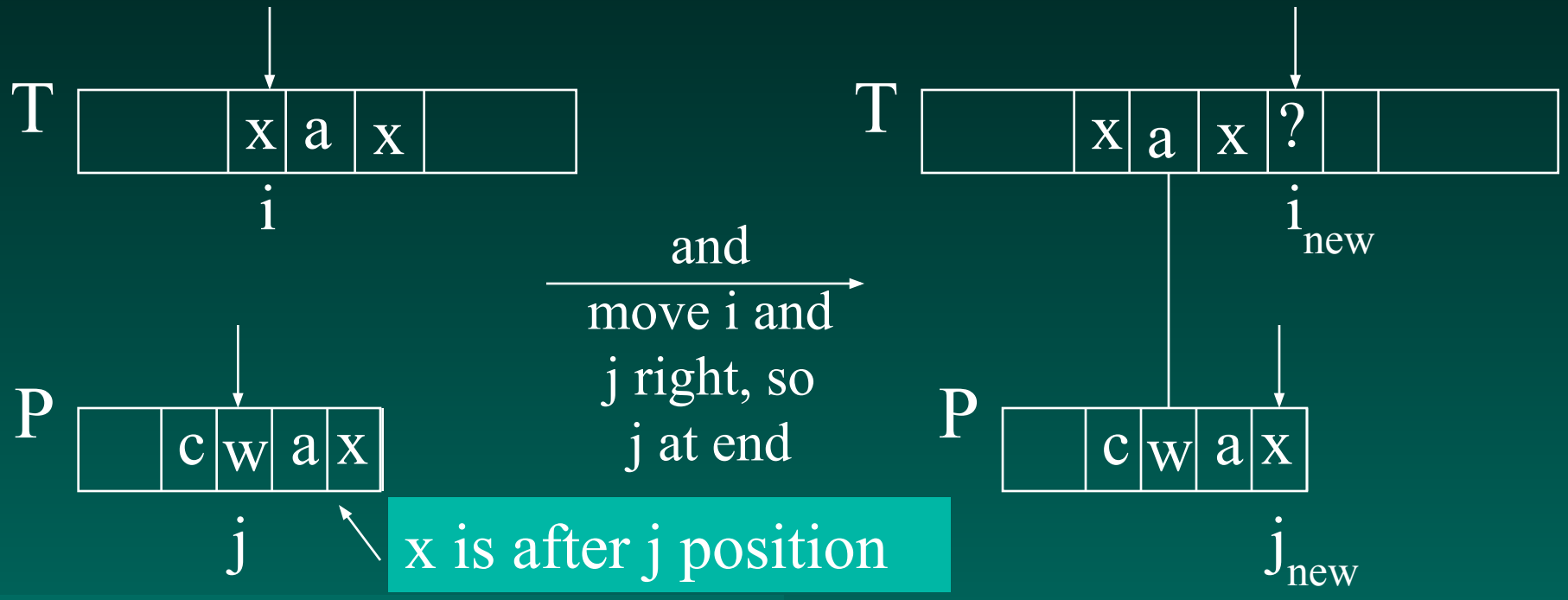
Case 1

- If P contains x somewhere, then try to *shift P* right to align the last occurrence of x in P with T[i].



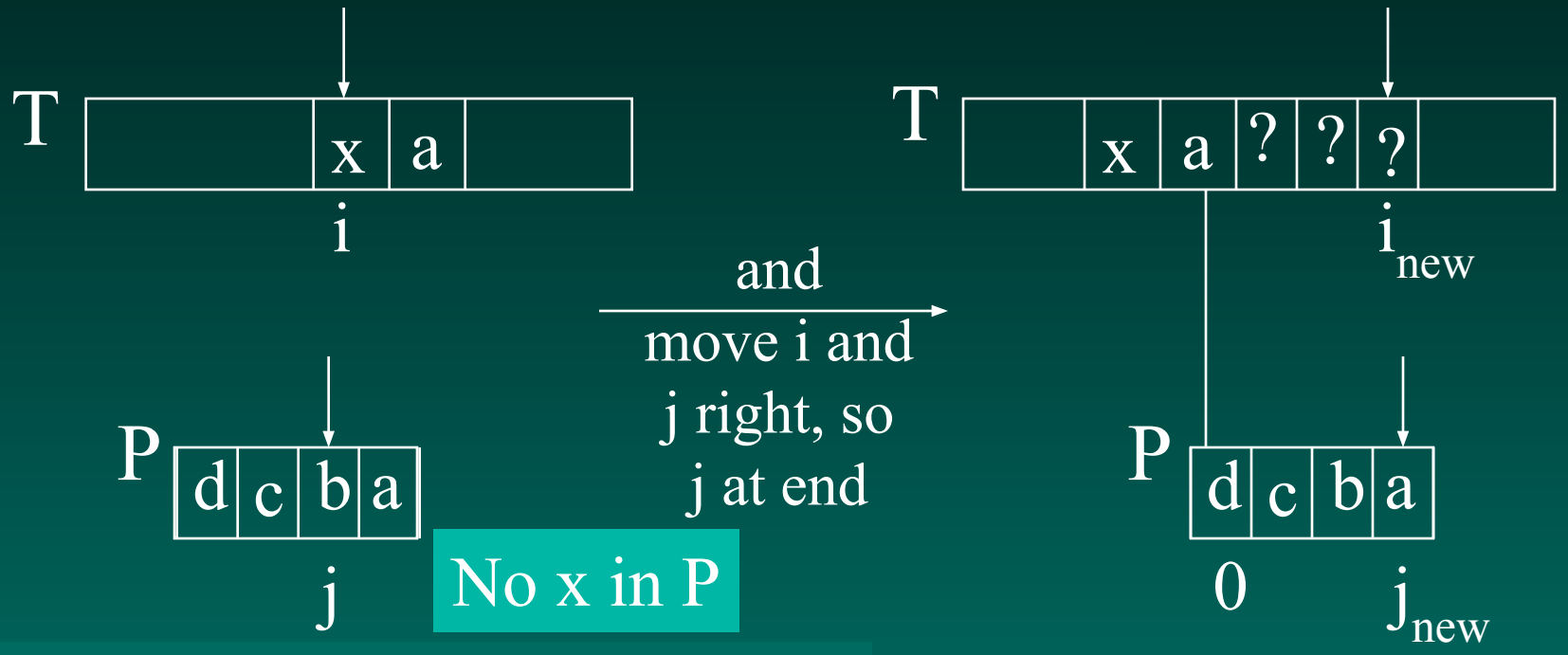
Case 2

- If P contains x somewhere, but a shift right to the last occurrence is *not* possible, then *shift P* right by 1 character to T[i+1].

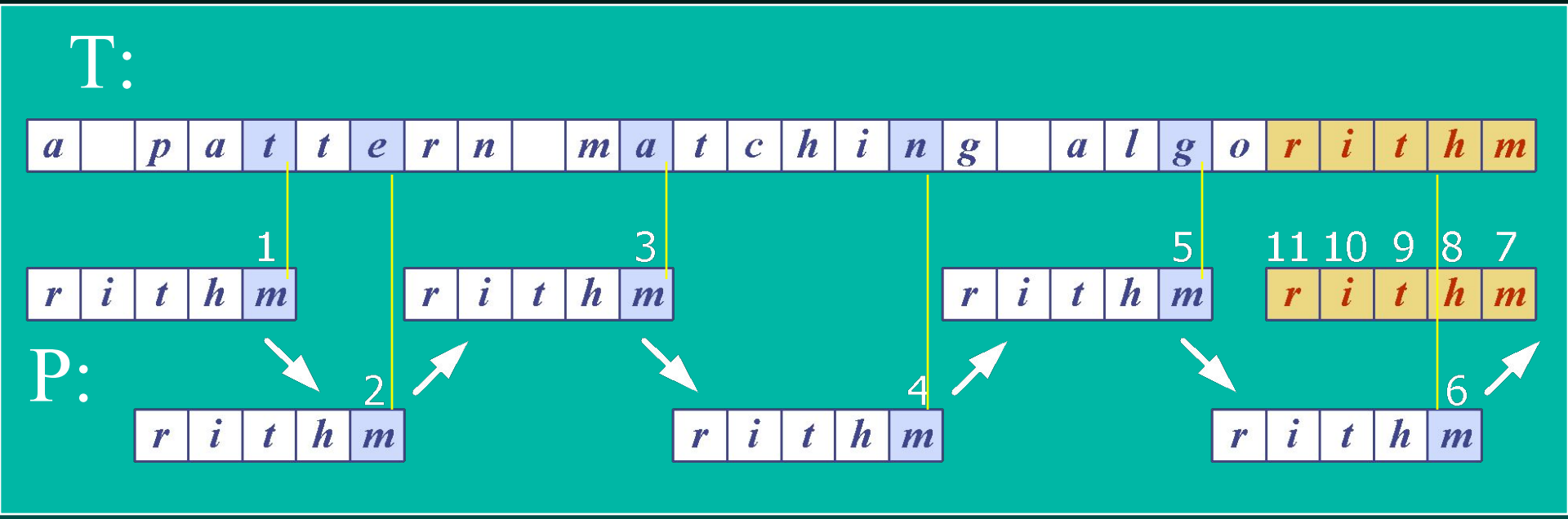


Case 3

- If cases 1 and 2 do not apply, then *shift* P to align P[0] with T[i+1].



3. The Boyer-Moore Algorithm



Last Occurrence Function

- Boyer-Moore's algorithm preprocesses the pattern P and the alphabet A to build a last occurrence function L()
  - L() maps all the letters in A to integers

- L(x) is defined as: // x is a letter in A
  - the largest index i such that P[i] == x, or
  - -1 if no such index exists

L() Example

P

a	b	a	c	a	b
0	1	2	3	4	5

- A = {a, b, c, d}
- P: "abacab"

x	***a***	***b***	***c***	***d***
***L*(*x*)**	4	5	3	-1

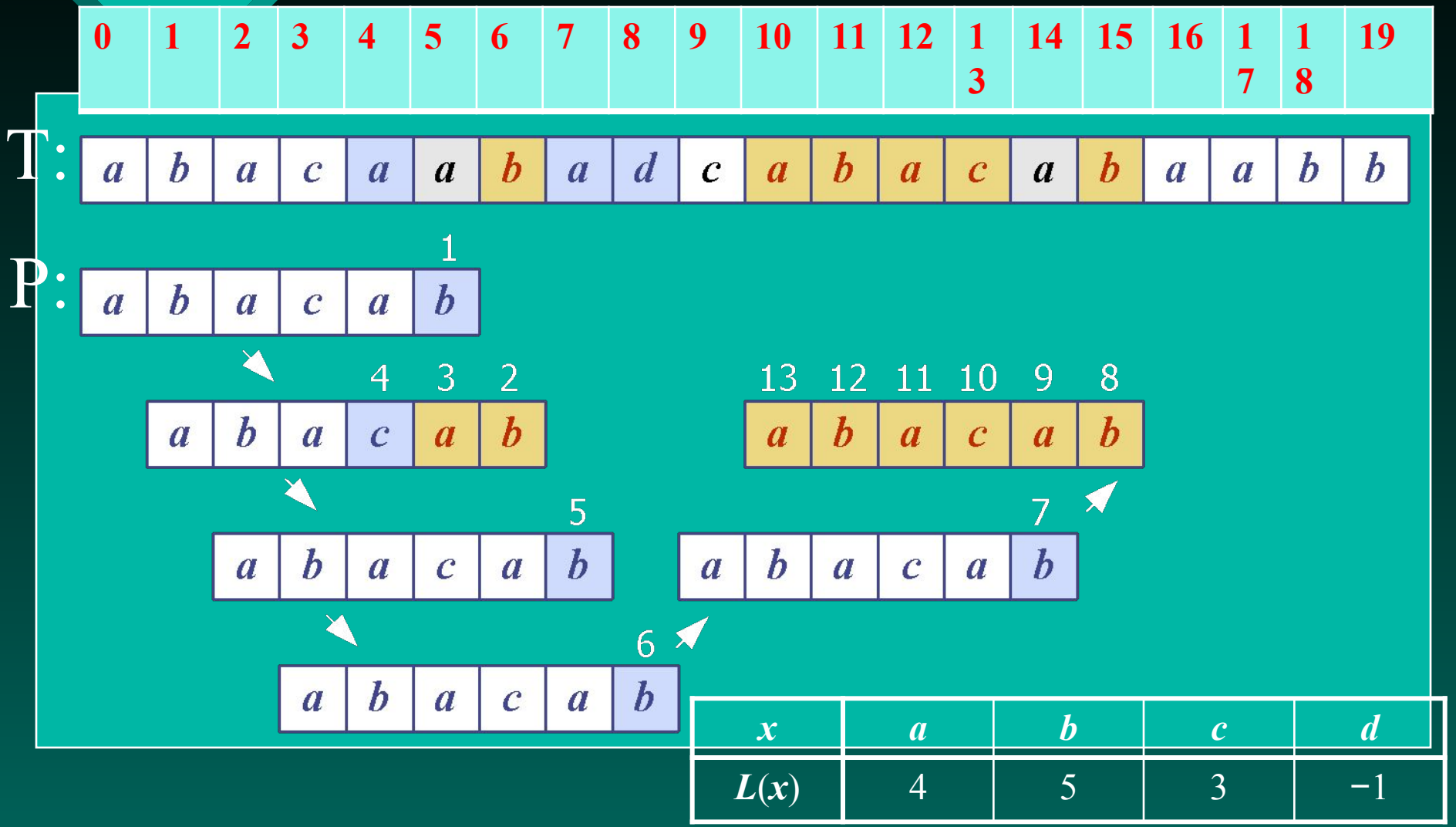
L() stores indexes into P[]

- In Boyer-Moore code, L() is calculated when the pattern P is read in.

- Usually L() is stored as an array
  - something like the table

3. The Boyer-Moore Algorithm

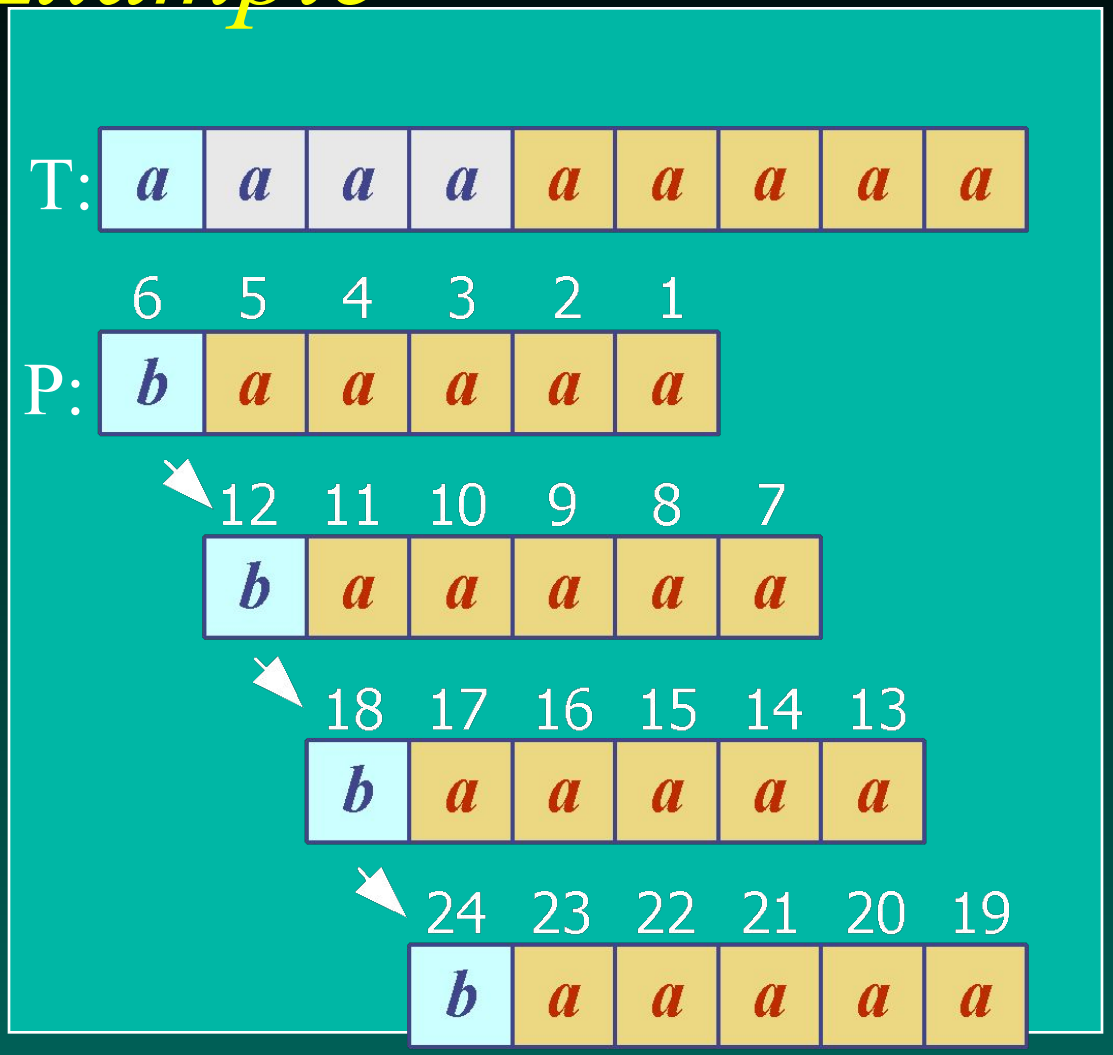
Example (2)



x	a	b	c	d
$L(x)$	4	5	3	−1

Worst Case Example

- T: "aaaaa…a"
- P: "baaaaa"



Analysis

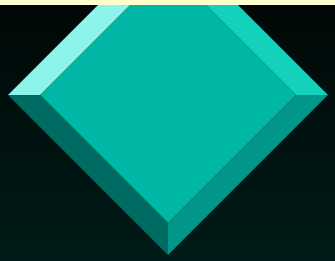
- Boyer-Moore is fast when the alphabet (A) is large, slow when the alphabet is small.
  - e.g. good for English text, poor for binary

- Boyer-Moore is *significantly faster than brute force* for searching English text.

Algorithm for searching the pattern P in text T using Boyer Moore

```
Algorithm Boyer – Moore (char T[], char P[],  int n, int m)
 {      call BuildLast(P,m,L);
          int  i=j = m-1;
         do {
             if (P[j] == T[i])
                   {    if (j == 0)   return i; // match
                     else { // looking-glass technique
                             i--;  j--;
                            }
                   }
               else { // character jump technique
                     int lo = L[T[i]];  //last occurence
                     i = i + m - min(j, 1+lo);
                     j = m - 1;
                   }
   } while (i <= n-1);
     return -1; // no match
 } // end of Bayer More
```

Algorithm for Building the Last Array Values



```
Algorithm BuildLast(char P[], int m,L)
{ // Return array storing index of last
  // occurrence of each char in pattern.
   int L[26]
   char C[26] = {a,b,c,d,e,f,g,h,i,,k,l,m,n,o,p,q,r,s,t,u,v,x,y,z}

   for(int i=0; i < 26; i++)

     L[i] = -1; // initialize array

   for (int i = 0; i< m; i++)

     {   A=p[i]

         for(int j=0;j<26;j++)

            if (A==C[j]) exit;

         L[j]= i;

     }

     return(L);

} // end of BuildLast()
```