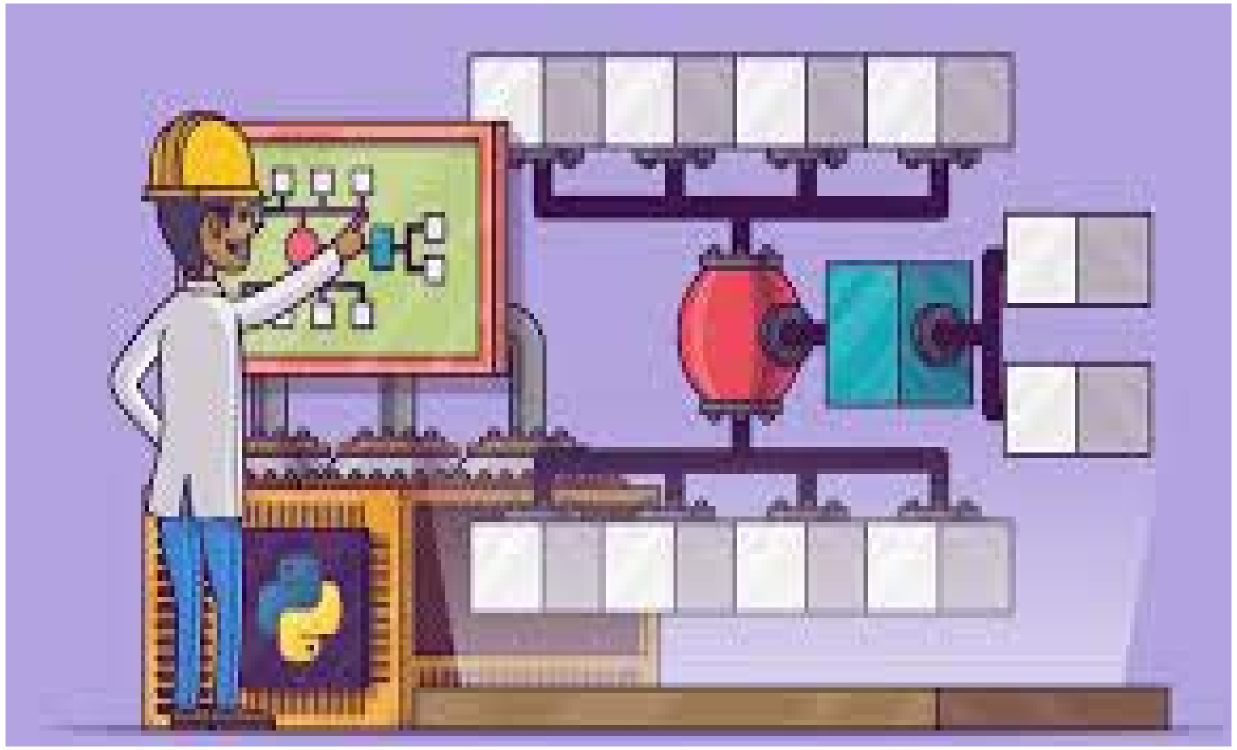
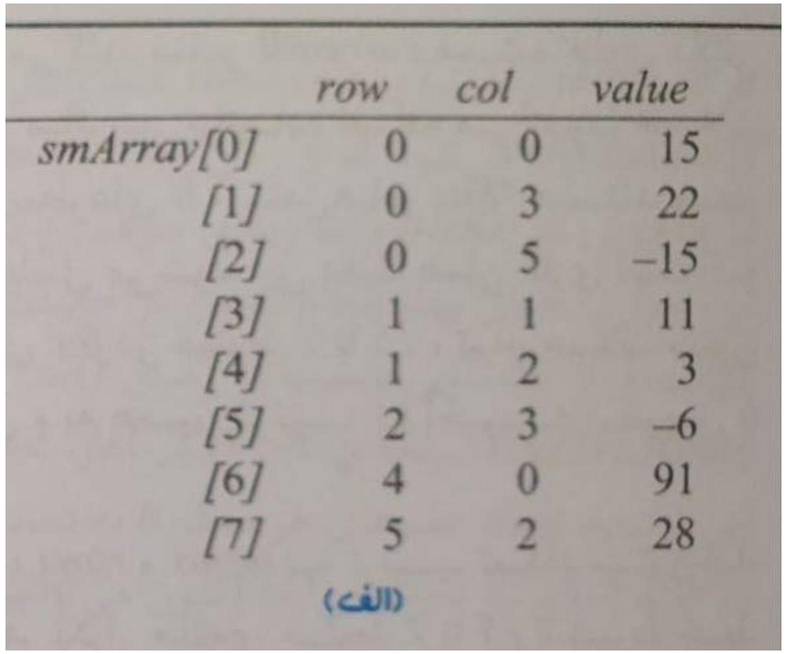
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

FastTranspose تابع



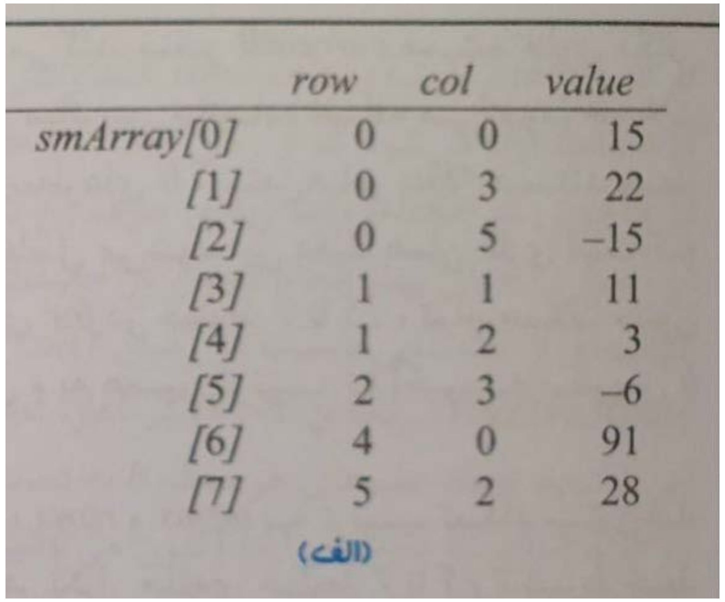
	row	col	value
smArray[0]	0	0	15
[1]	0	3	22
[2]	0	5	−15
[3]	1	1	11
[4]	1	2	3
[5]	2	3	−6
[6]	4	0	91
[7]	5	2	28

(الف)

$\Rightarrow$

	[0]	[1]	[2]	[3]	[4]	[5]
rowsize =	2	1	2	2	0	1
rowstart =	0	2	3	5	7	7

FastTranspose تابع

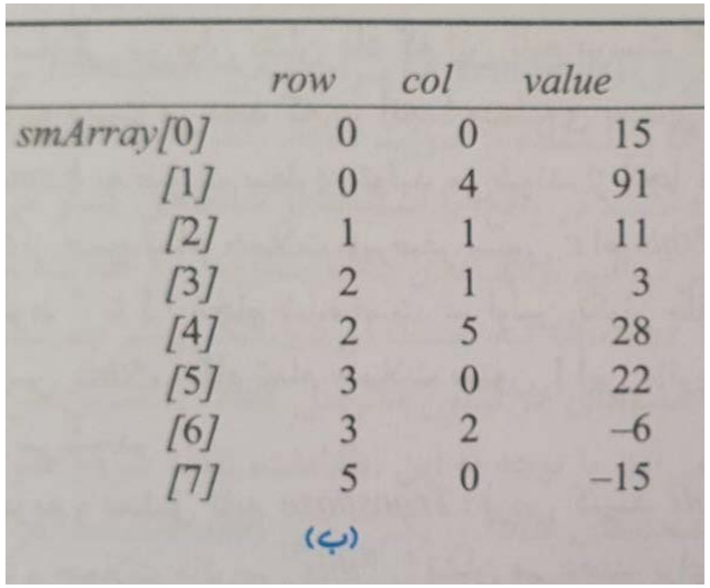


	row	col	value
smArray[0]	0	0	15
[1]	0	3	22
[2]	0	5	−15
[3]	1	1	11
[4]	1	2	3
[5]	2	3	−6
[6]	4	0	91
[7]	5	2	28

(الف)

```
              [0] [1] [2] [3] [4] [5]
rowsize   =  2   1   2   2   0  1
rowstart = 0    2   3   5   7  7
```

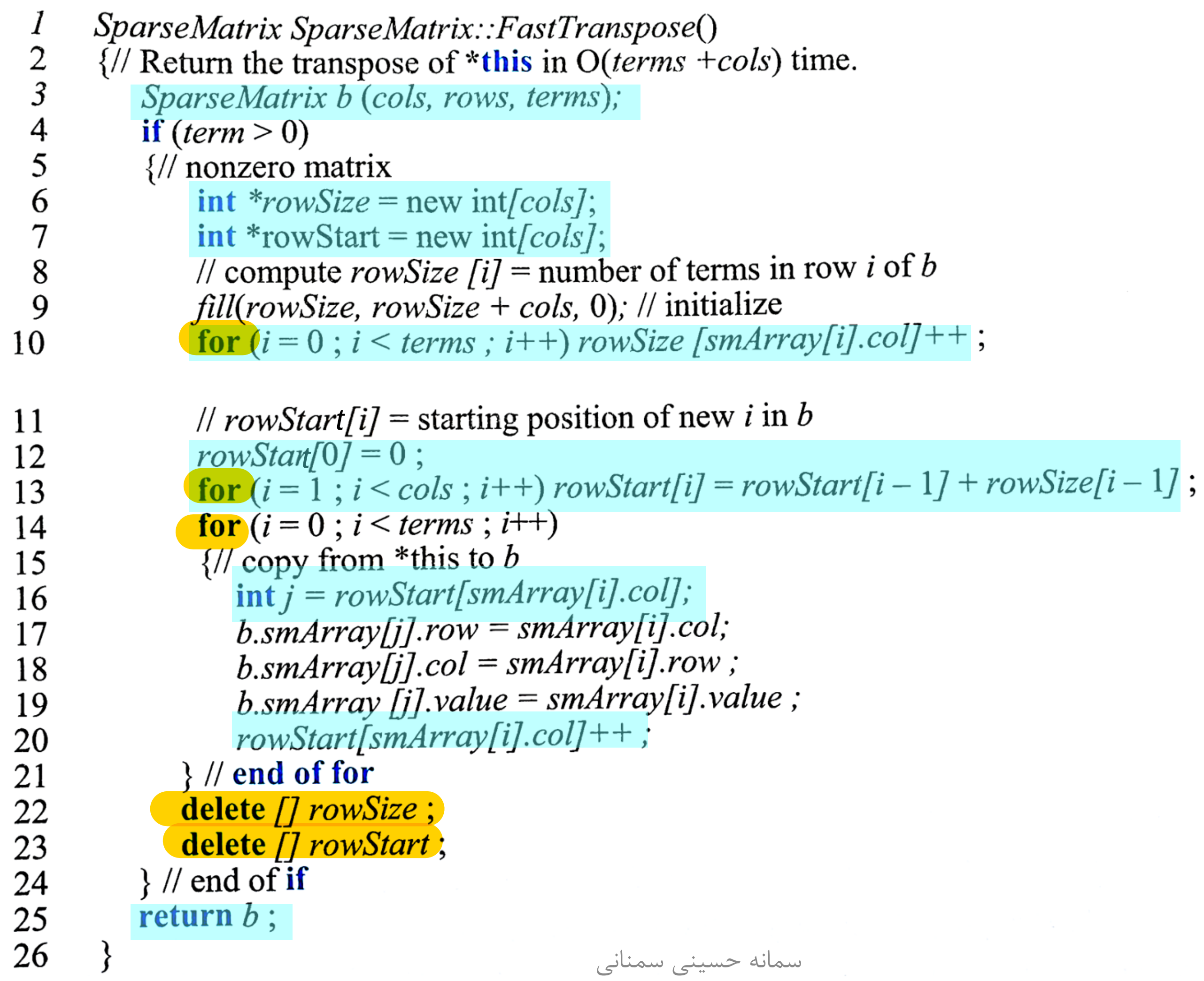
$\Longrightarrow$



	row	col	value
smArray[0]	0	0	15
[1]	0	4	91
[2]	1	1	11
[3]	2	1	3
[4]	2	5	28
[5]	3	0	22
[6]	3	2	−6
[7]	5	0	−15

(ب)

تابع FastTranspose



```
SparseMatrix SparseMatrix::FastTranspose()
{// Return the transpose of *this in O(terms +cols) time.
    SparseMatrix b (cols, rows, terms);
    if (term > 0)
    {// nonzero matrix
        int *rowSize = new int[cols];
        int *rowStart = new int[cols];
        // compute rowSize [i] = number of terms in row i of b
        fill(rowSize, rowSize + cols, 0); // initialize
        for (i = 0 ; i < terms ; i++) rowSize [smArray[i].col]++ ;

        // rowStart[i] = starting position of new i in b
        rowStart[0] = 0 ;
        for (i = 1 ; i < cols ; i++) rowStart[i] = rowStart[i – 1] + rowSize[i – 1] ;
        for (i = 0 ; i < terms ; i++)
         {// copy from *this to b
            int j = rowStart[smArray[i].col];
            b.smArray[j].row = smArray[i].col;
            b.smArray[j].col = smArray[i].row ;
            b.smArray [j].value = smArray[i].value ;
            rowStart[smArray[i].col]++ ;
        } // end of for
       delete [] rowSize ;
       delete [] rowStart ;
    } // end of if
    return b ;
}
```

تحلیل تابع FastTranspose

$$O(cols + terms)$$

- برای ماتریس غیر خلوت $terms = cols \times rows$
- زمان اجرا $O(cols \times rows) \xrightarrow{cols \ll cols \times rows} O(cols + terms)$
- پیچیدگی مکانی FastTranspose بیشتر است.$(rowSize\ , rowStart)$

ADT رشته

ADT رشته

- ذخیره سازی در حافظه:
  - در زبان c++، رشته ها به صورت آرایه های کاراکتری که به کاراکتر تهی، \0،ختم می شوند نگهداری می گردد.

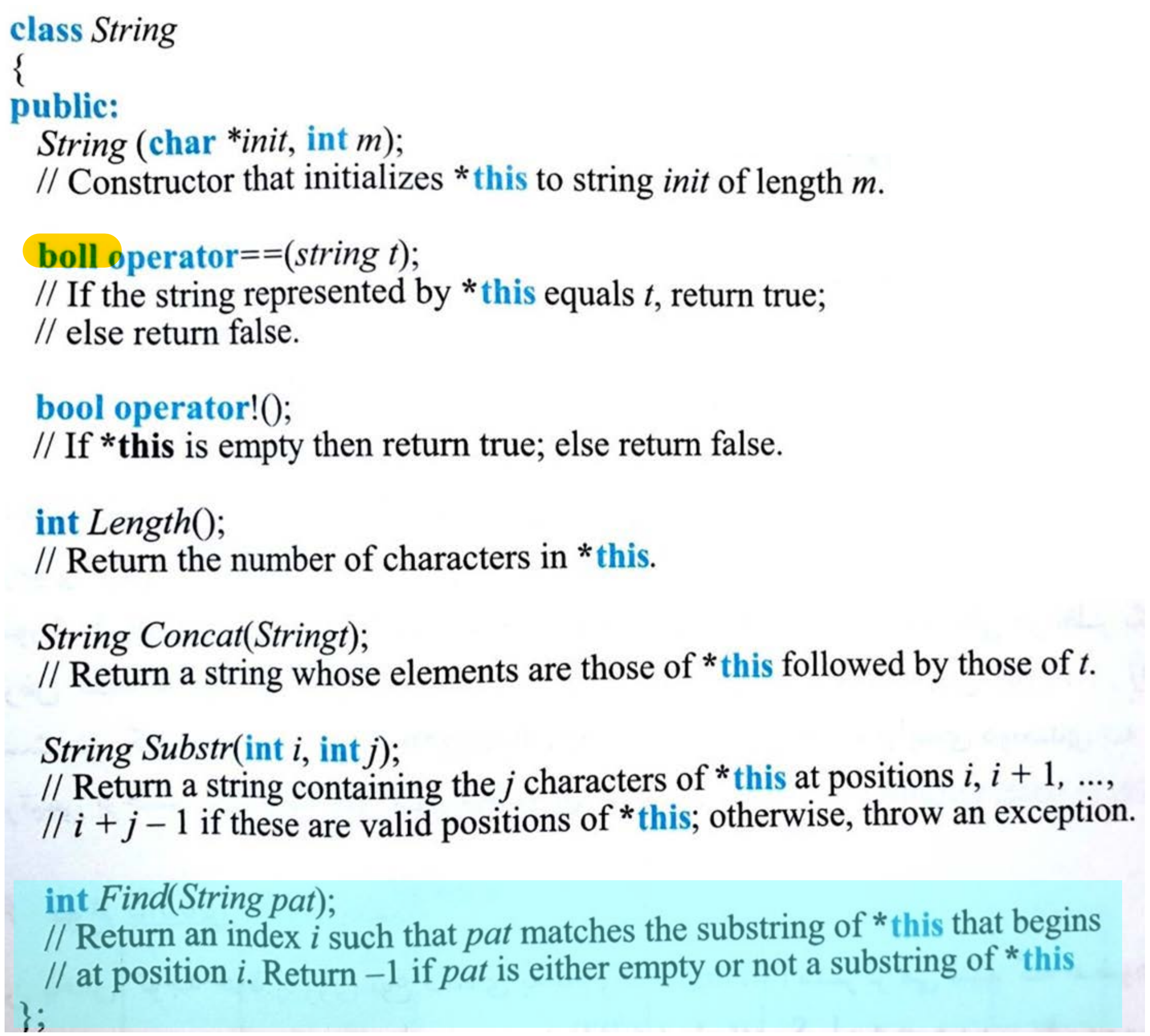
s[0]	s[1]	s[2]	s[3]
d	o	g	\0

```
char  s[] = “dog”;
```

ADT رشته

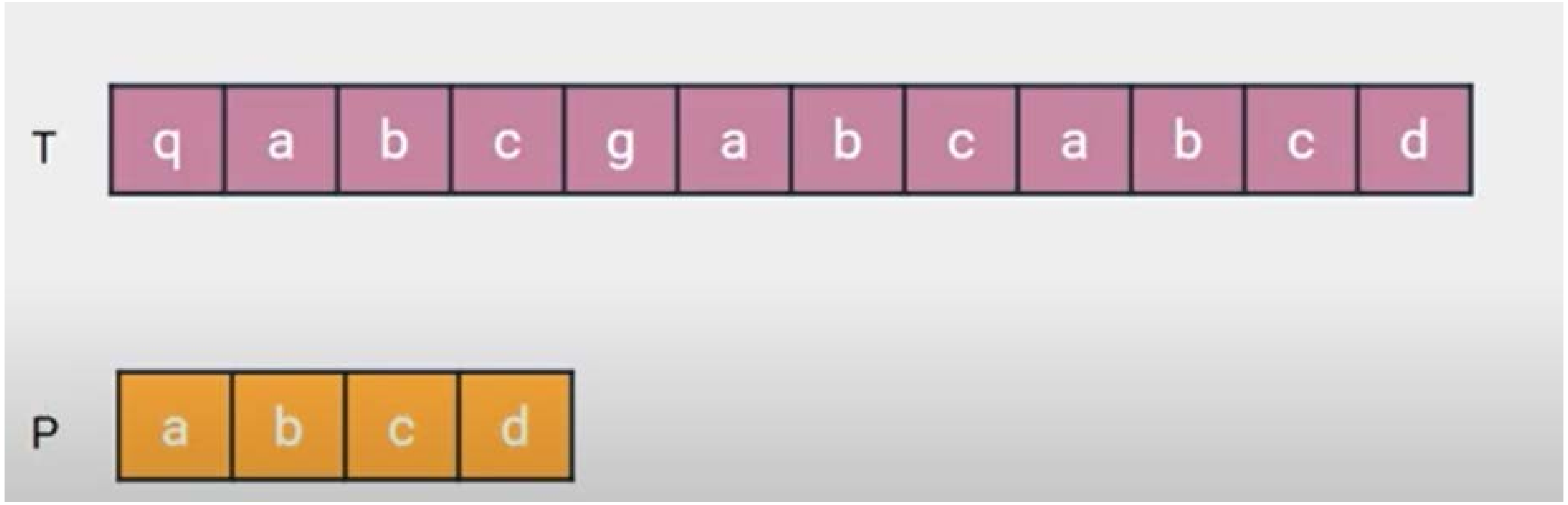
- اعمال:
  - ایجاد یک رشته تهی جدید
  - خواندن یا نوشتن یک رشته
  - ضمیمه کردن دو رشته به یکدیگر (concatenation)
  - کپی کردن یک رشته
  - مقایسه رشته ها
  - درج کردن یک زیر رشته به داخل رشته
  - برداشتن یک زیر رشته از یک رشته مشخص
  - پیدا کردن یک الگو (pattern) یا عبارت در یک رشته

ADT رشته



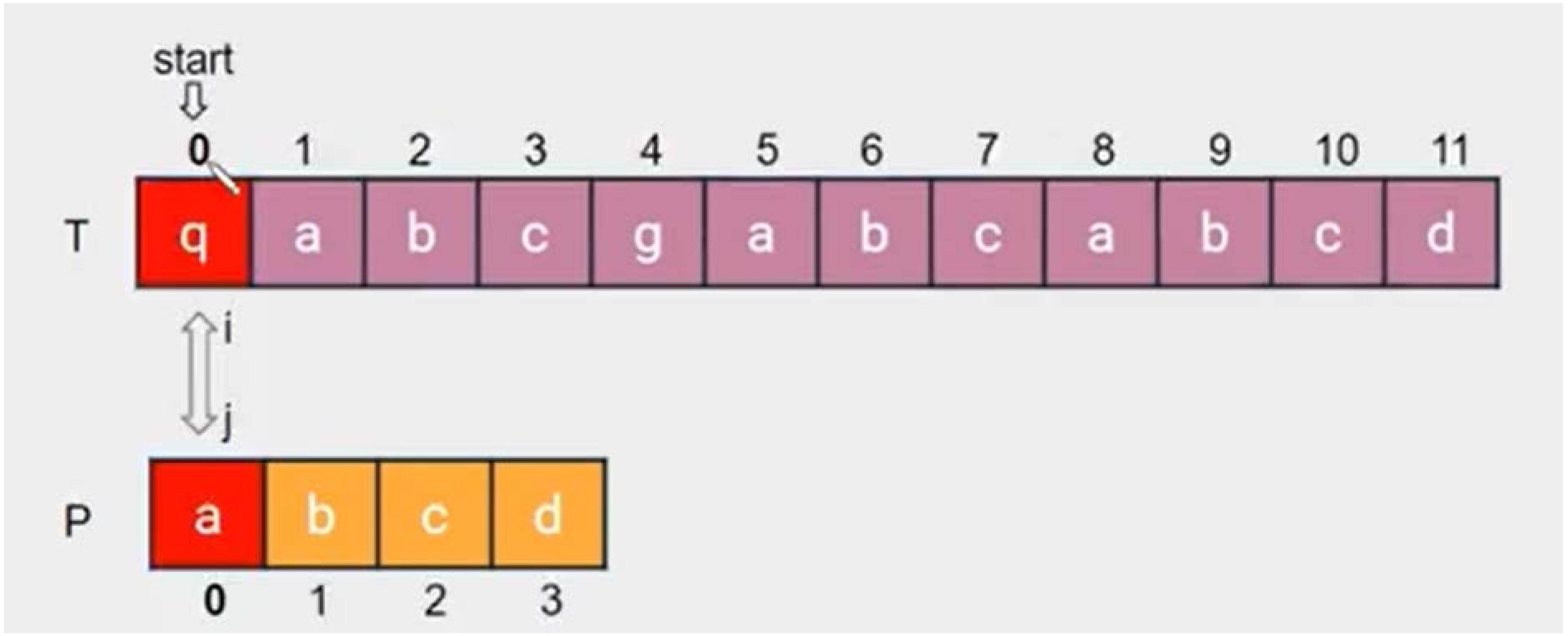
```
class String
{
public:
  String (char *init, int m);
  // Constructor that initializes *this to string init of length m.

  boll operator==(string t);
  // If the string represented by *this equals t, return true;
  // else return false.

  bool operator!();
  // If *this is empty then return true; else return false.

  int Length();
  // Return the number of characters in *this.

  String Concat(Stringt);
  // Return a string whose elements are those of *this followed by those of t.

  String Substr(int i, int j);
  // Return a string containing the j characters of *this at positions i, i + 1, ...,
  // i + j – 1 if these are valid positions of *this; otherwise, throw an exception.

  int Find(String pat);
  // Return an index i such that pat matches the substring of *this that begins
  // at position i. Return –1 if pat is either empty or not a substring of *this
};
```

پیدا کردن یک الگو در یک رشته- الگوریتم ساده

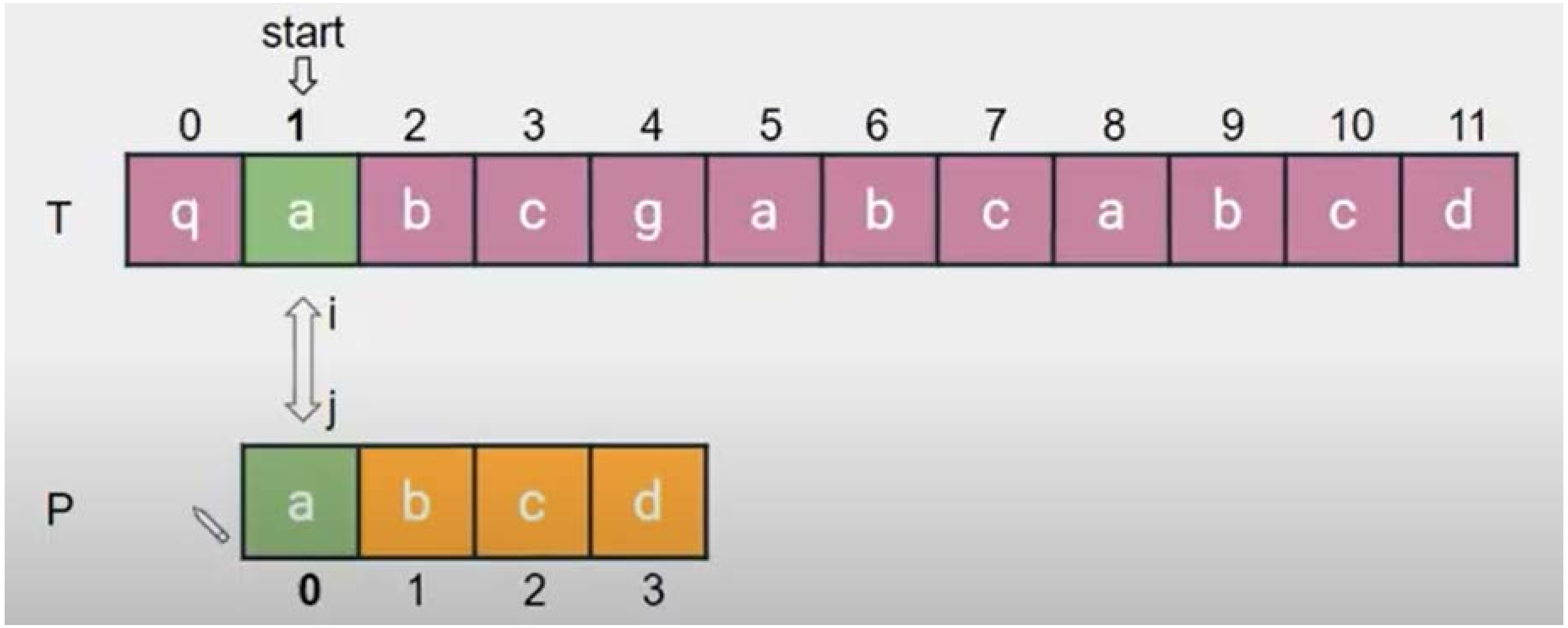


https://www.youtube.com/watch?reload=9&v=4jY57Ehc14Y

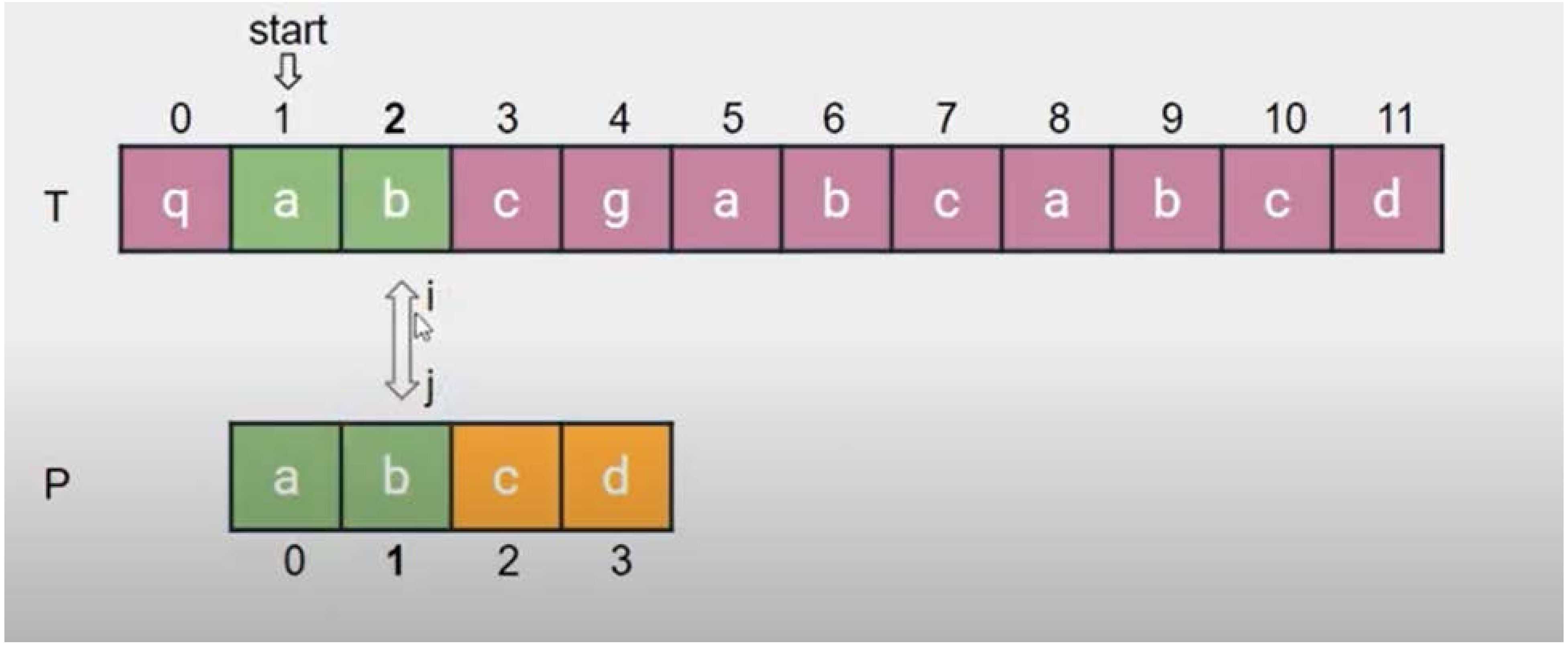
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



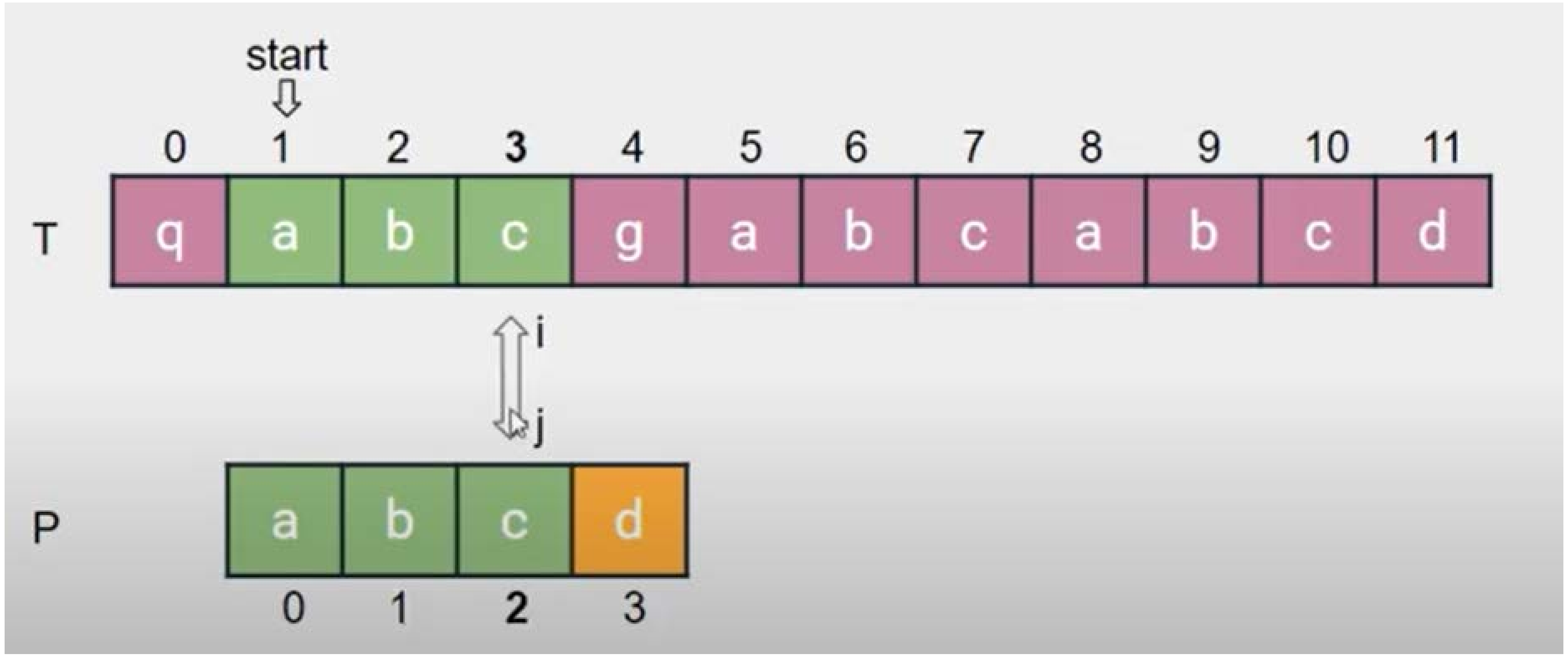
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



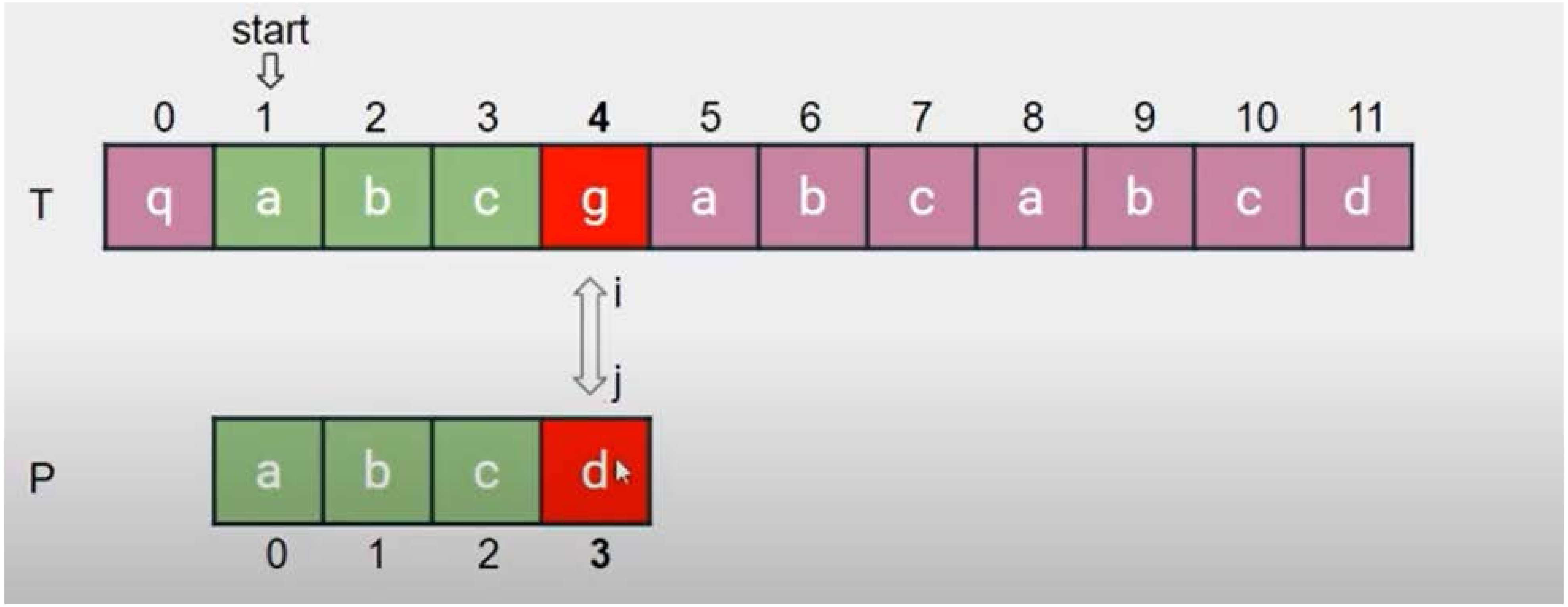
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



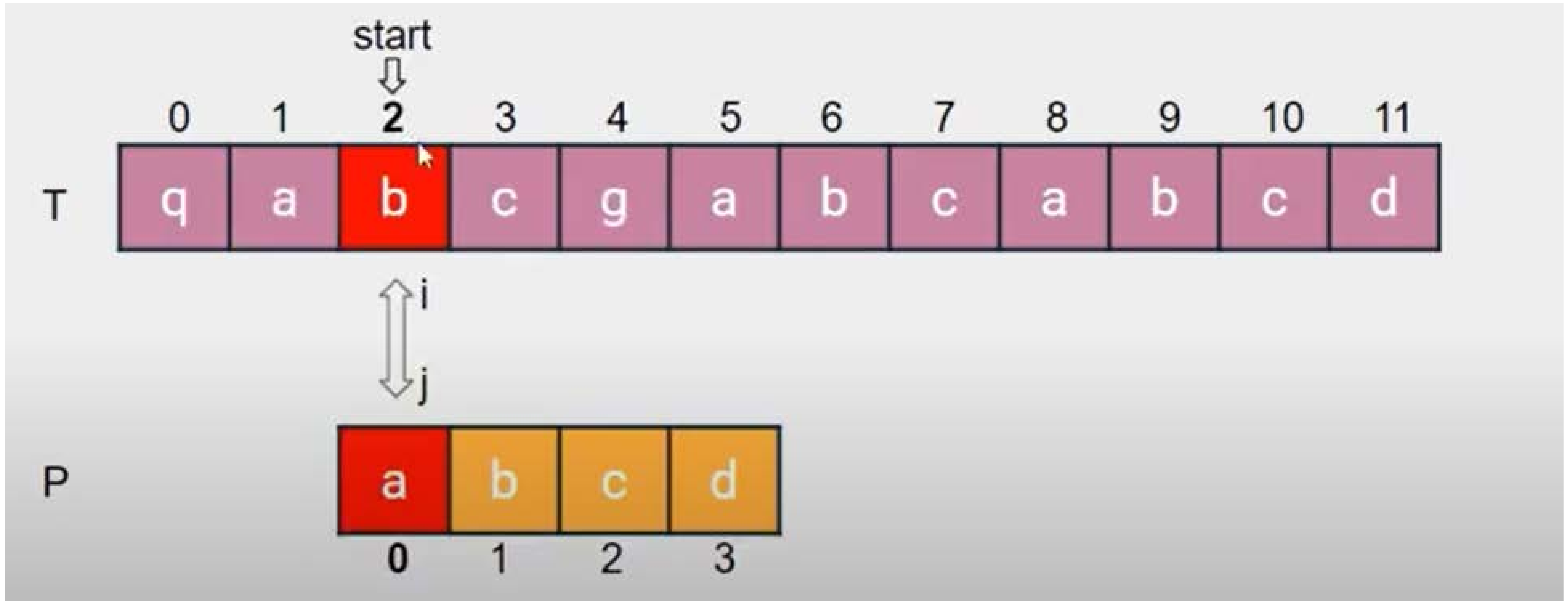
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



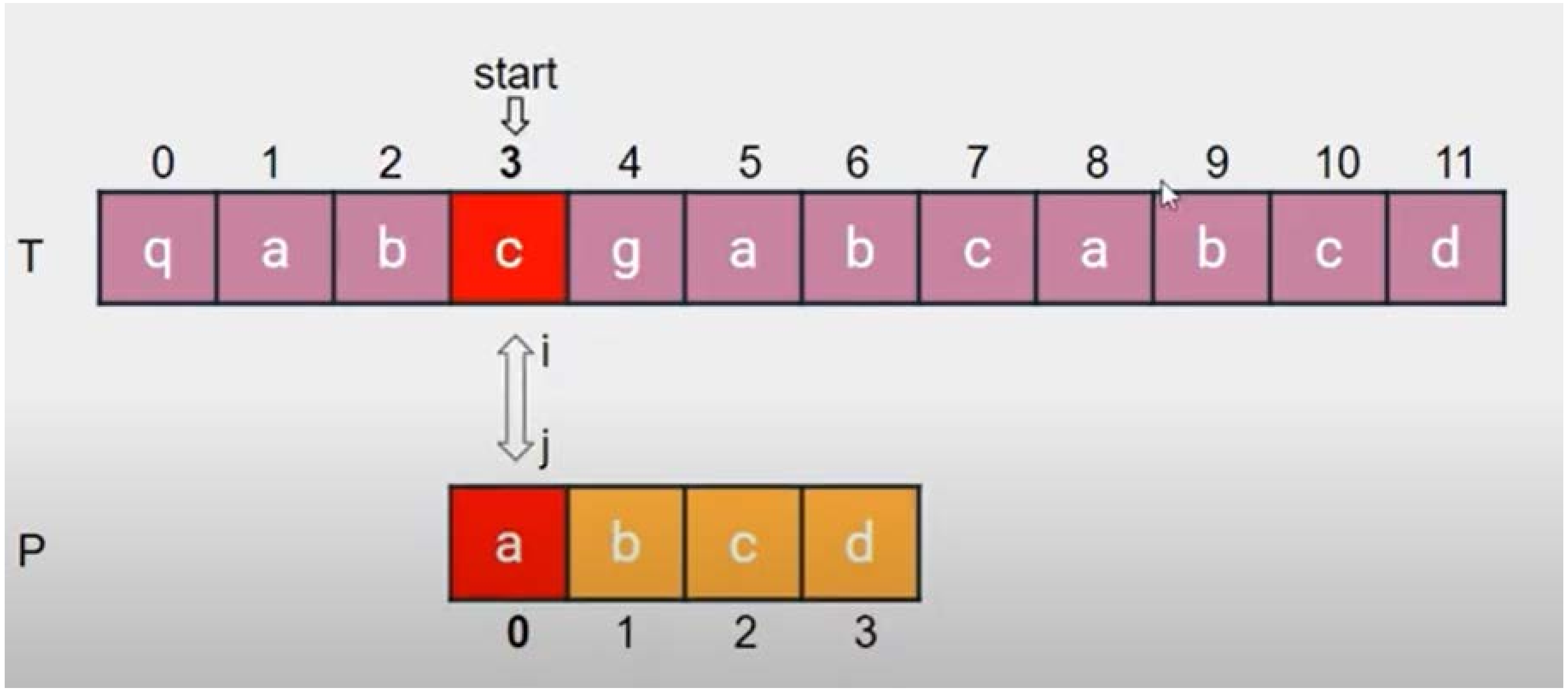
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



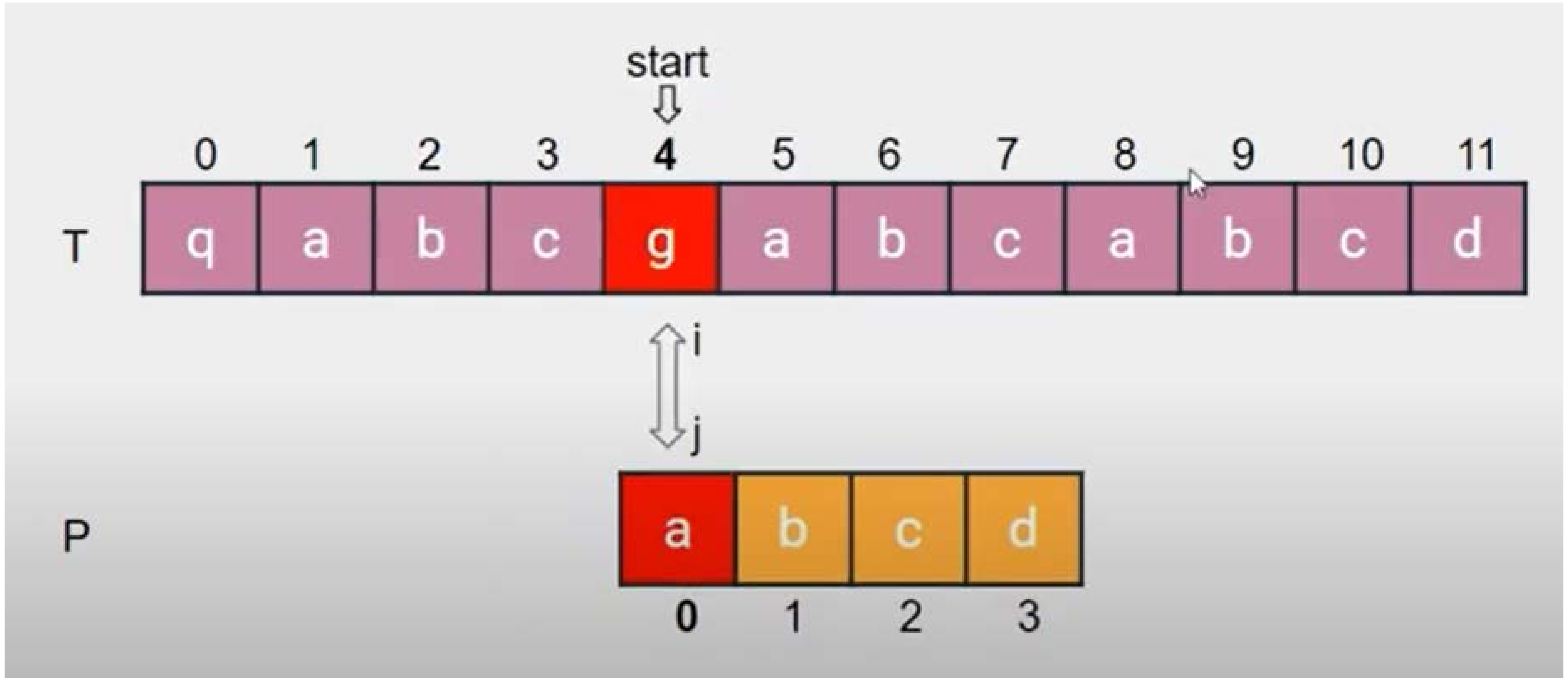
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



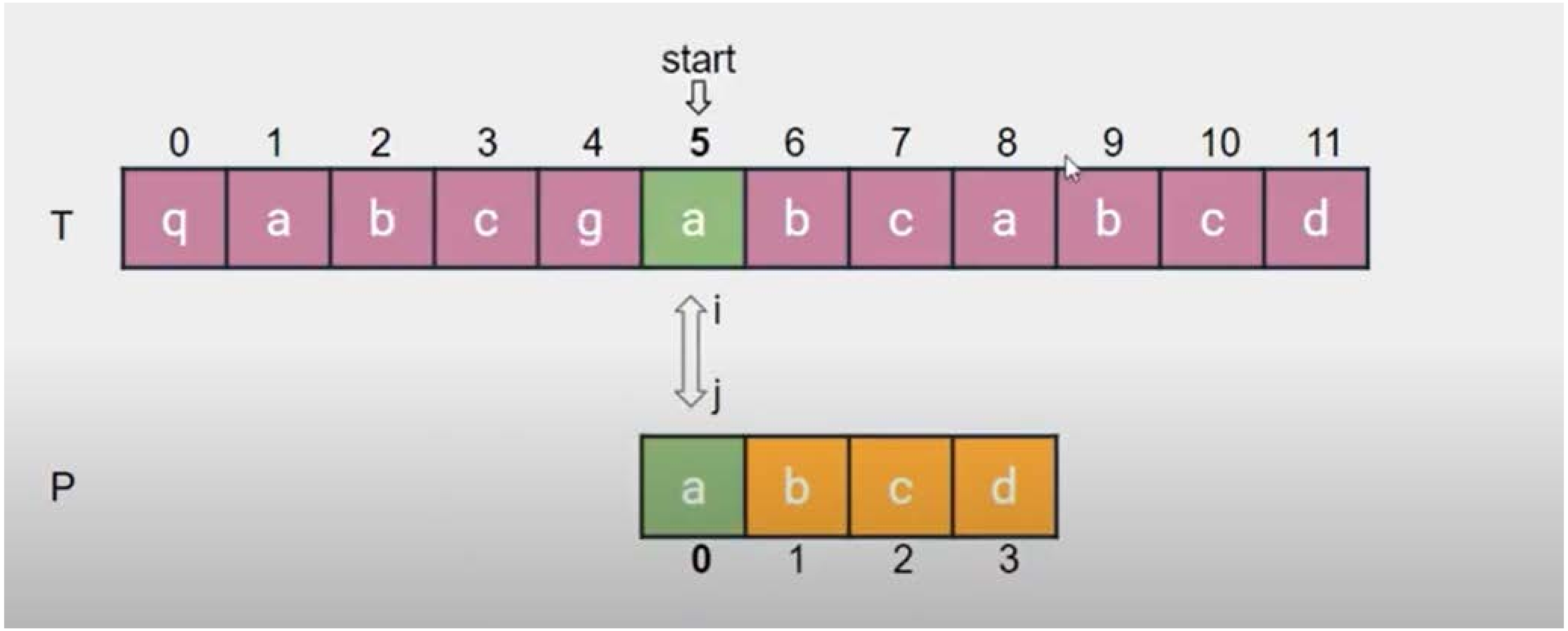
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



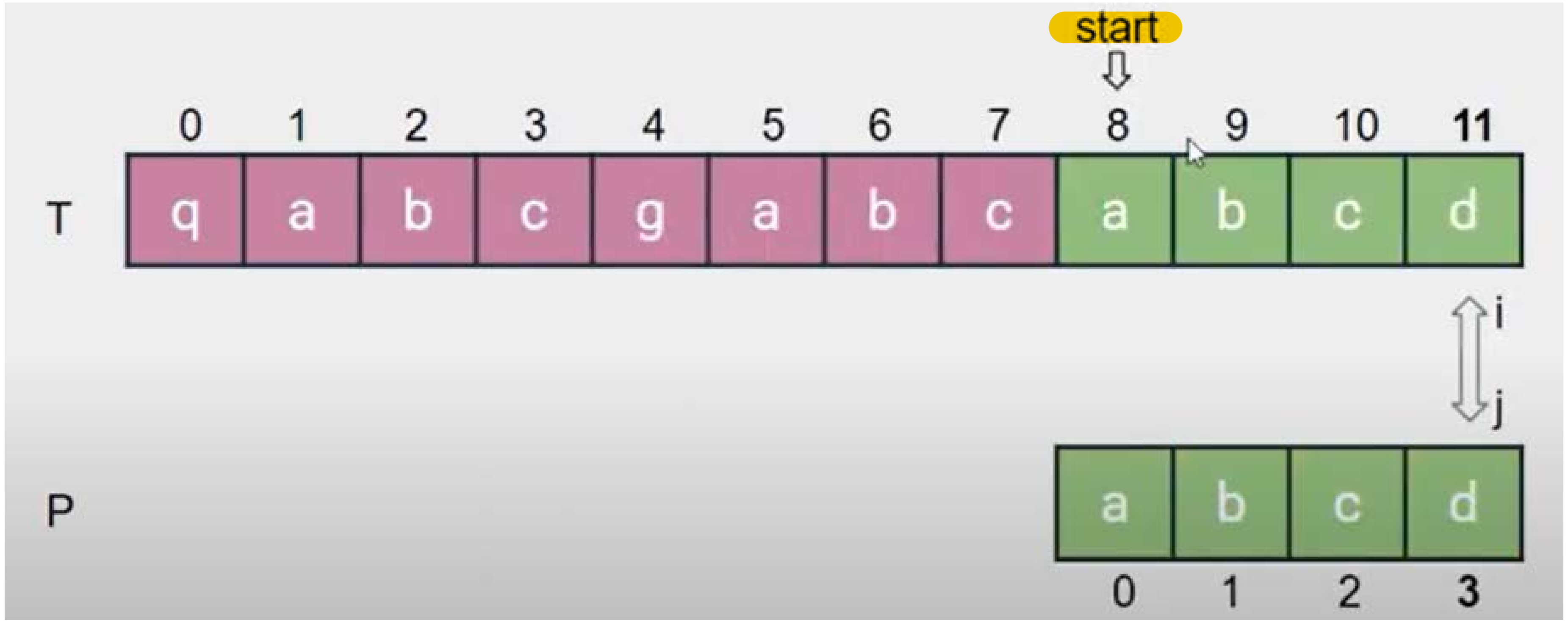
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



پیدا کردن یک الگو در یک رشته- الگوریتم ساده

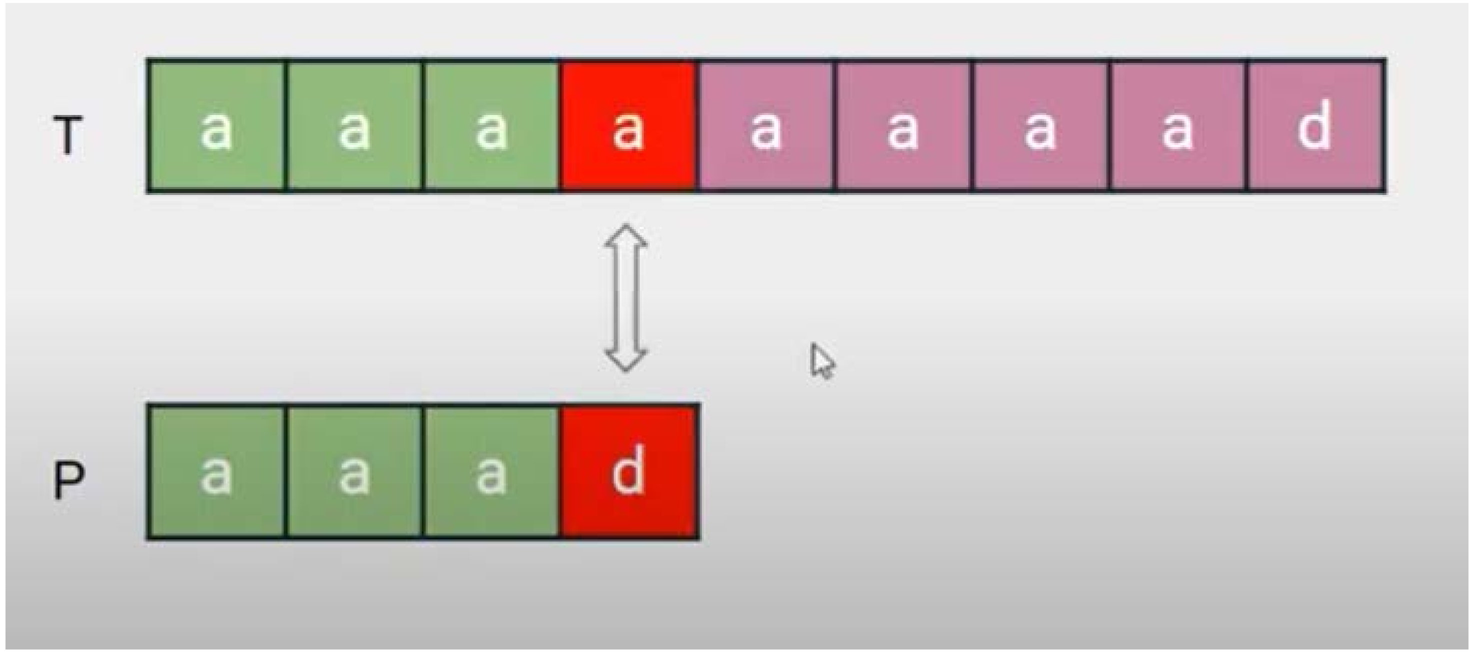


پیدا کردن یک الگو در یک رشته- الگوریتم ساده



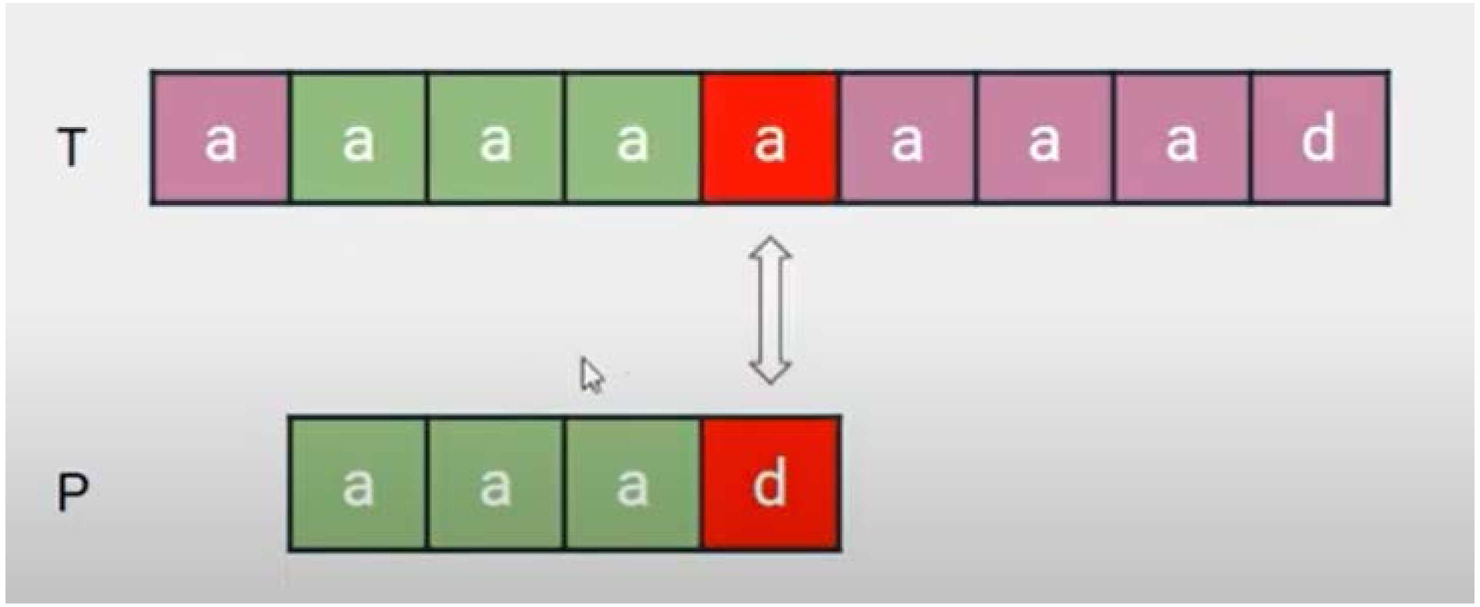
پیدا کردن یک الگو در یک رشته- الگوریتم ساده

- بدترین حالت



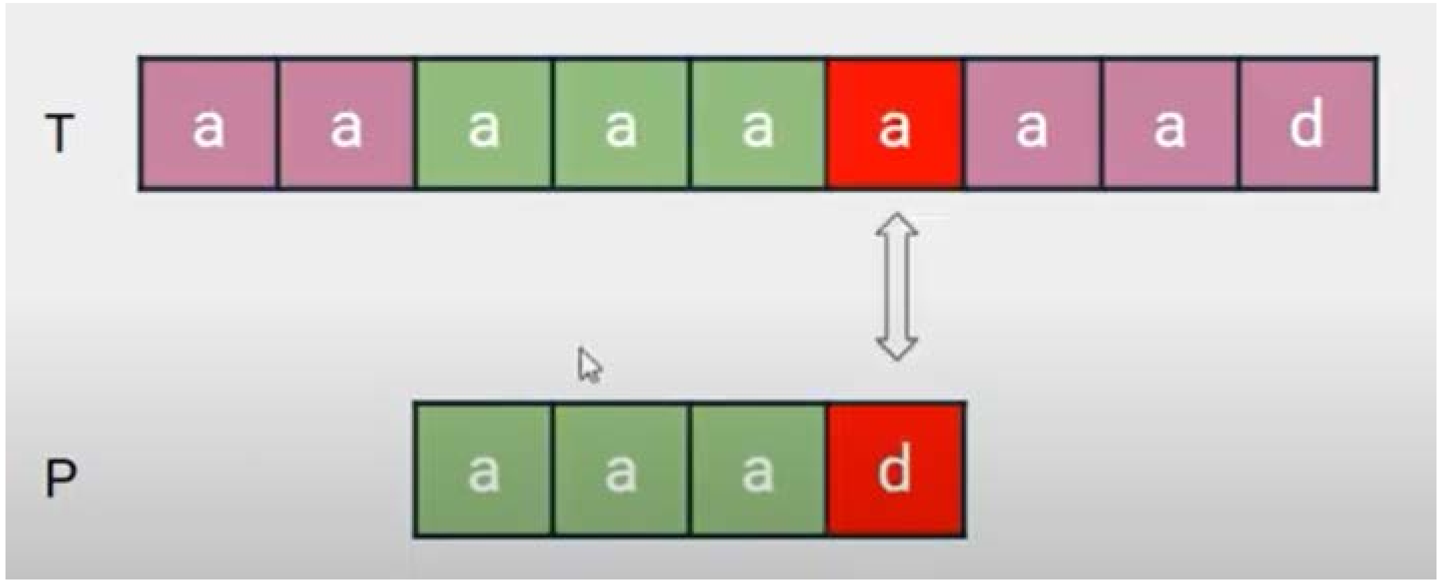
پیدا کردن یک الگو در یک رشته- الگوریتم ساده

- بدترین حالت



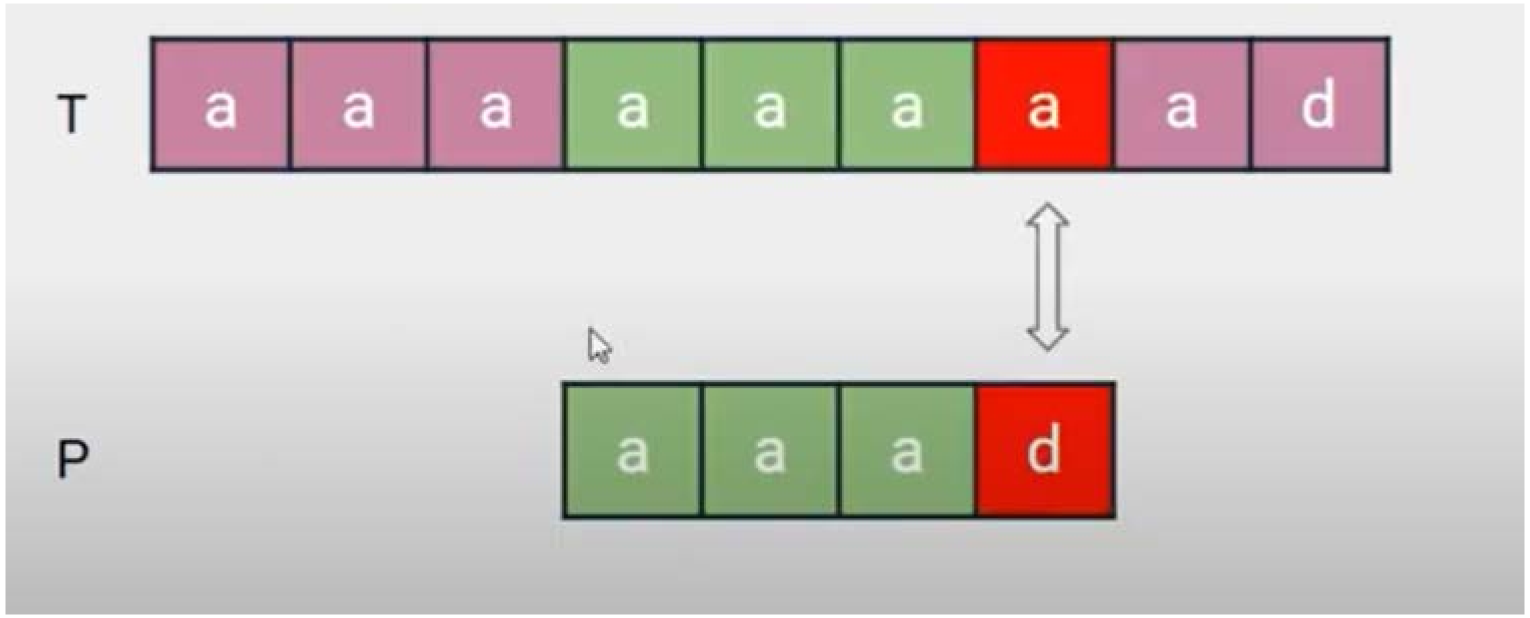
پیدا کردن یک الگو در یک رشته- الگوریتم ساده

- بدترین حالت



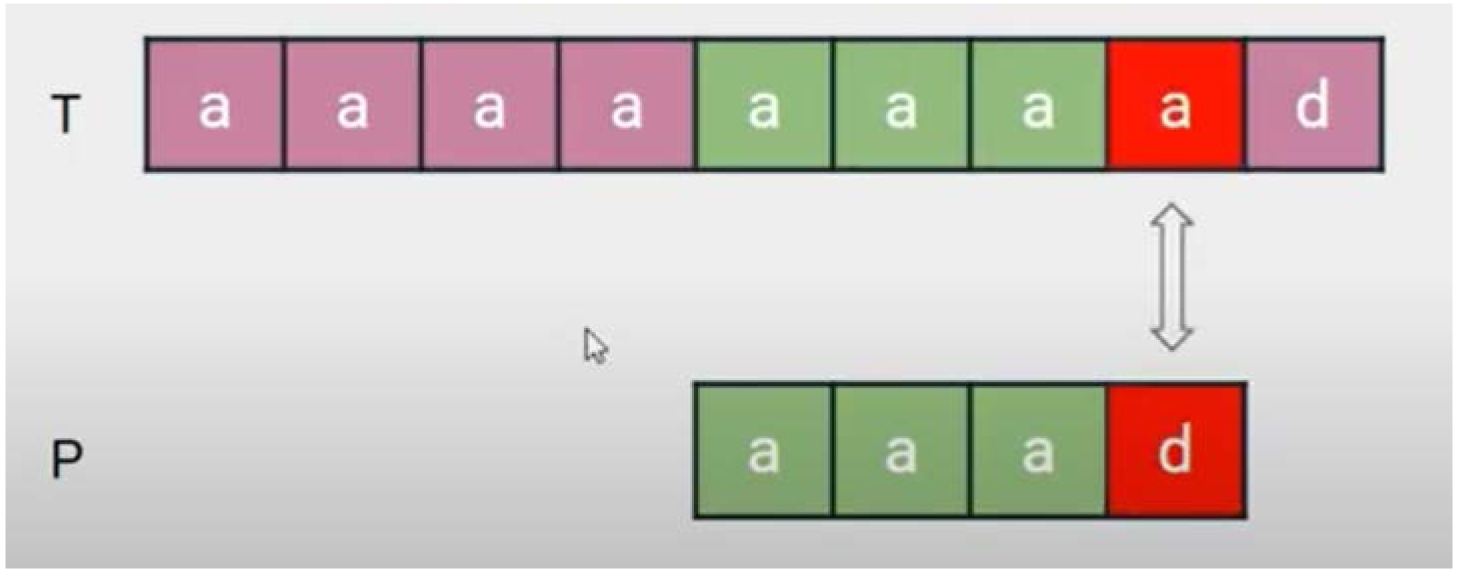
پیدا کردن یک الگو در یک رشته- الگوریتم ساده

- بدترین حالت



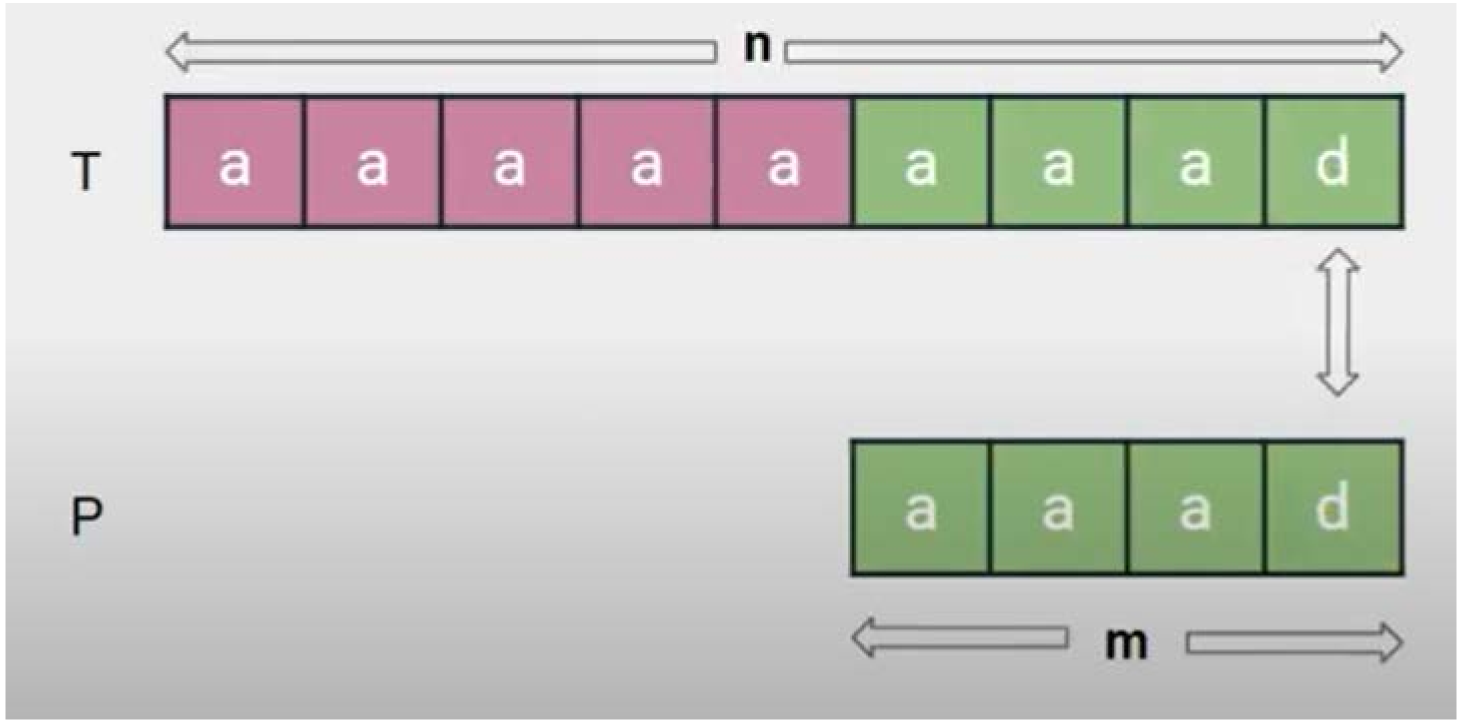
پیدا کردن یک الگو در یک رشته- الگوریتم ساده

- بدترین حالت

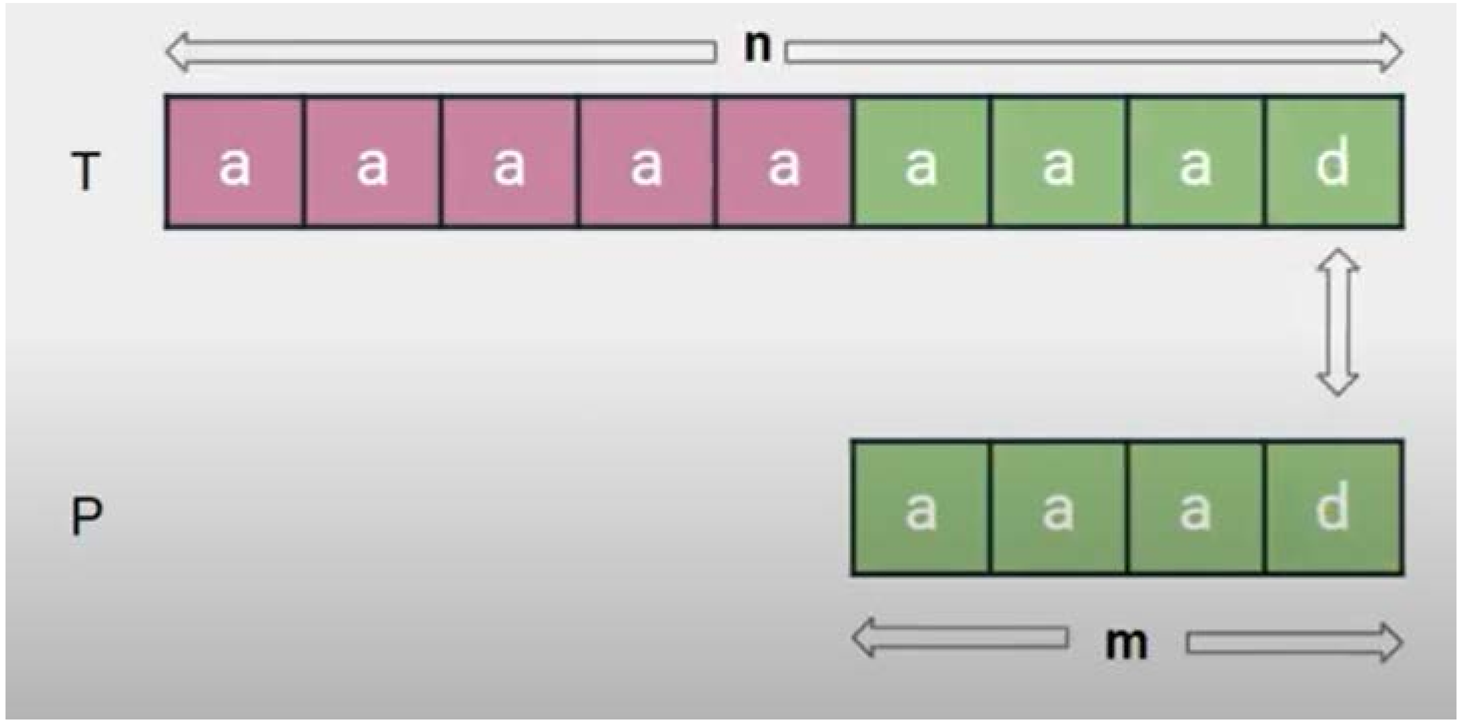


پیدا کردن یک الگو در یک رشته- الگوریتم ساده

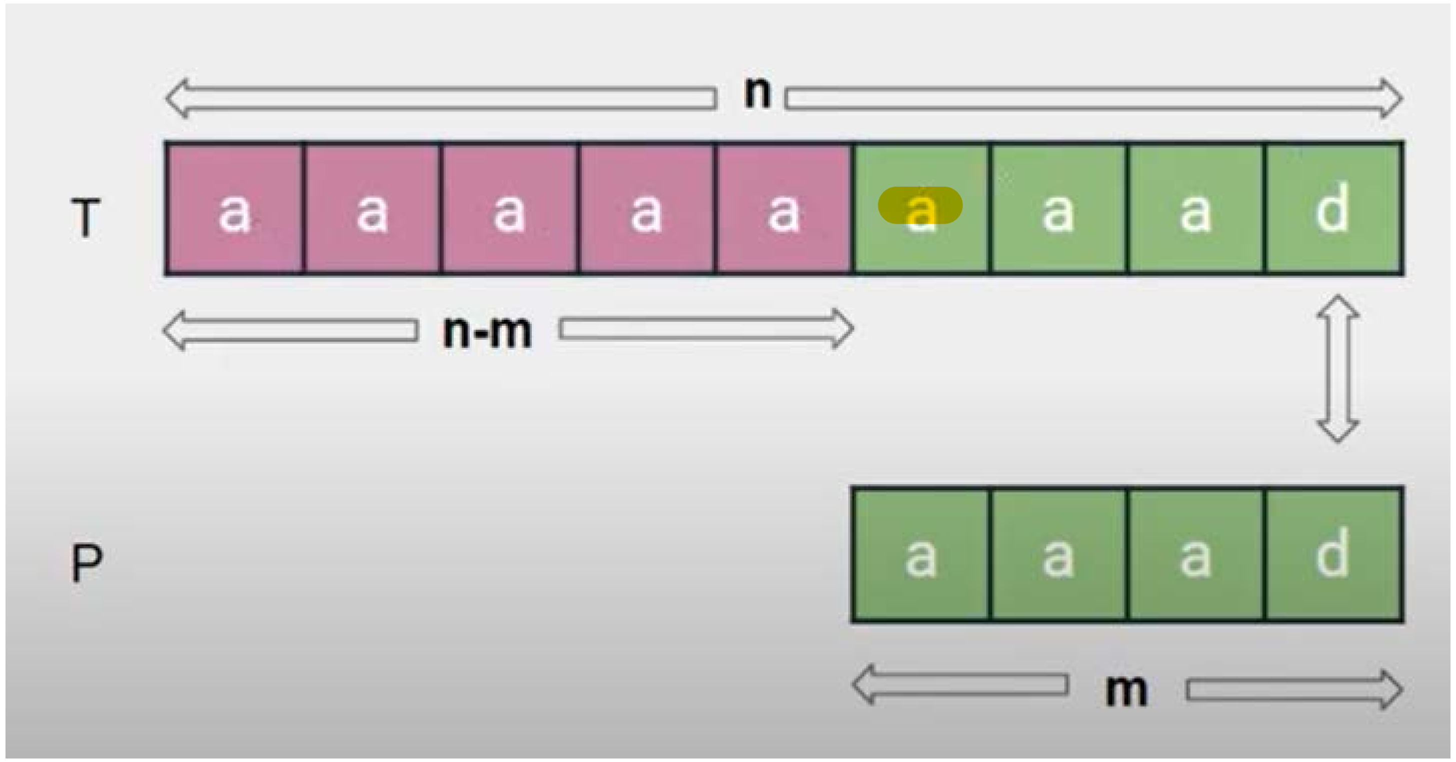
- بدترین حالت



تحلیل پیچیدگی الگوریتم ساده



تحلیل پیچیدگی الگوریتم ساده

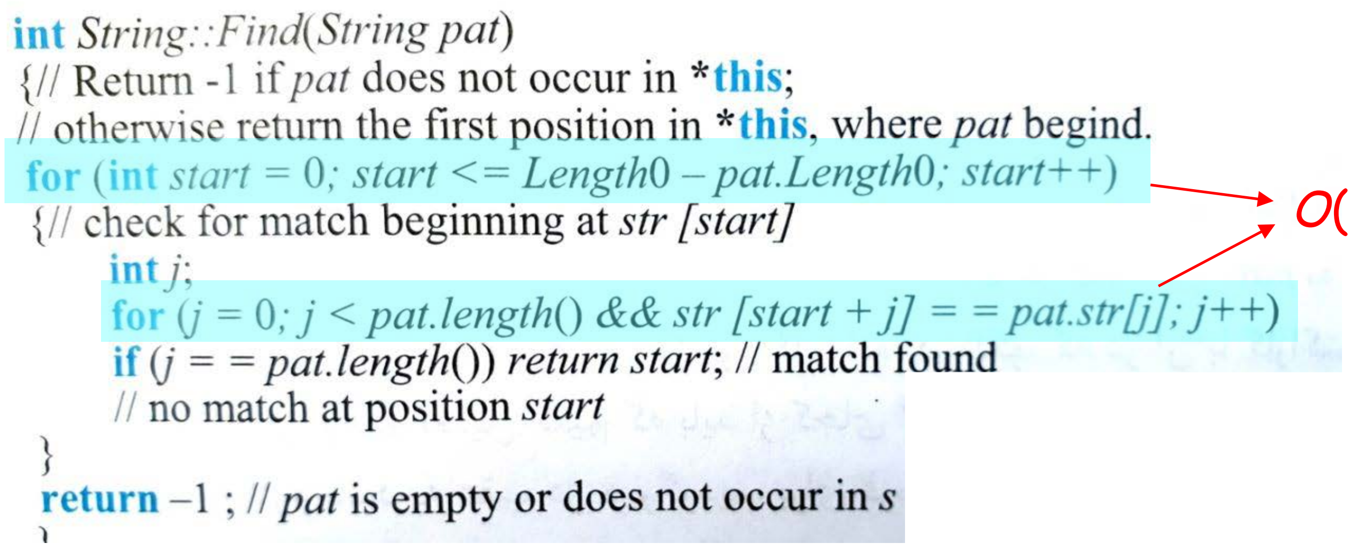


O([n-m+1]*m)

~ O(nm)

when n >>> m

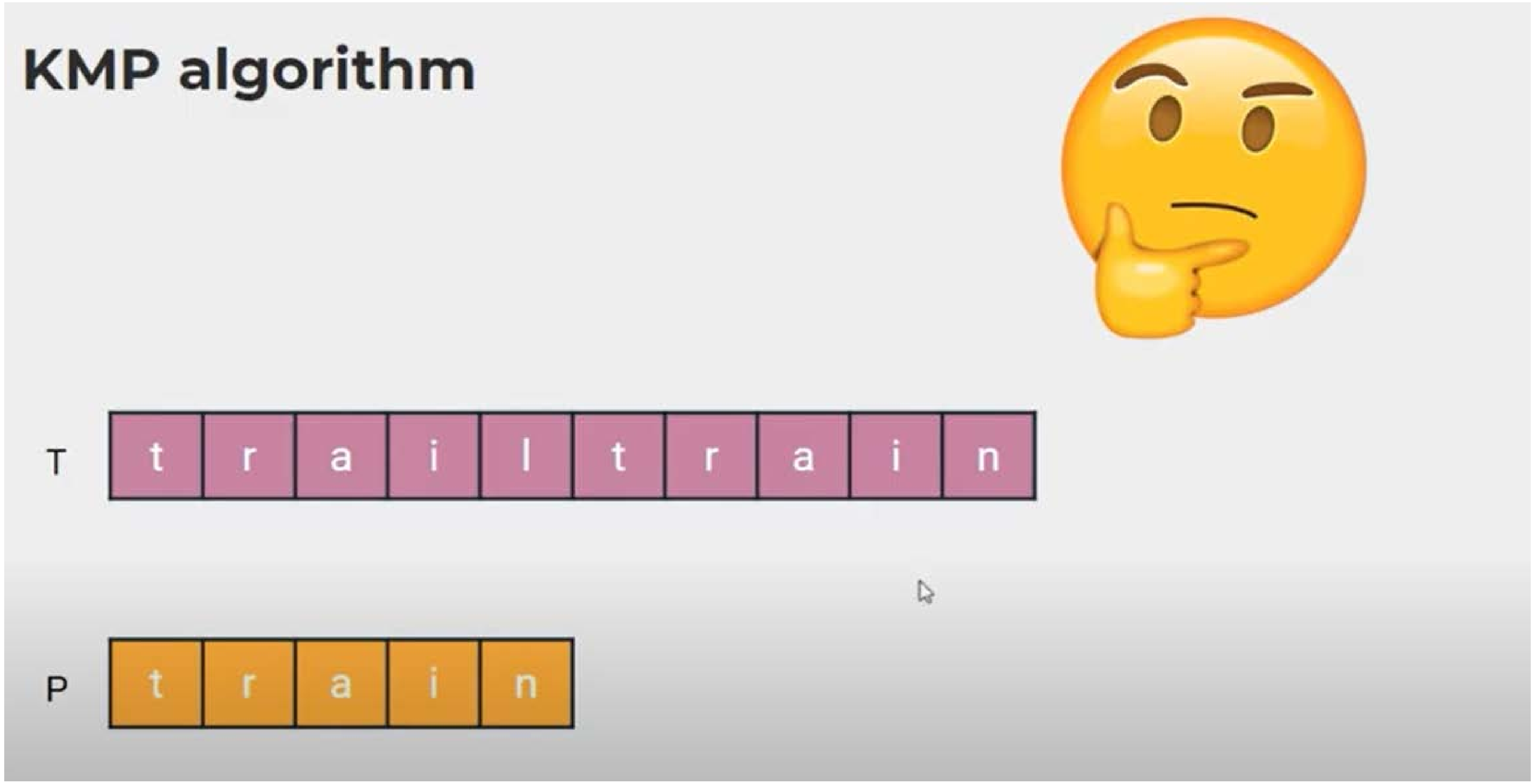
پیدا کردن یک الگو در یک رشته- الگوریتم ساده



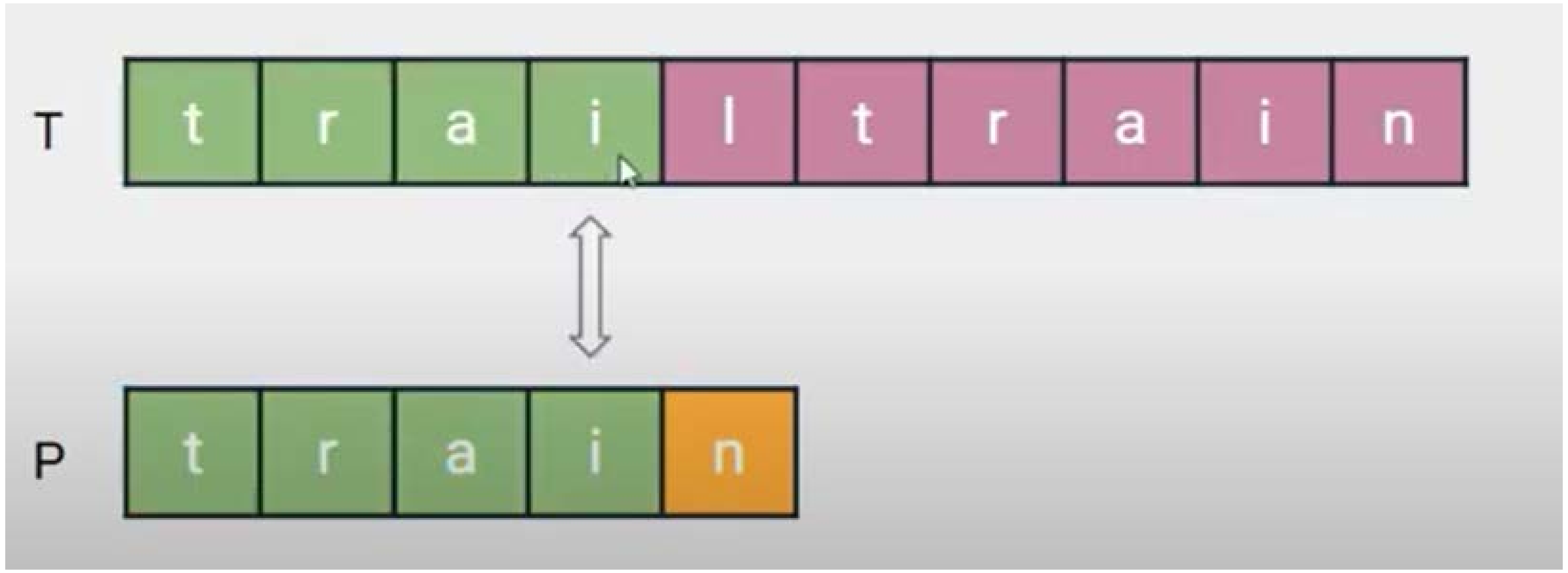
```
int String::Find(String pat)
{// Return -1 if pat does not occur in *this;
// otherwise return the first position in *this, where pat begind.
 for (int start = 0; start <= Length0 – pat.Length0; start++)
 {// check for match beginning at str [start]
      int j;
      for (j = 0; j < pat.length() && str [start + j] = = pat.str[j]; j++)
      if (j = = pat.length()) return start; // match found
      // no match at position start
 }
 return –1 ; // pat is empty or does not occur in s
}
```

O(lengthS * LengthP)

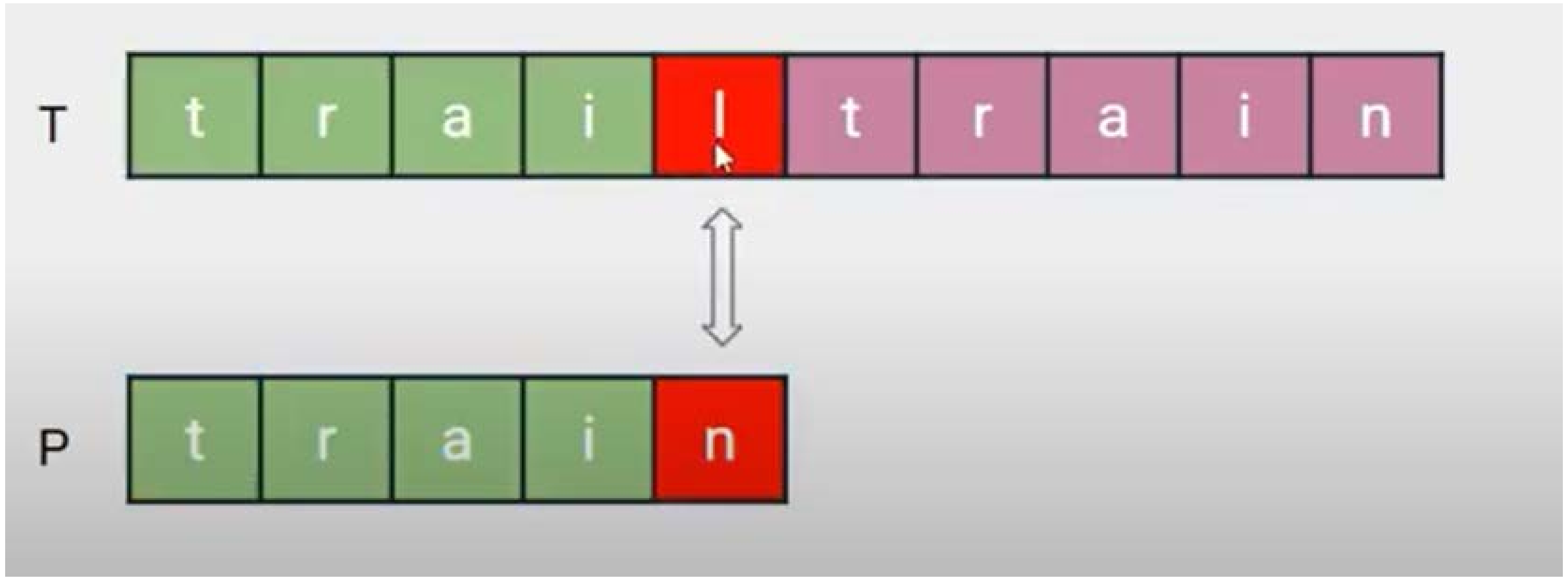
الگوریتم کنوث-موریس-پرات



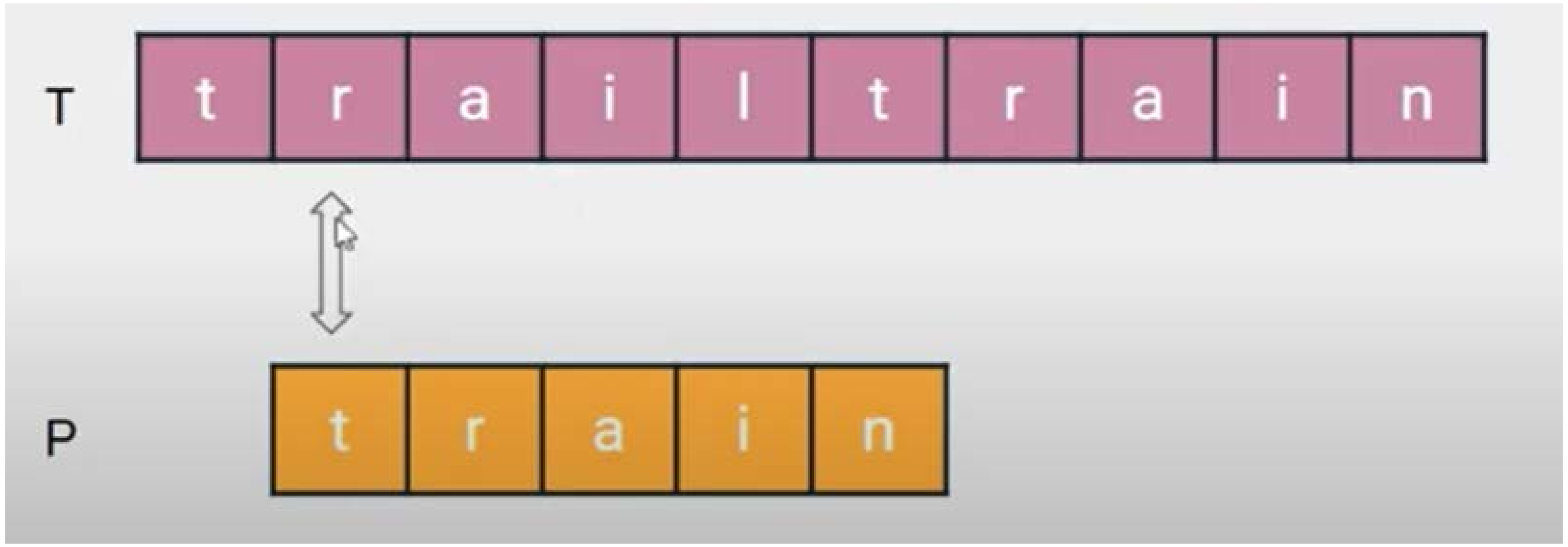
الگوریتم کنوث-موریس-پرات KMP



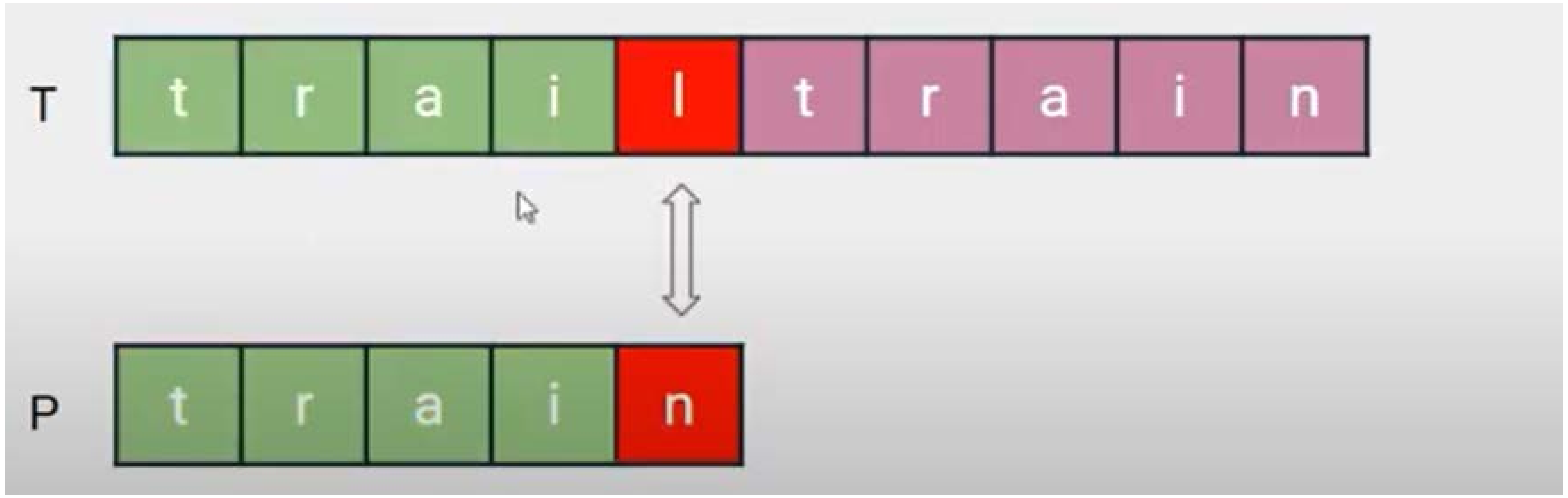
الگوریتم کنوث-موریس-پرات KMP



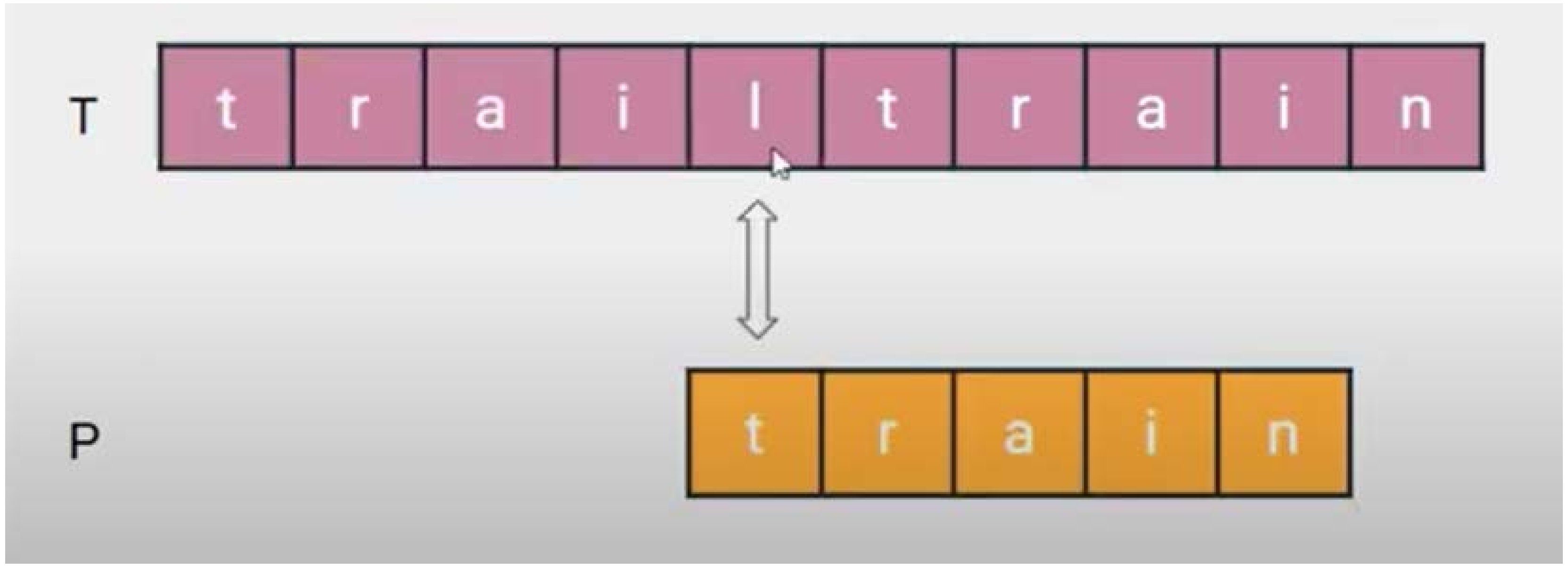
الگوریتم کنوث-موریس-پرات KMP



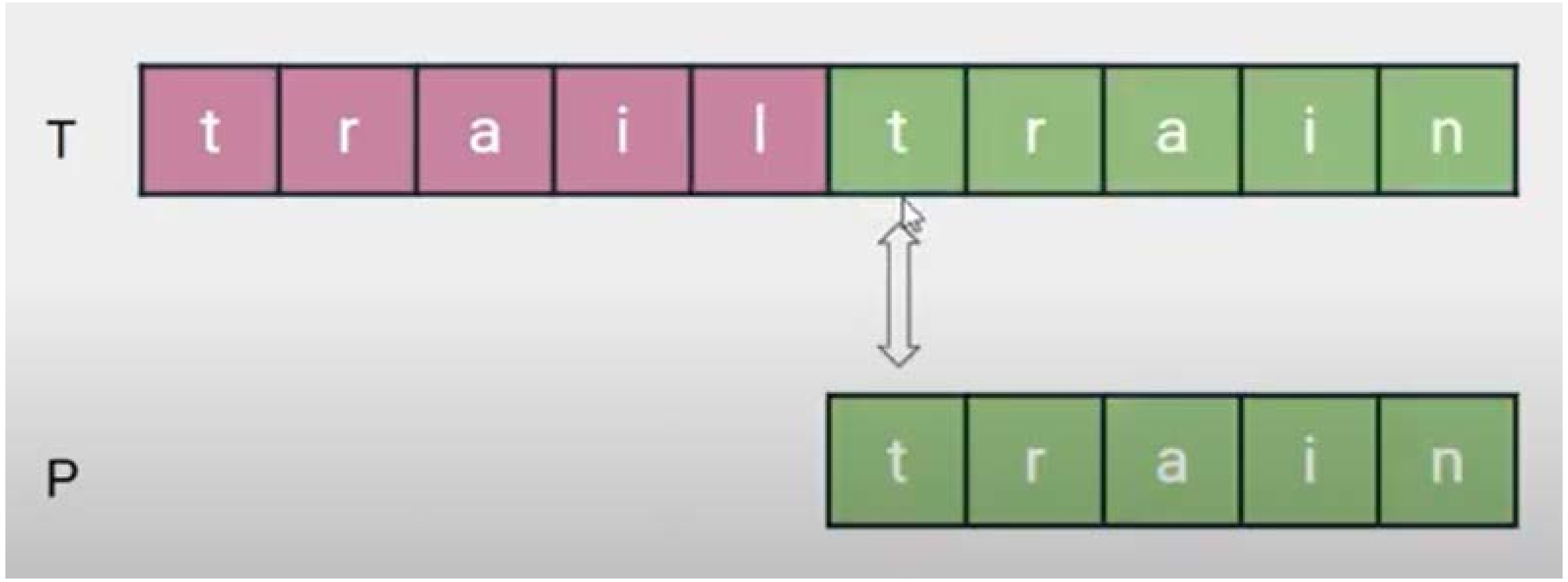
الگوریتم کنوث-موریس-پرات KMP



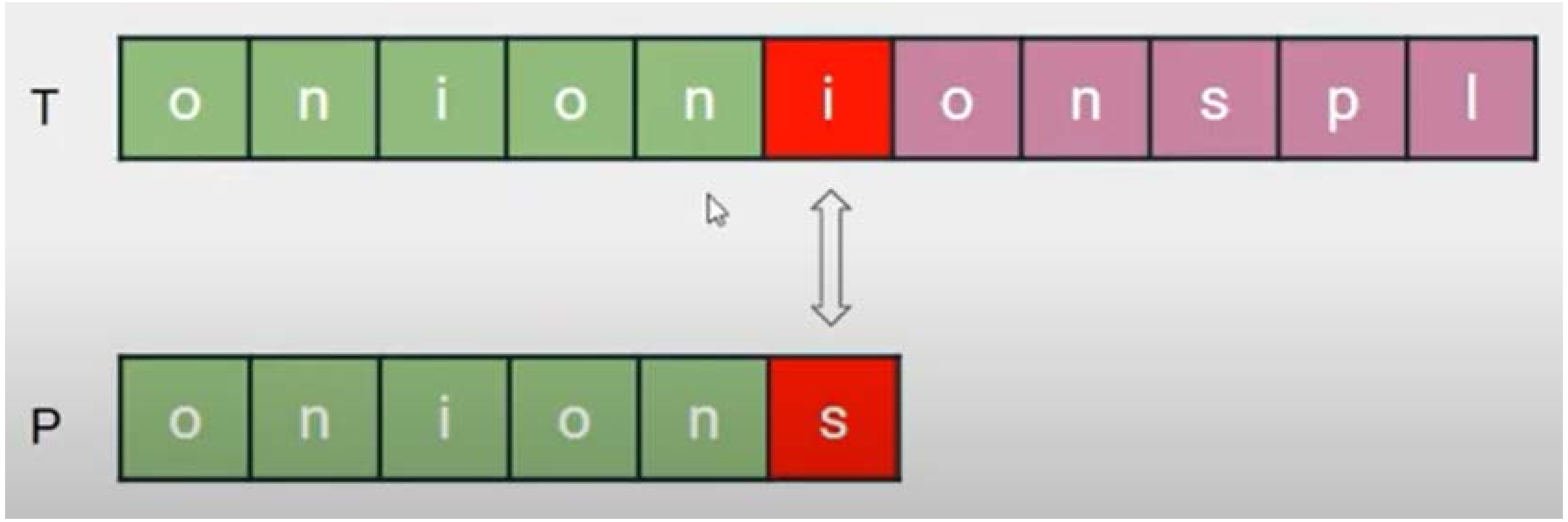
الگوریتم کنوث-موریس-پرات KMP



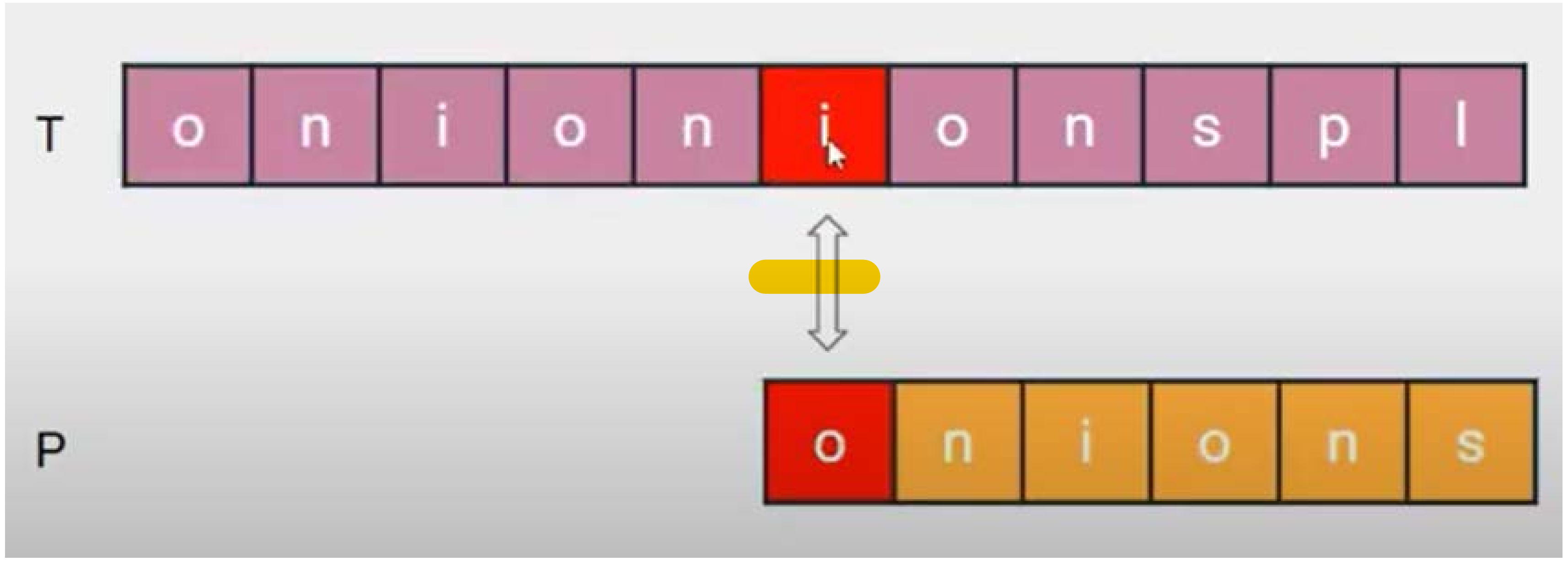
الگوریتم کنوث-موریس-پرات KMP



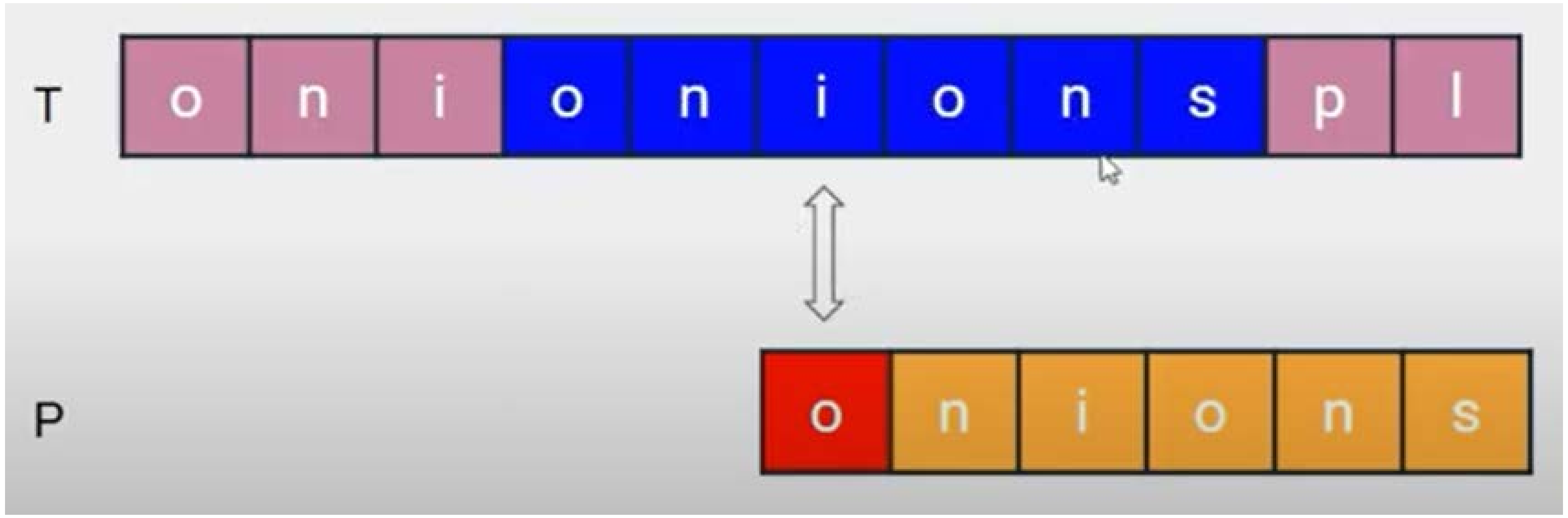
الگوریتم کنوث-موریس-پرات KMP



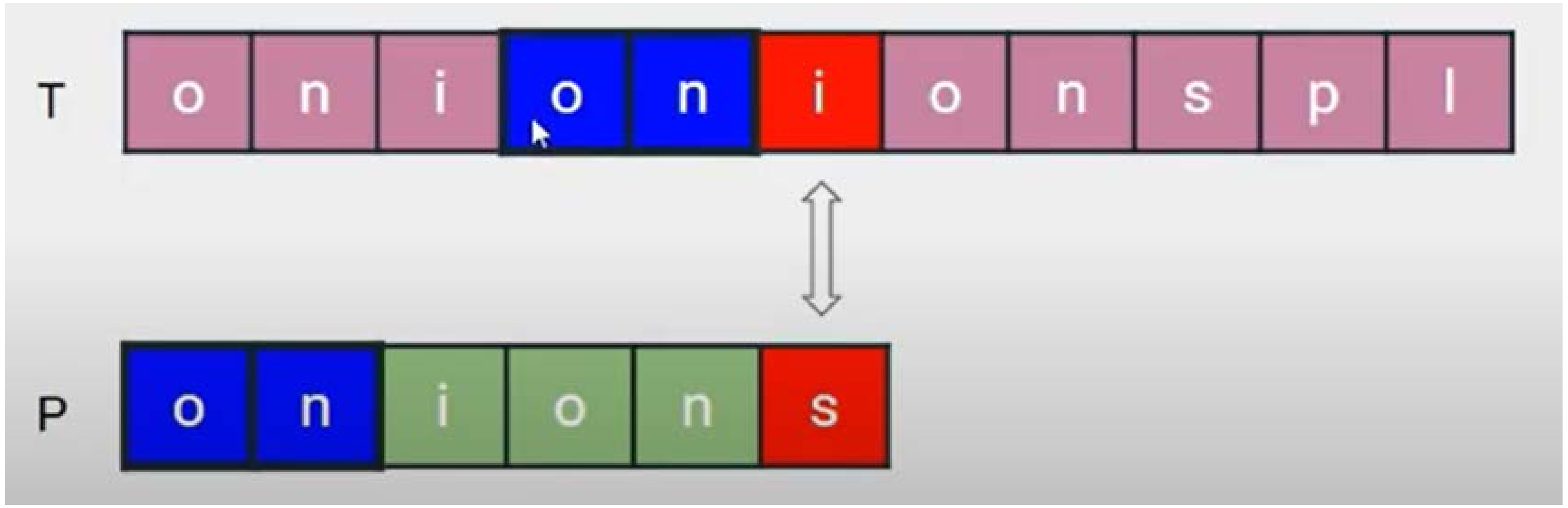
الگوریتم کنوث-موریس-پرات KMP



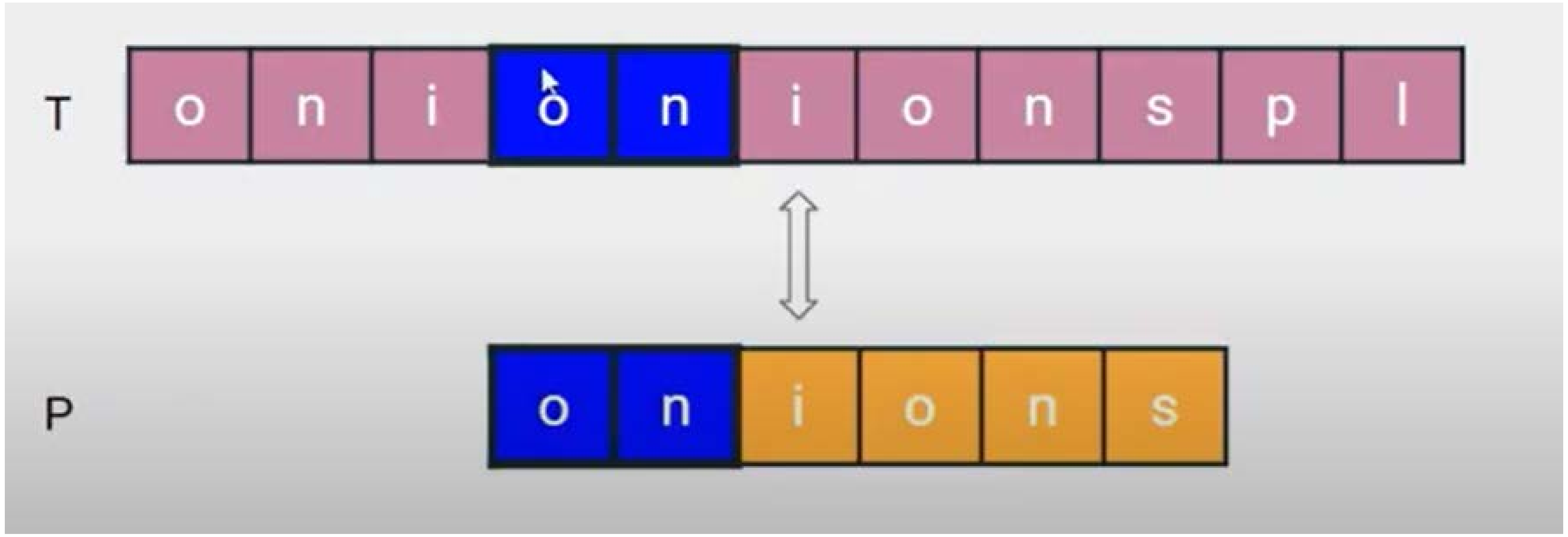
الگوریتم کنوث-موریس-پرات KMP



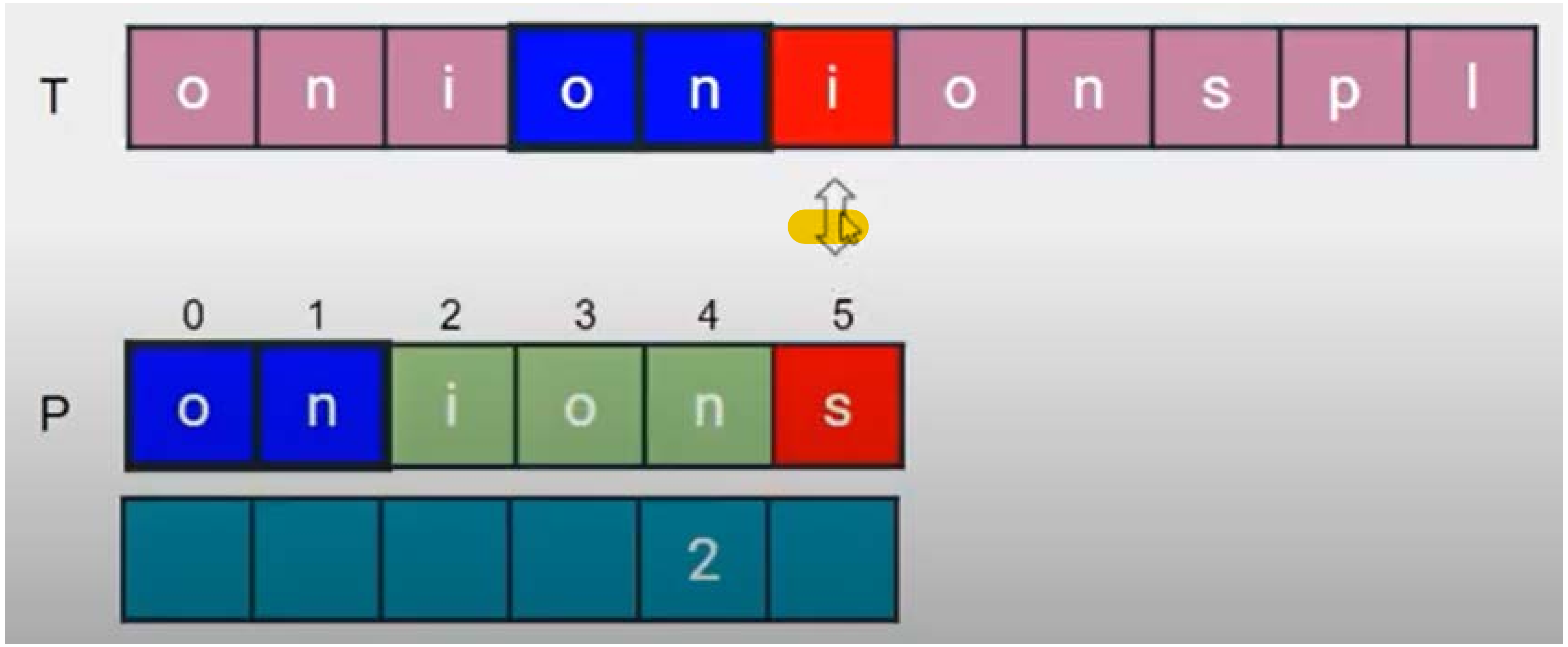
الگوریتم کنوث-موریس-پرات KMP



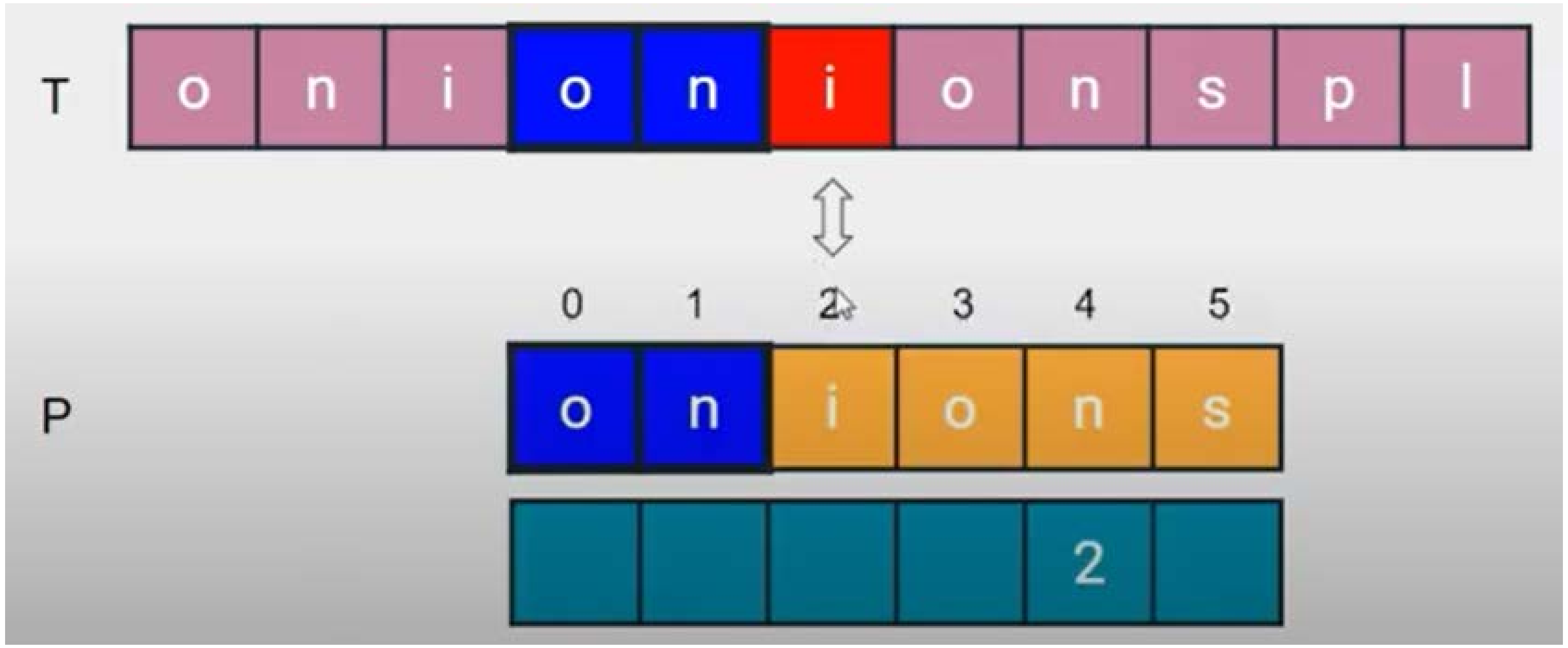
الگوریتم کنوث-موریس-پرات KMP



الگوریتم کنوث-موریس-پرات KMP

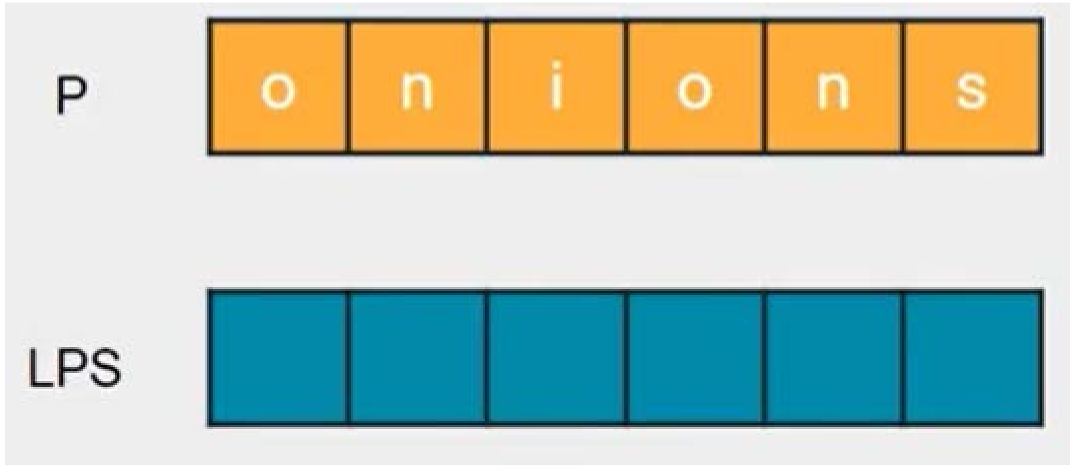


الگوریتم کنوث-موریس-پرات KMP



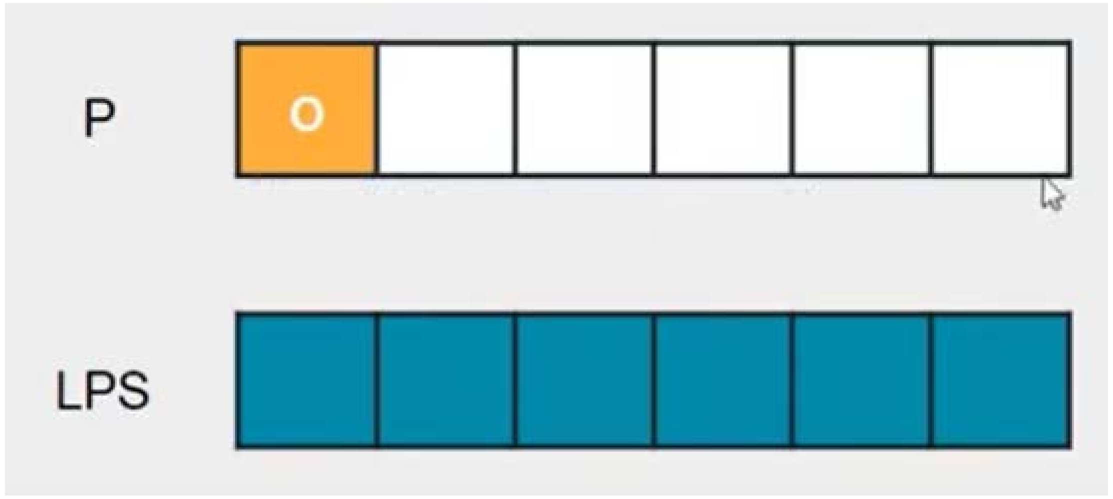
LPS array

- Longest prefix that is also suffix



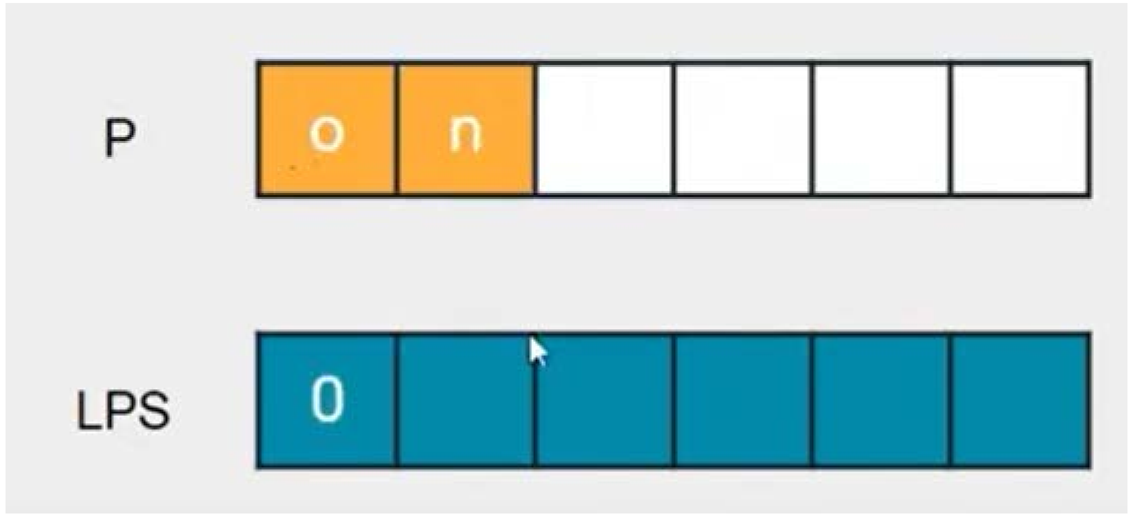
LPS array

- Longest prefix that is also suffix



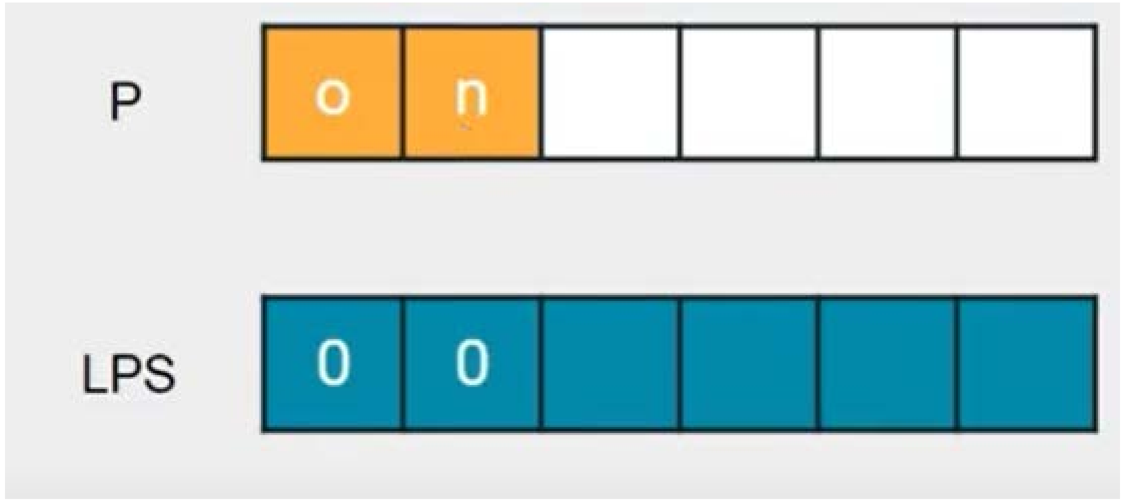
LPS array

- Longest prefix that is also suffix



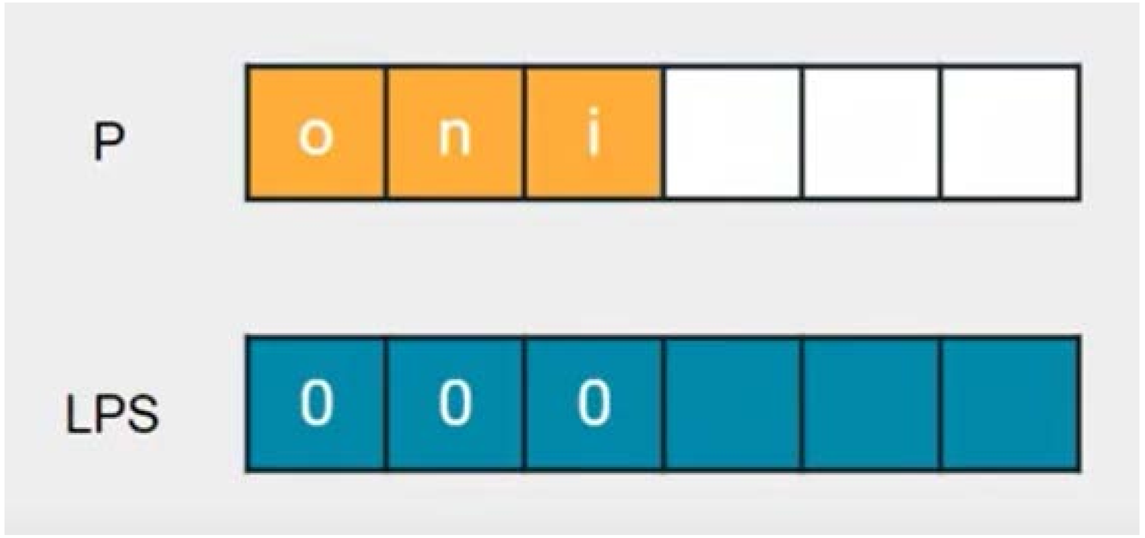
LPS array

- Longest prefix that is also suffix



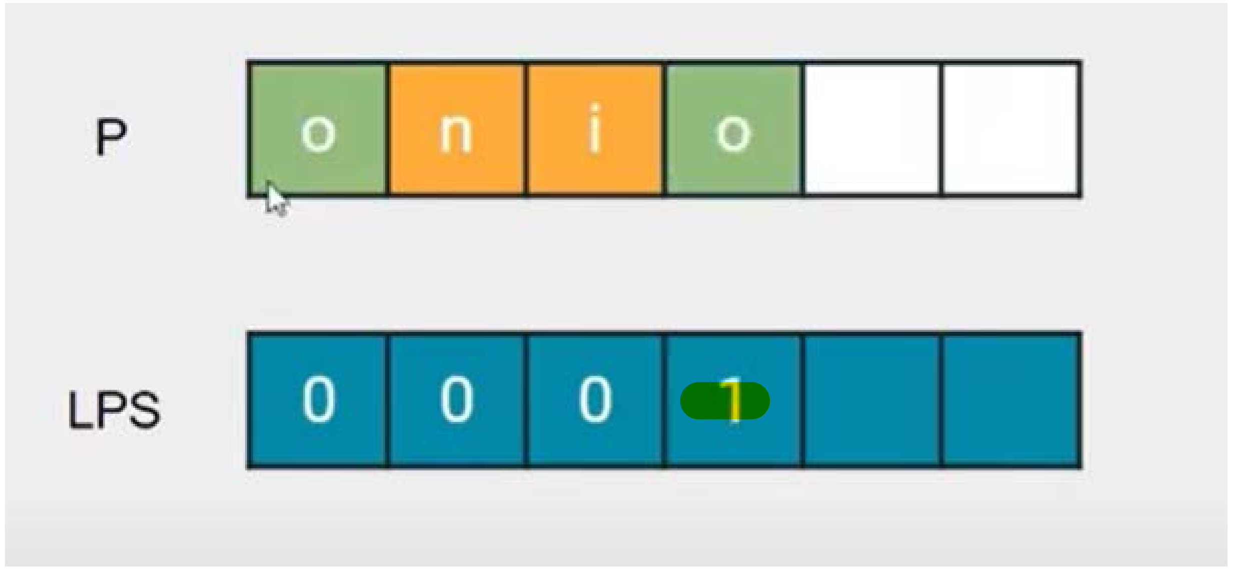
LPS array

- Longest prefix that is also suffix



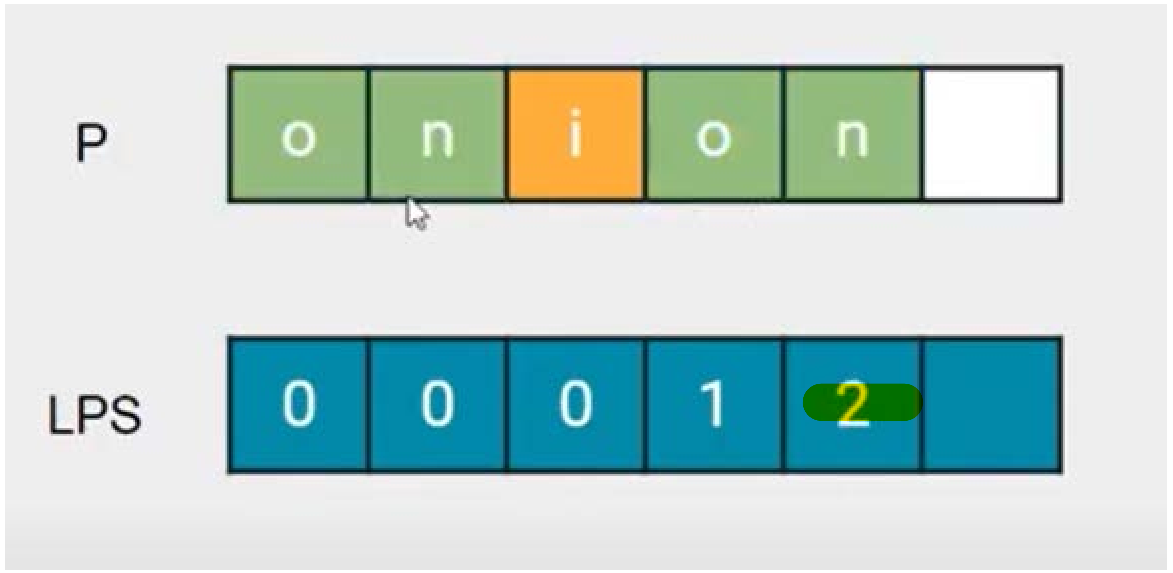
LPS array

- Longest prefix that is also suffix



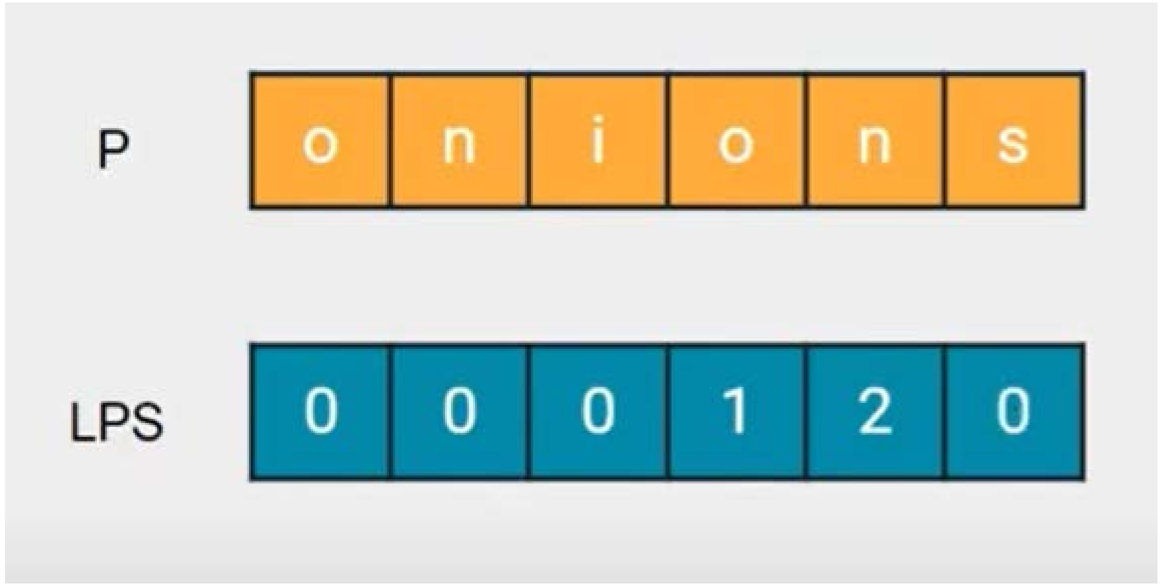
LPS array

- Longest prefix that is also suffix



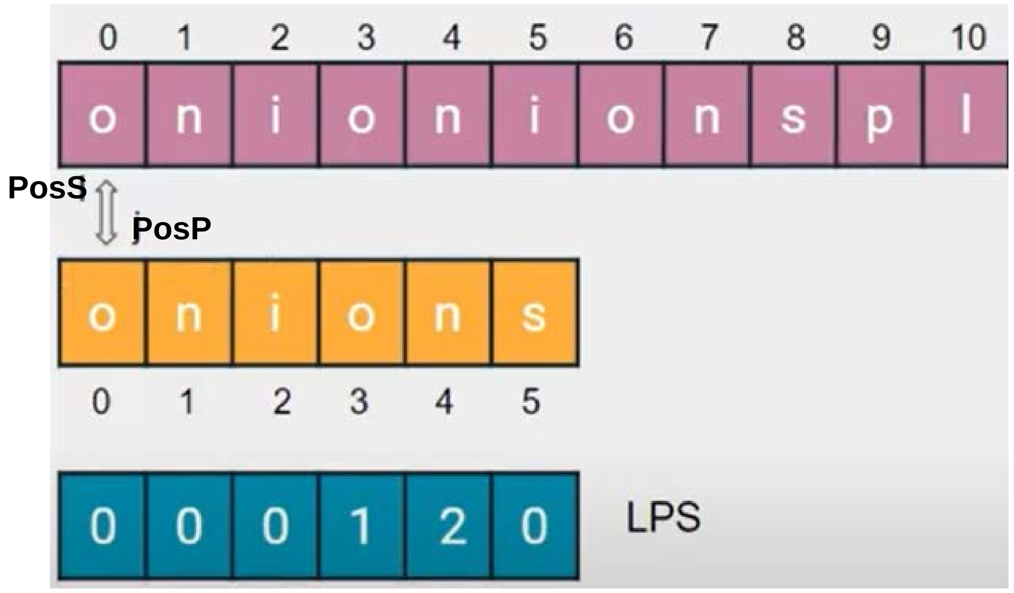
LPS array

- Longest prefix that is also suffix



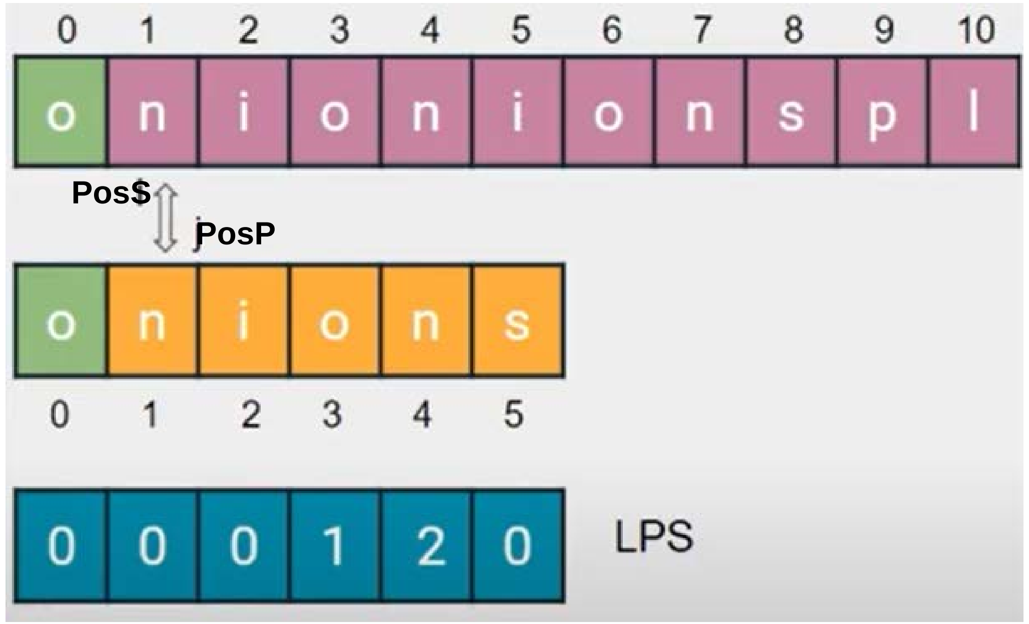
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {
                   PosP++;  PosS++;
          }
          else
                  if ( PosP == 0) PosS++;
                  else PosP=  pat.f [ PosP -1 ]   ;
   if ( PosP<lengthP)  return -1;
   else  return PosS- lengthp;
}   // end of FastFind
```



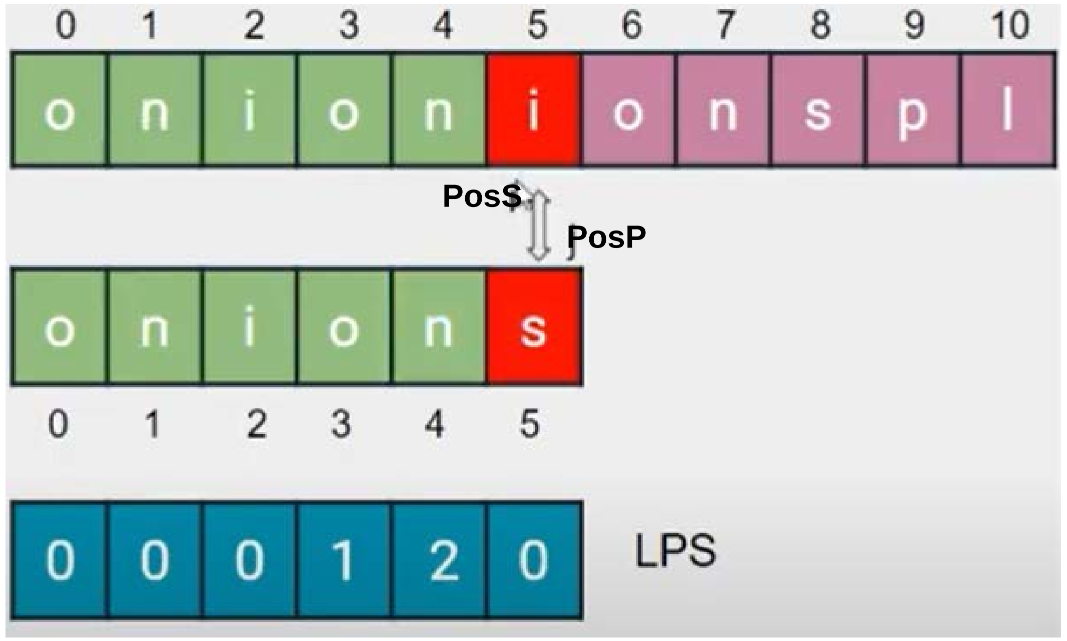
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {
                  PosP++;  PosS++;
          }
          else
                  if ( PosP == 0) PosS++;
                  else PosP=  pat.f [ PosP -1 ] ;
  if ( PosP<lengthP)  return -1;
  else  return PosS- lengthp;
}   // end of FastFind
```



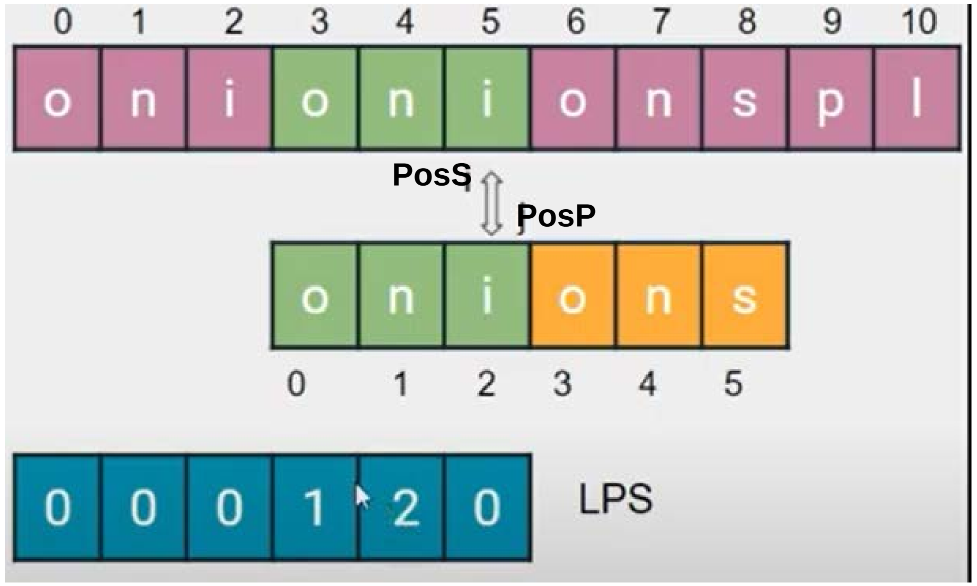
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
           if (pat.str [PosP] == str [PosS] ) {
                    PosP++;  PosS++;
           }
           else
                    if ( PosP == 0) PosS++;
                    else PosP=  pat.f [ PosP -1 ]   ;
   if ( PosP<lengthP)  return -1;
   else  return PosS- lengthp;
}   // end of FastFind
```



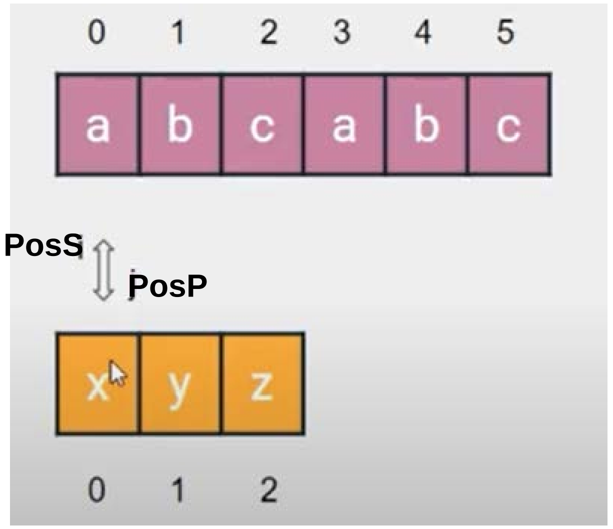
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
           if (pat.str [PosP] == str [PosS] ) {
                    PosP++;  PosS++;
           }
           else
                    if ( PosP  == 0) PosS++;
                    else PosP=  pat.f [ PosP -1 ]   ;
   if ( PosP<lengthP)  return -1;
   else  return PosS- lengthp;
} // end of FastFind
```



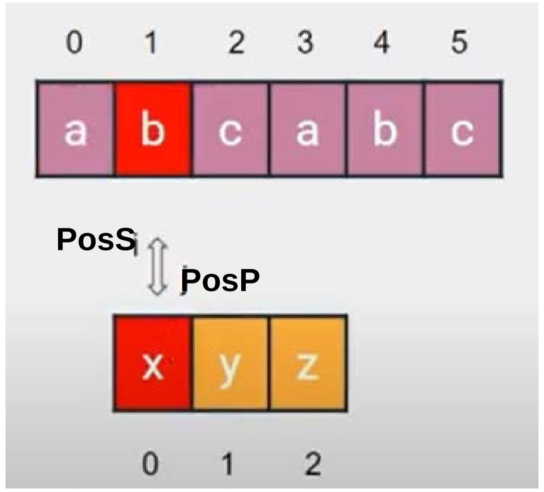
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
 //Determine if pat is a substring of s
 int PosP=0, PosS=0;
 int  lengthP= pat.length(),  lengthS= length();
 while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {       //character match
                   PosP++;  PosS++;
          }
          else
               if ( PosP == 0) PosS++;
               else PosP=  pat.f [ PosP -1 ]   ;
  if ( PosP<lengthP)  return -1;
  else  return PosS- lengthp;
}   // end of FastFind
```



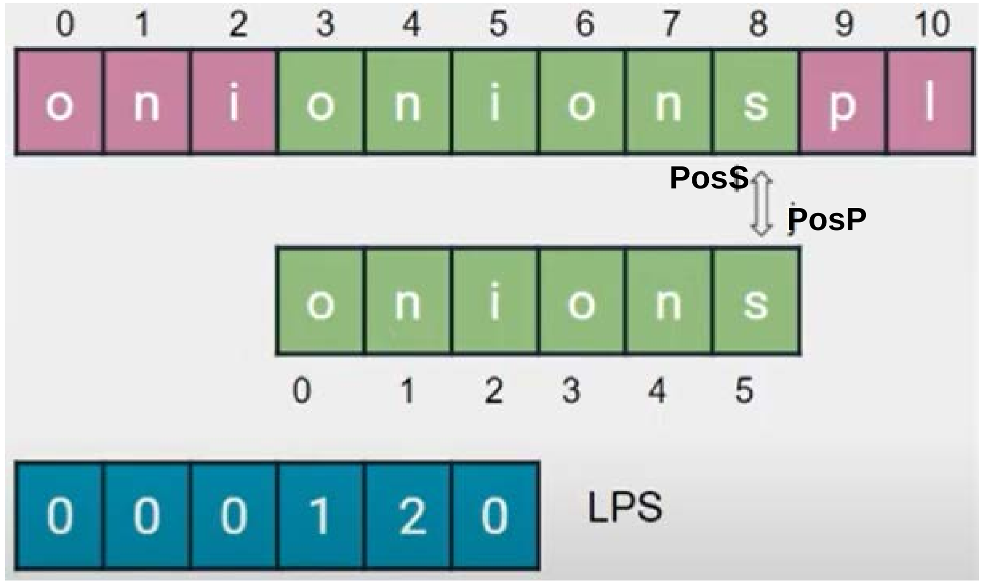
الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
            if (pat.str [PosP] == str [PosS] ) {       //character match
                        PosP++;  PosS++;
            }
            else
                     if ( PosP == 0) PosS++;
                     else PosP=  pat.f [ PosP -1 ]   ;
   if ( PosP<lengthP)  return -1;
   else  return PosS- lengthp;
}   // end of FastFind
```



الگوریتم کنوث-موریس-پرات KMP

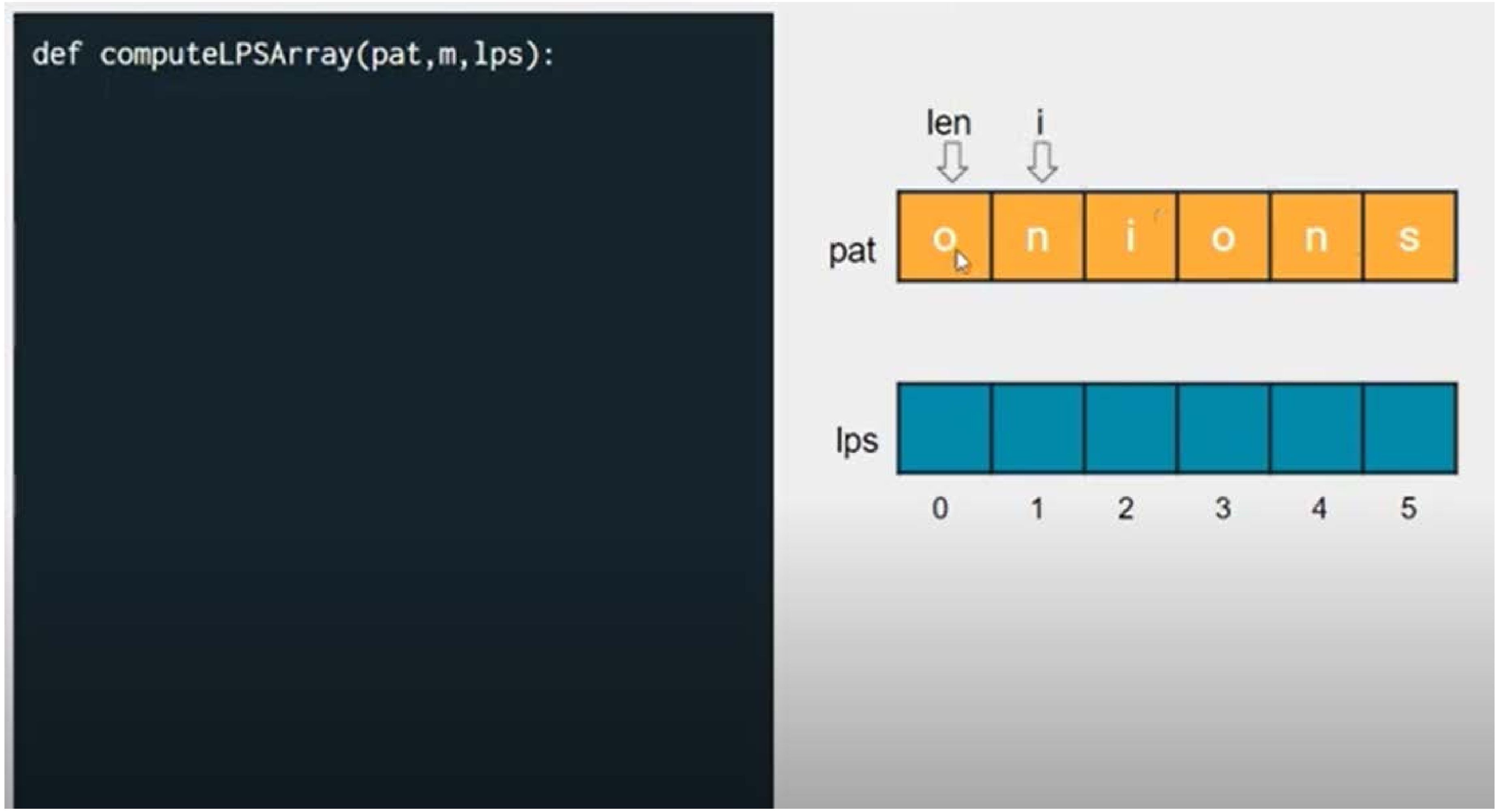
```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {
                  PosP++;  PosS++;
          }
          else
                  if ( PosP == 0) PosS++;
                  else PosP=  pat.f [ PosP -1 ]   ;
  if ( PosP<lengthP)  return -1;
  else  return PosS- lengthp;
}   // end of FastFind
```



تابع شکست در الگوریتم KMP

$$f(j) = \begin{cases} \text{مقدار بزرگترین } k < j & p_0 \cdots p_k = p_{j-k} \cdots p_j \\ 0 & \text{در غیر این صورت} \end{cases}$$

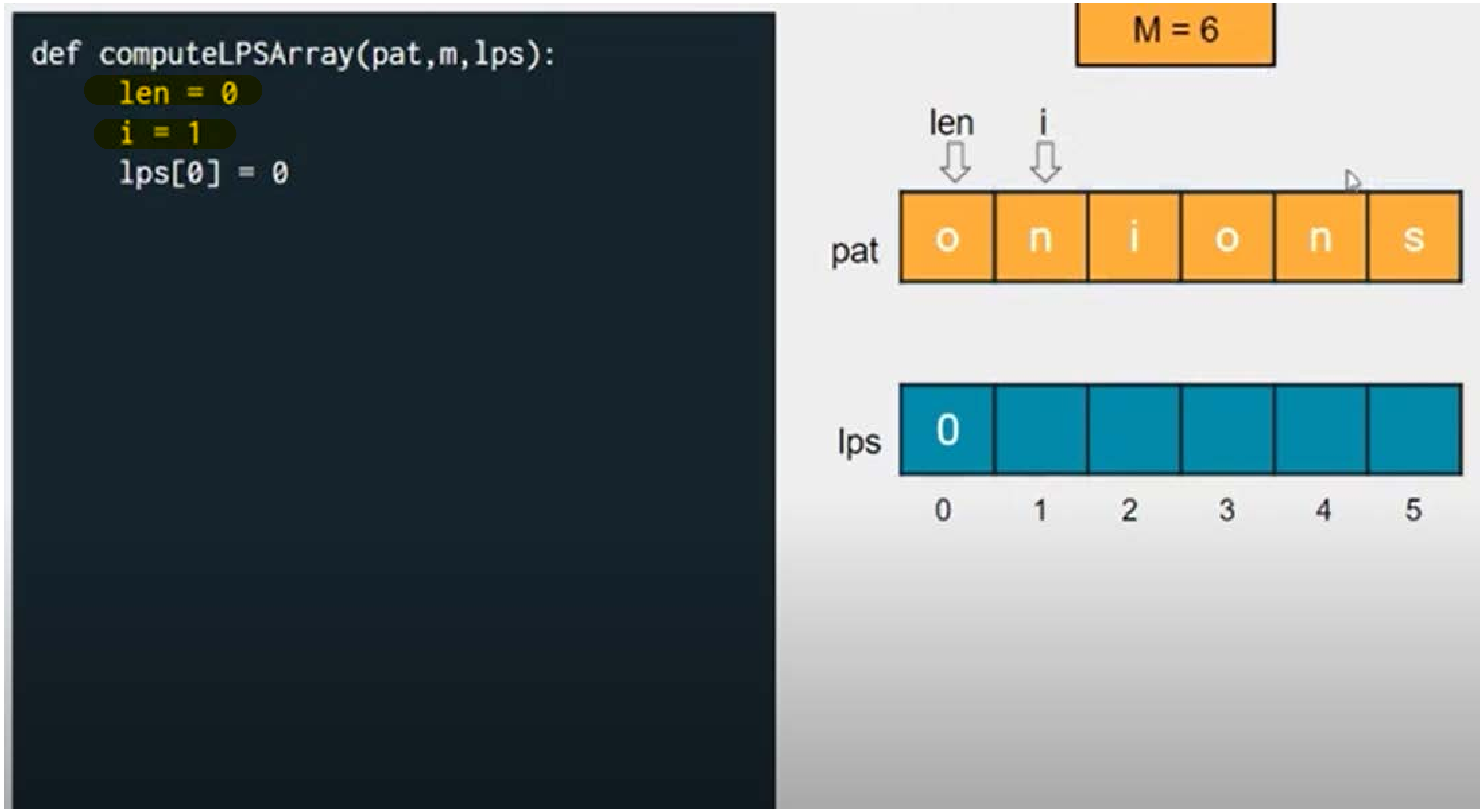
تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
```

https://www.youtube.com/watch?reload=9&v=4jY57Ehc14Y

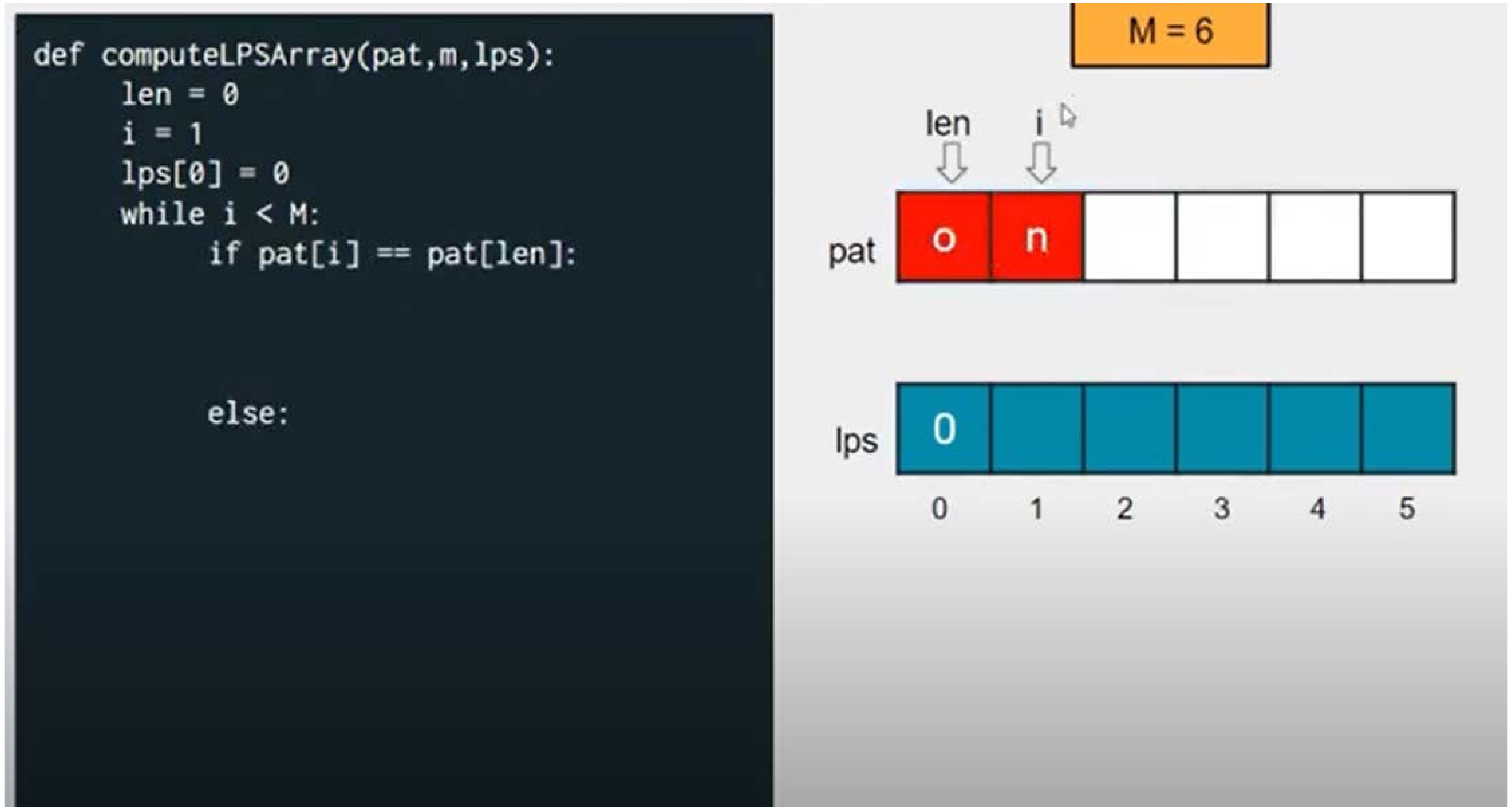
تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
```

https://www.youtube.com/watch?reload=9&v=4jY57Ehc14Y

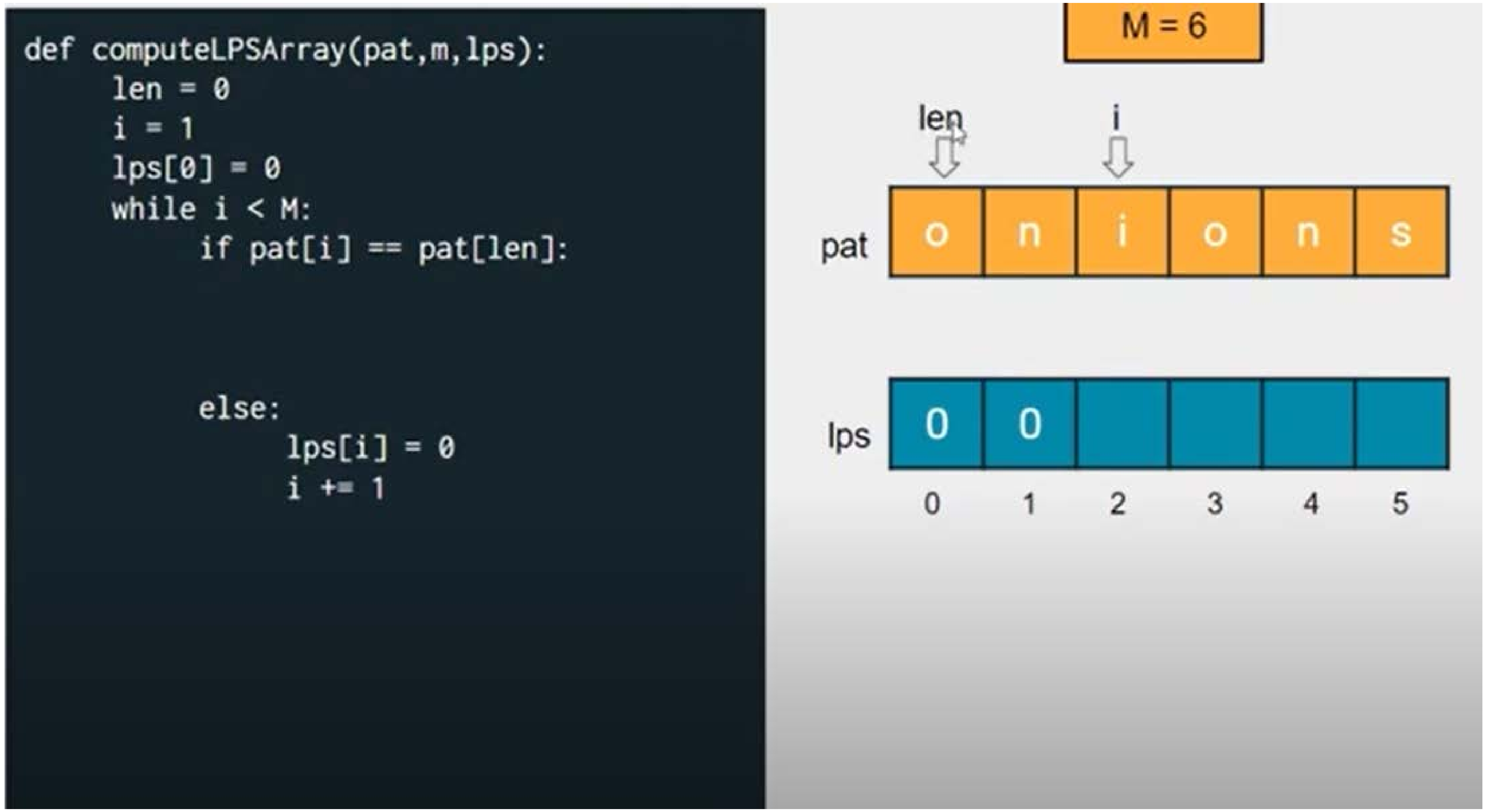
تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
        len = 0
        i = 1
        lps[0] = 0
        while i < M:
                if pat[i] == pat[len]:



                else:
```

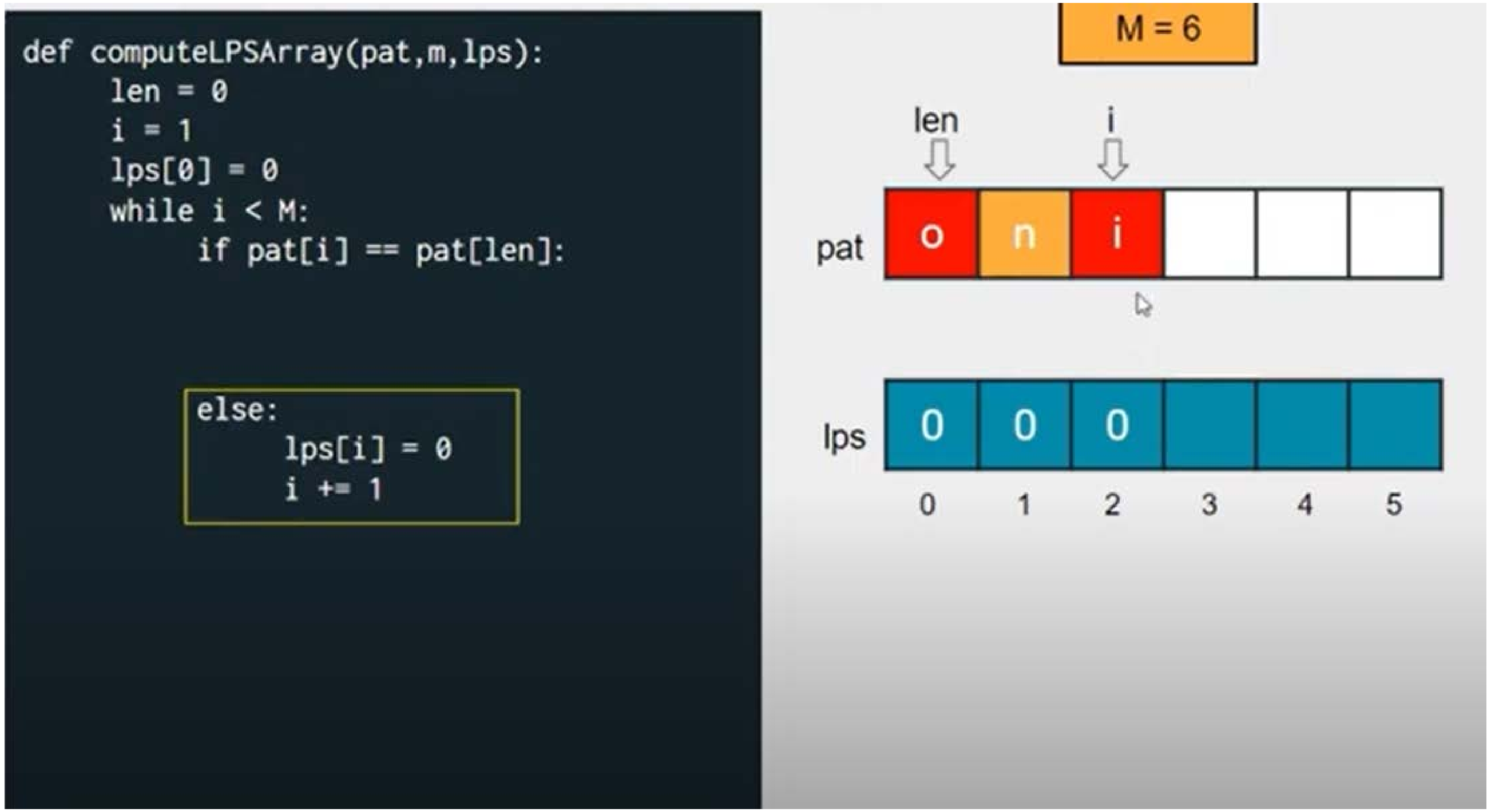
https://www.youtube.com/watch?reload=9&v=4jY57Ehc14Y

تابع شکست در الگوریتم KMP



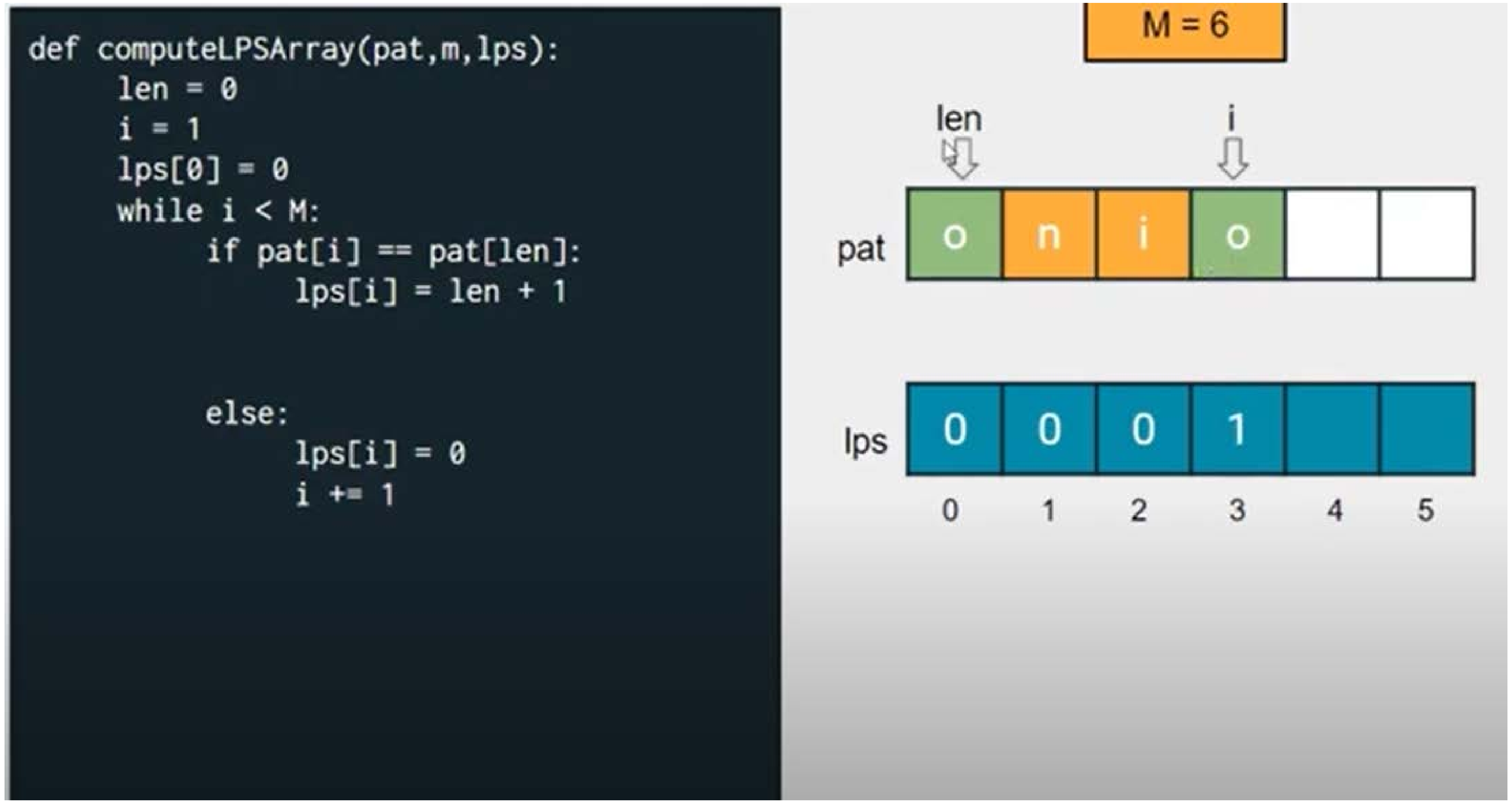
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:



        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



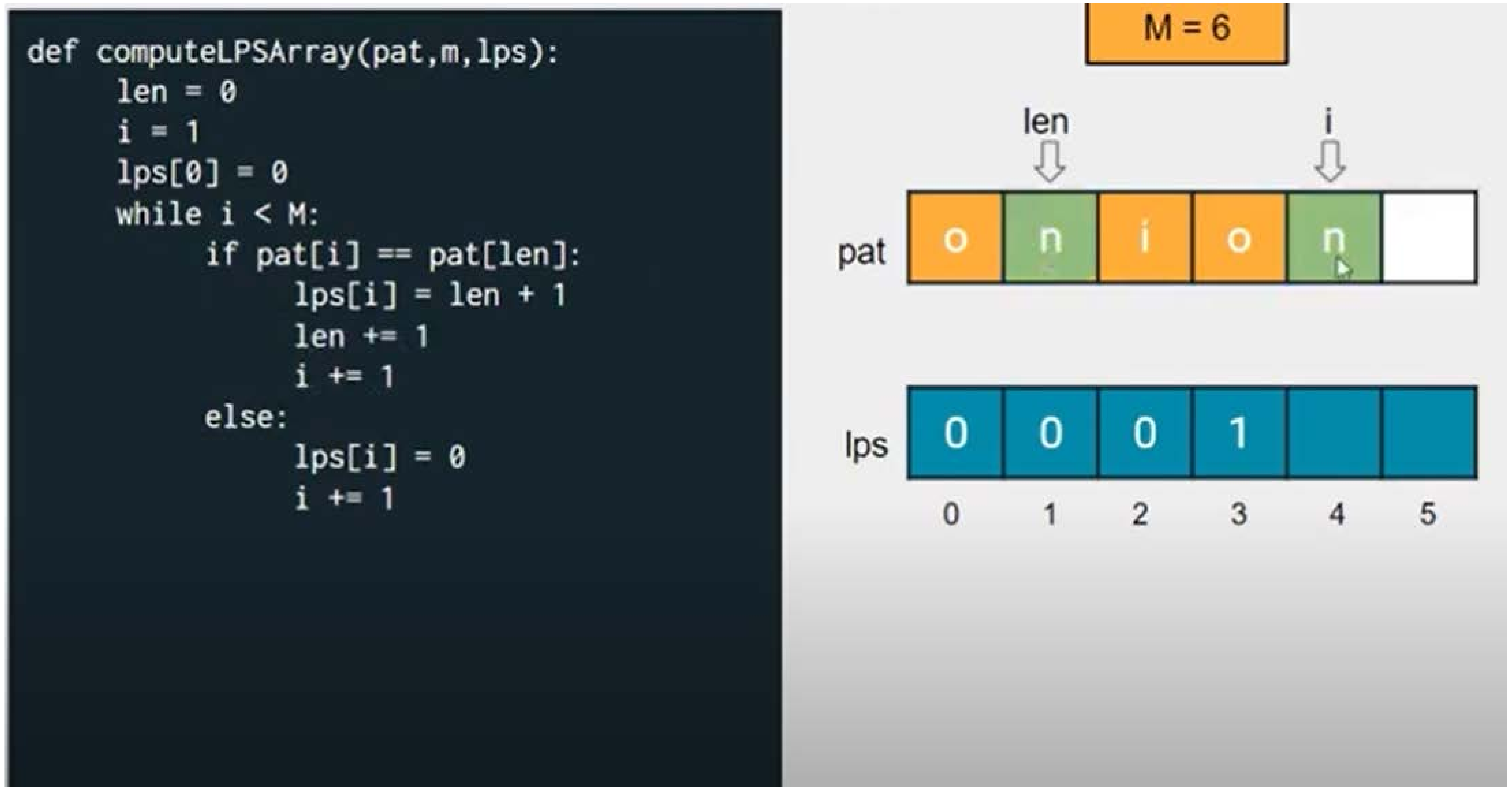
```
def computeLPSArray(pat,m,lps):
        len = 0
        i = 1
        lps[0] = 0
        while i < M:
                if pat[i] == pat[len]:



                else:
                        lps[i] = 0
                        i += 1
```

تابع شکست در الگوریتم KMP



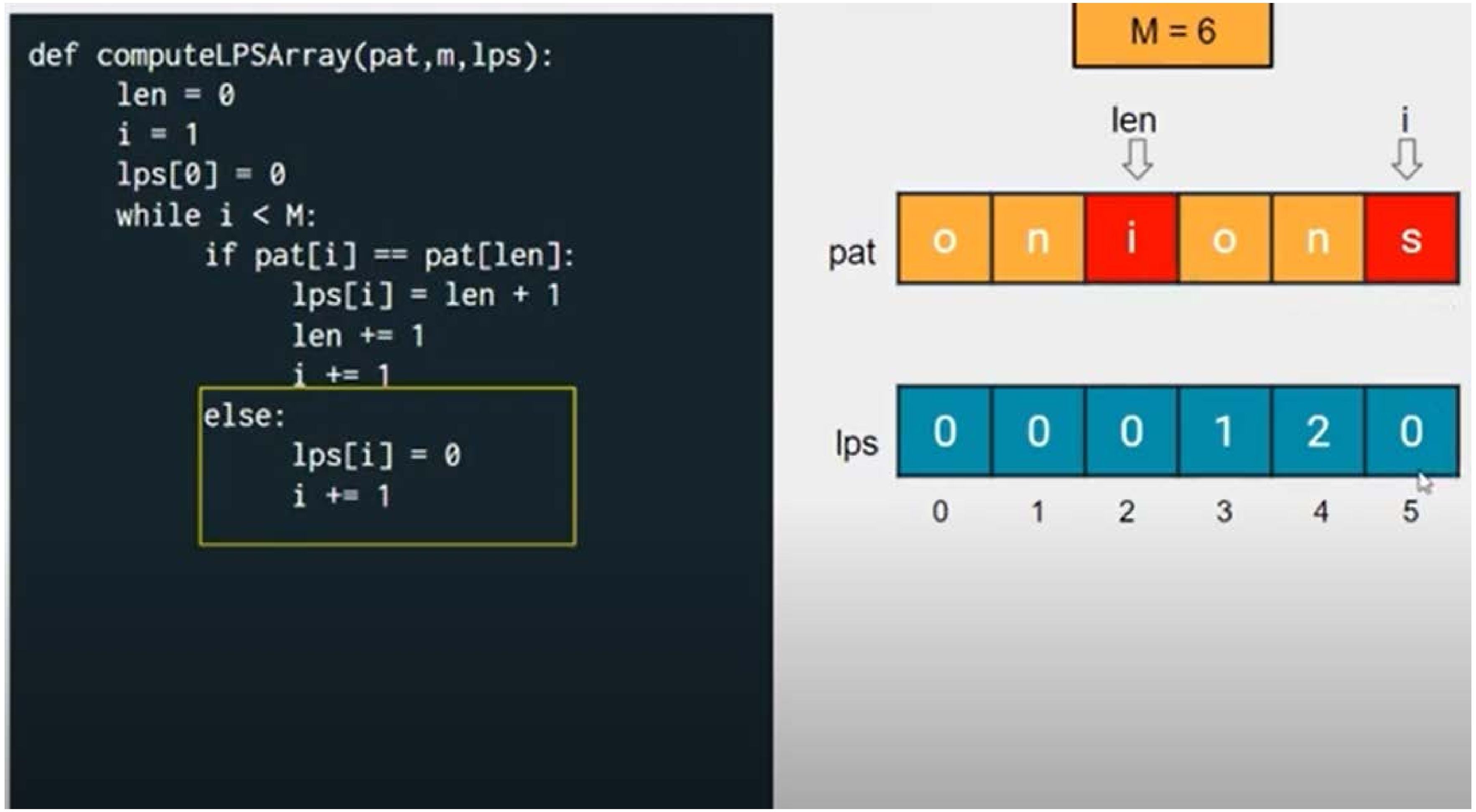
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1


        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



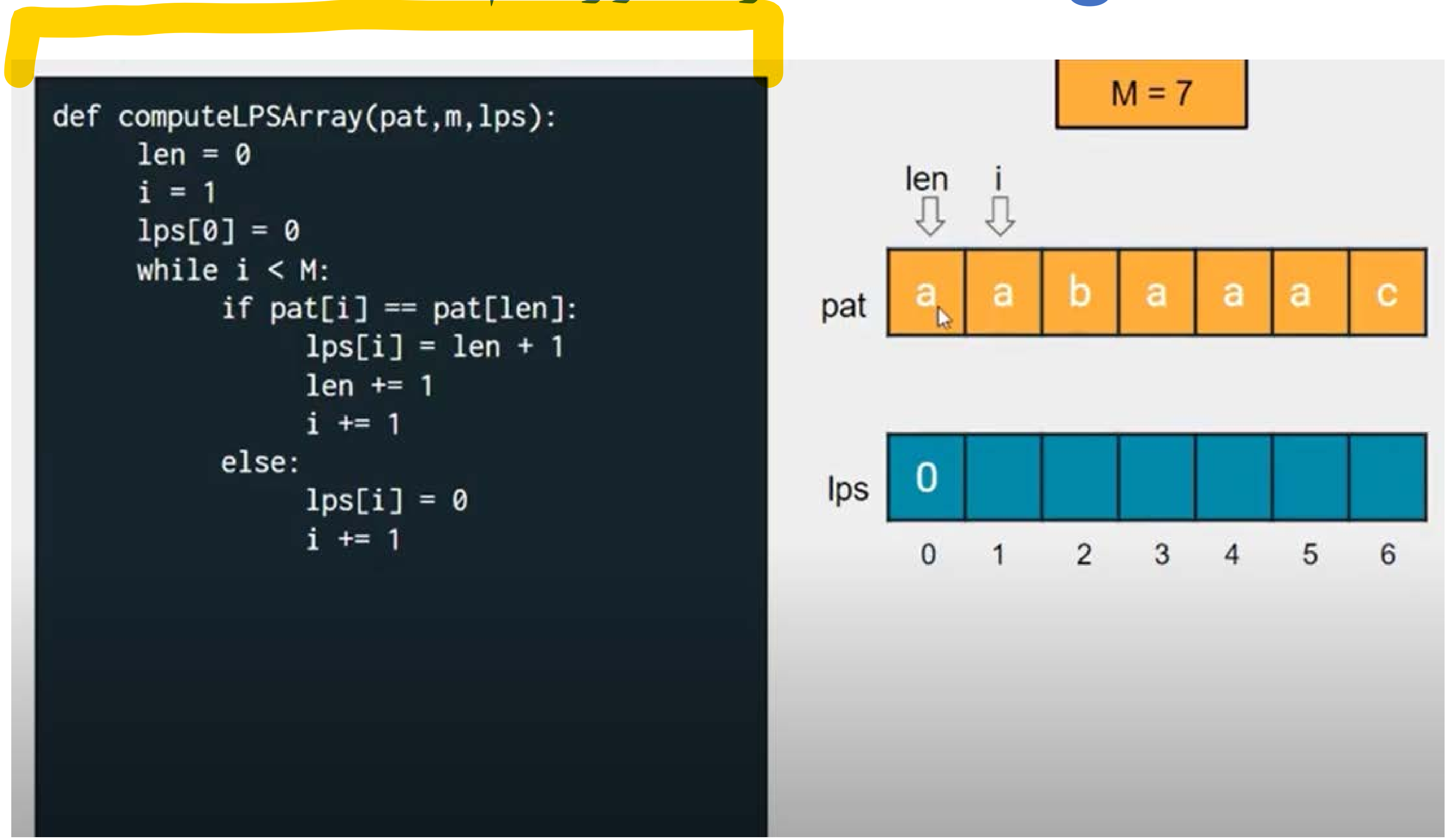
```
def computeLPSArray(pat,m,lps):
        len = 0
        i = 1
        lps[0] = 0
        while i < M:
                if pat[i] == pat[len]:
                        lps[i] = len + 1
                        len += 1
                        i += 1
                else:
                        lps[i] = 0
                        i += 1
```

تابع شکست در الگوریتم KMP



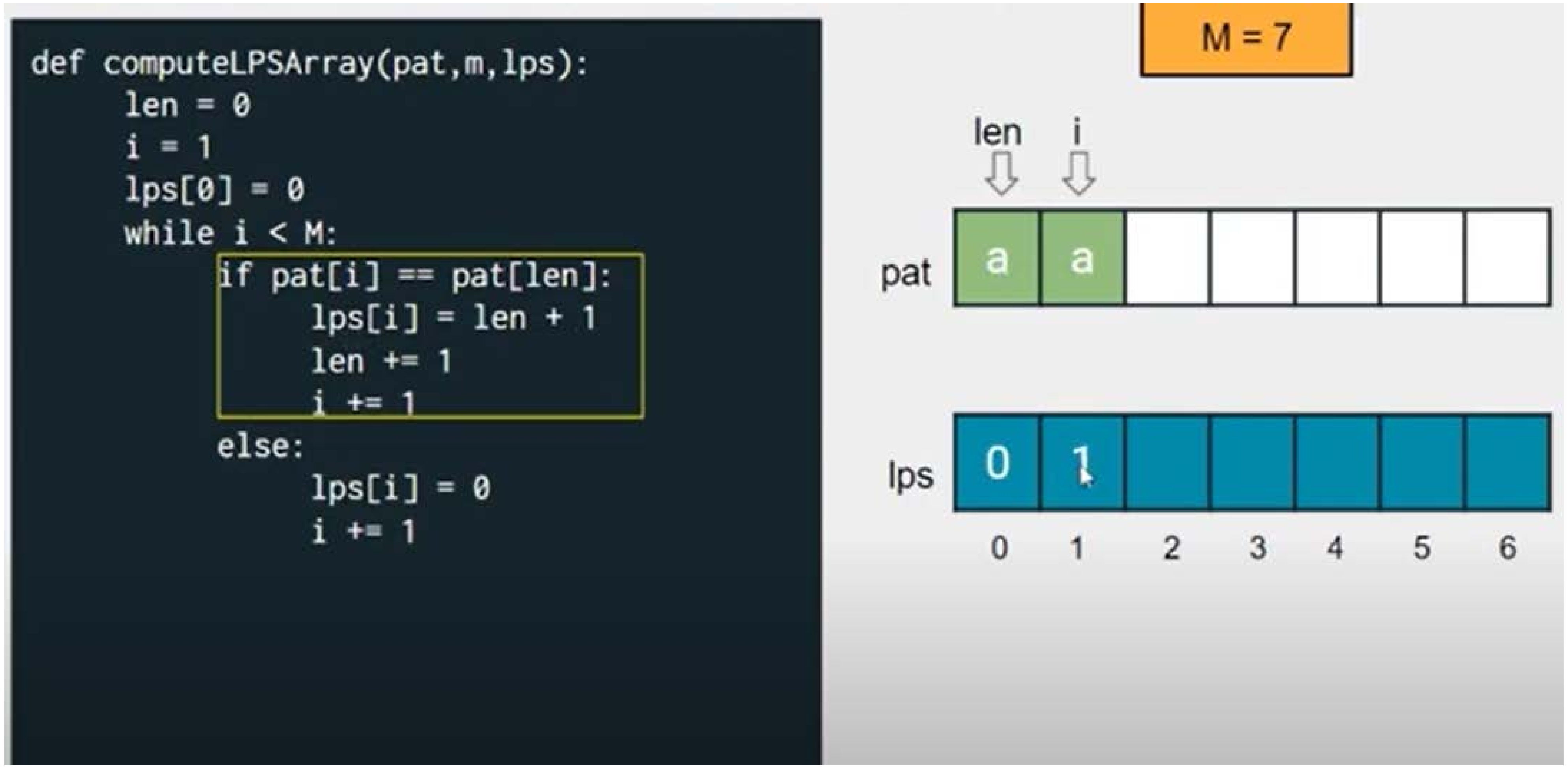
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



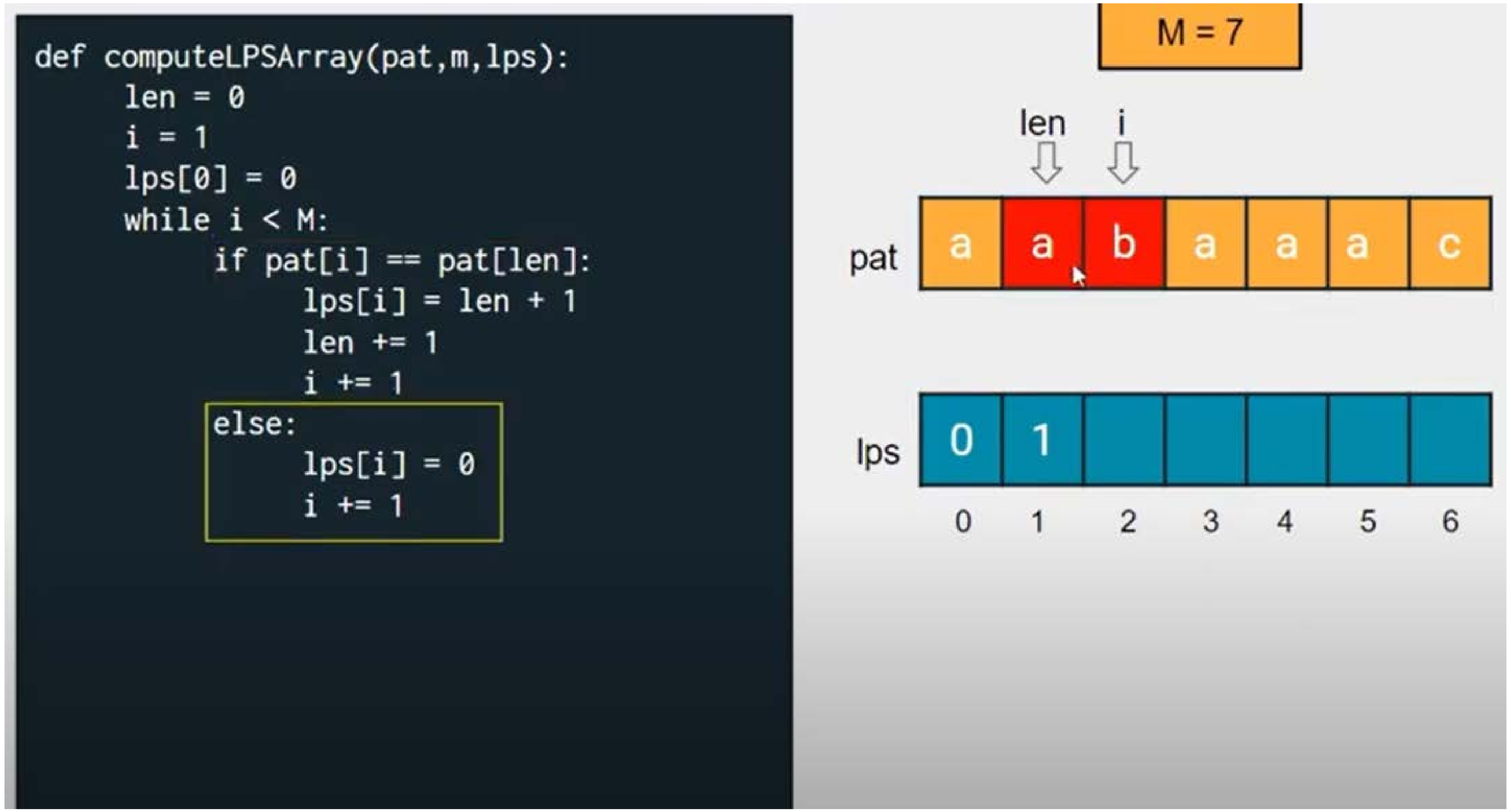
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



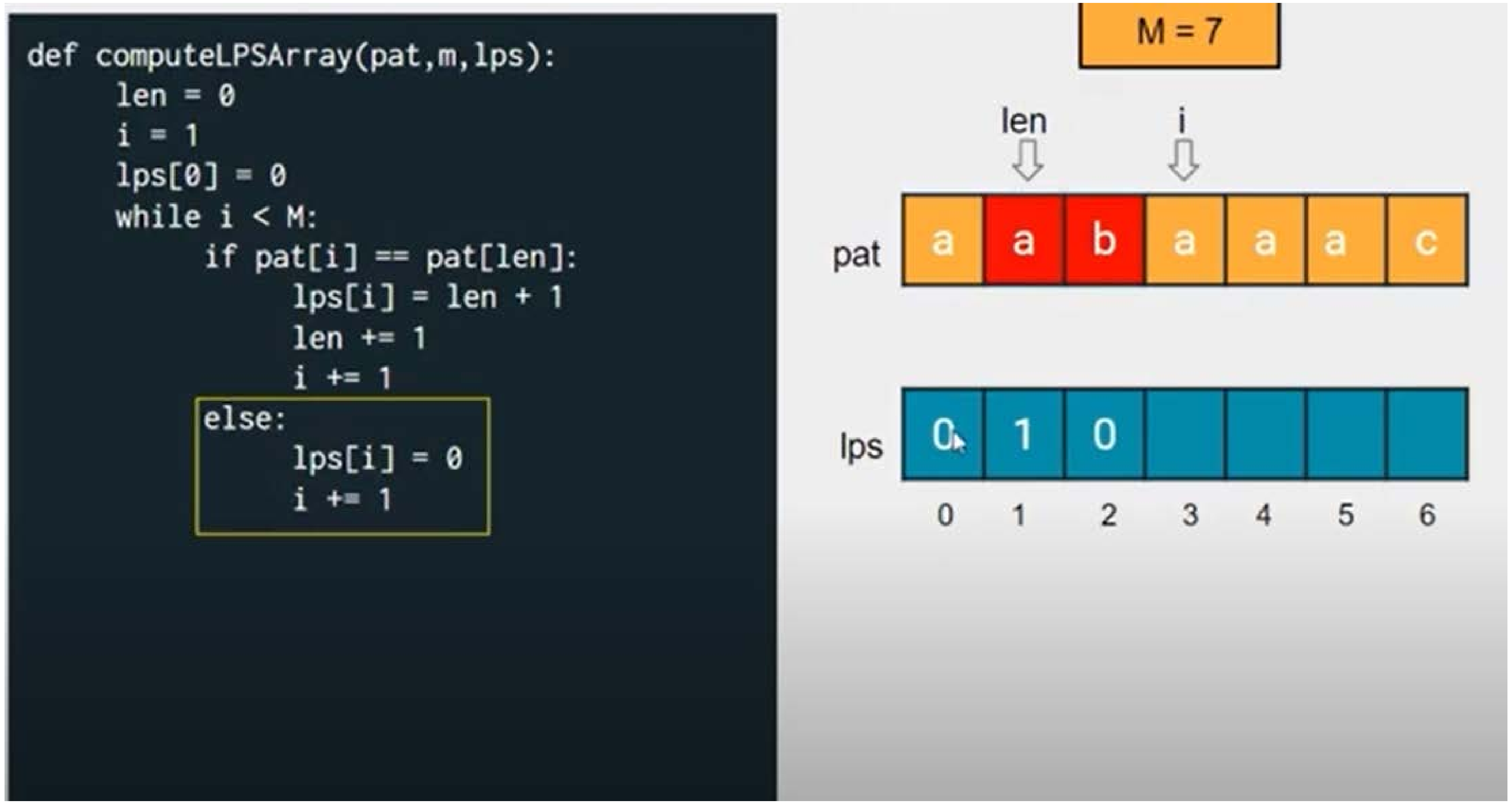
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



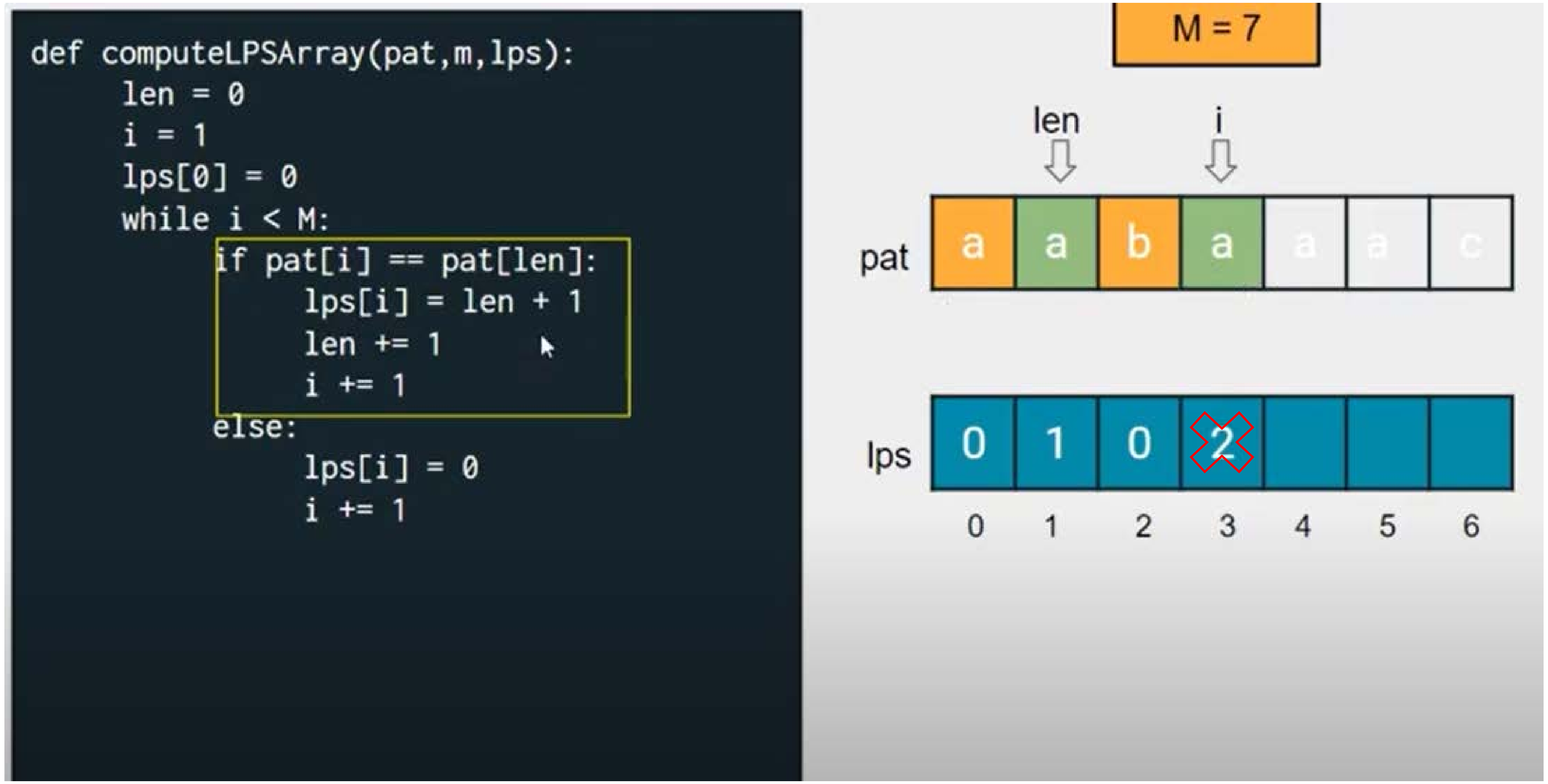
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



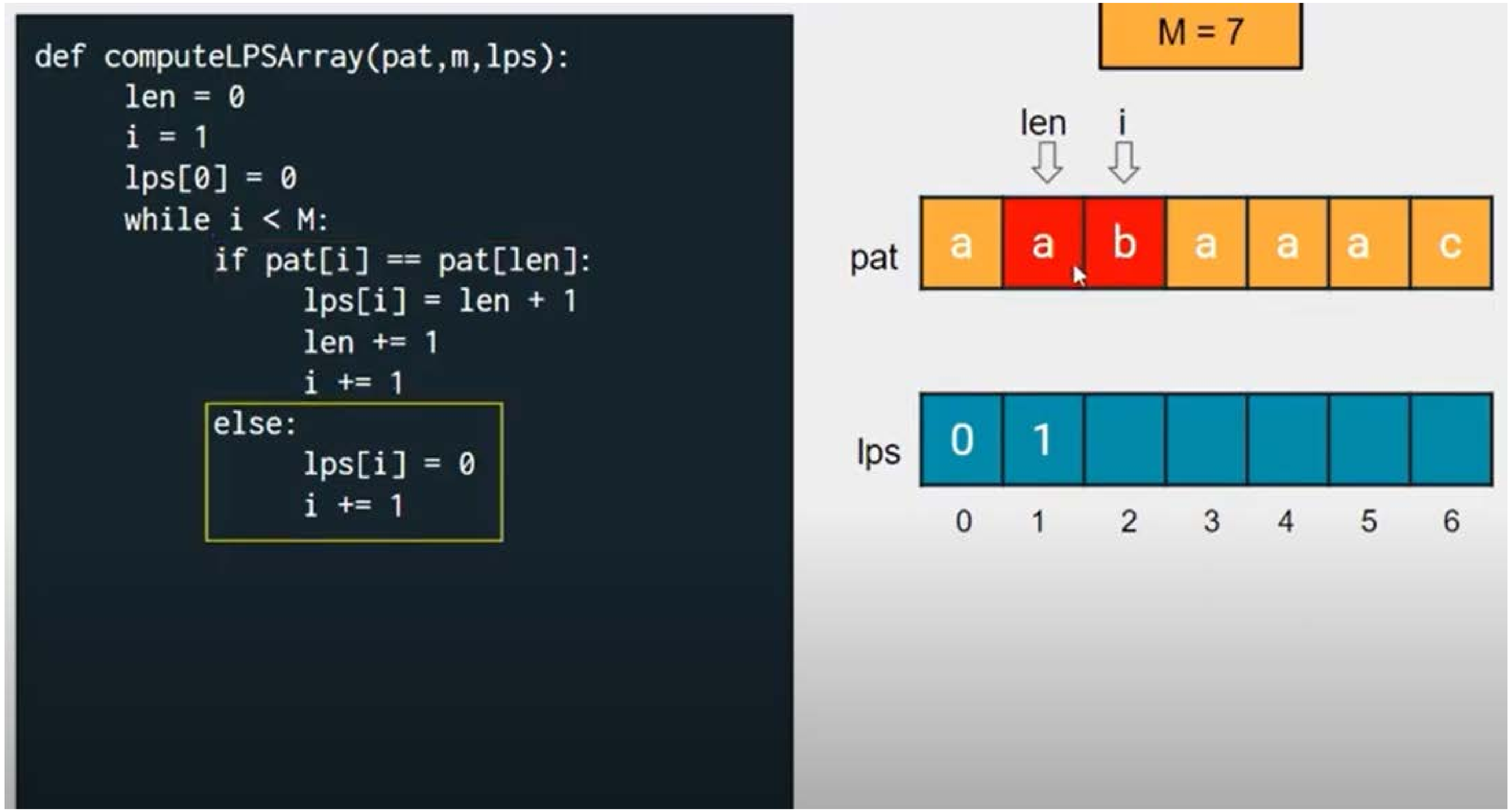
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



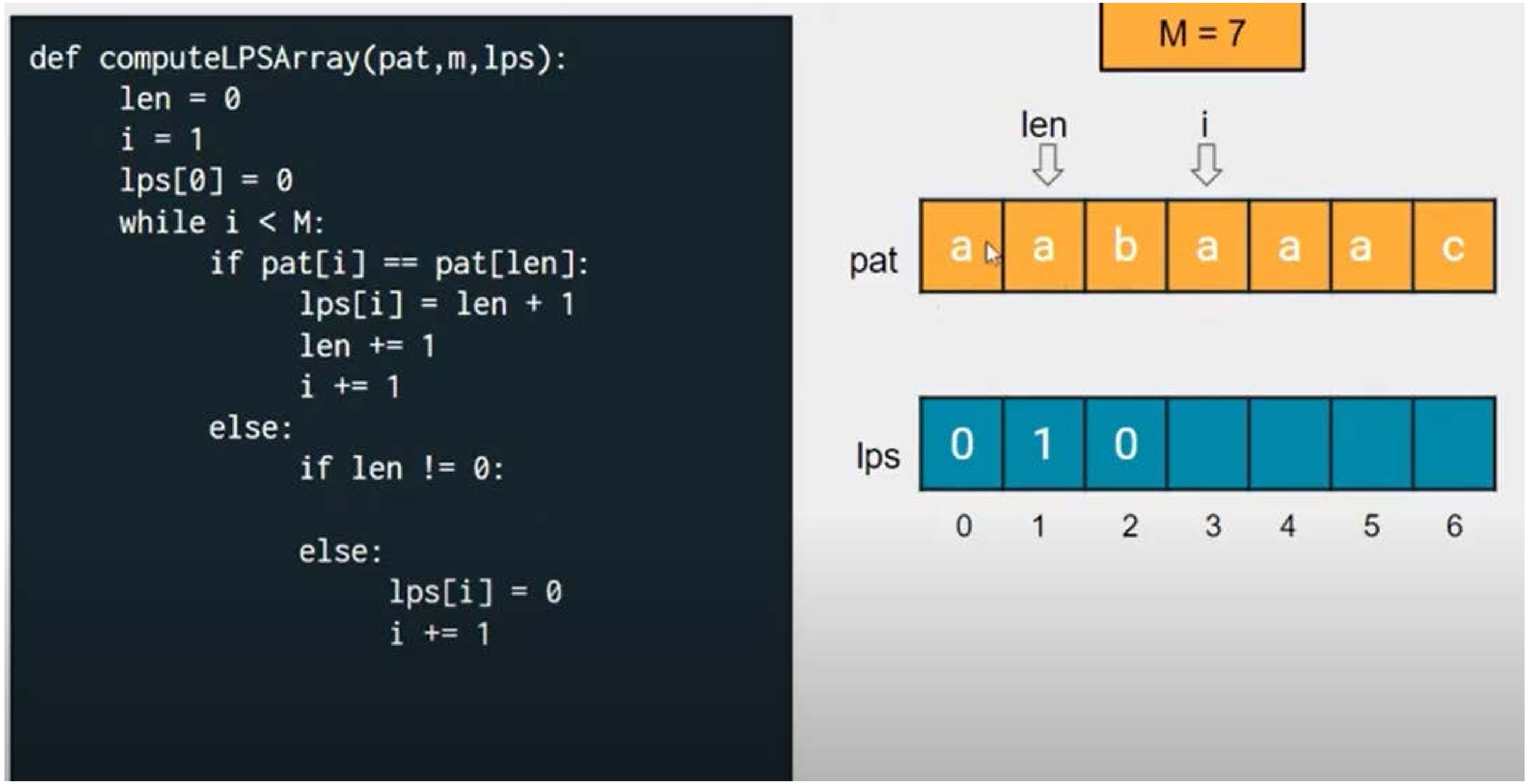
```
def computeLPSArray(pat,m,lps):
        len = 0
        i = 1
        lps[0] = 0
        while i < M:
                if pat[i] == pat[len]:
                        lps[i] = len + 1
                        len += 1
                        i += 1
                else:
                        lps[i] = 0
                        i += 1
```

تابع شکست در الگوریتم KMP



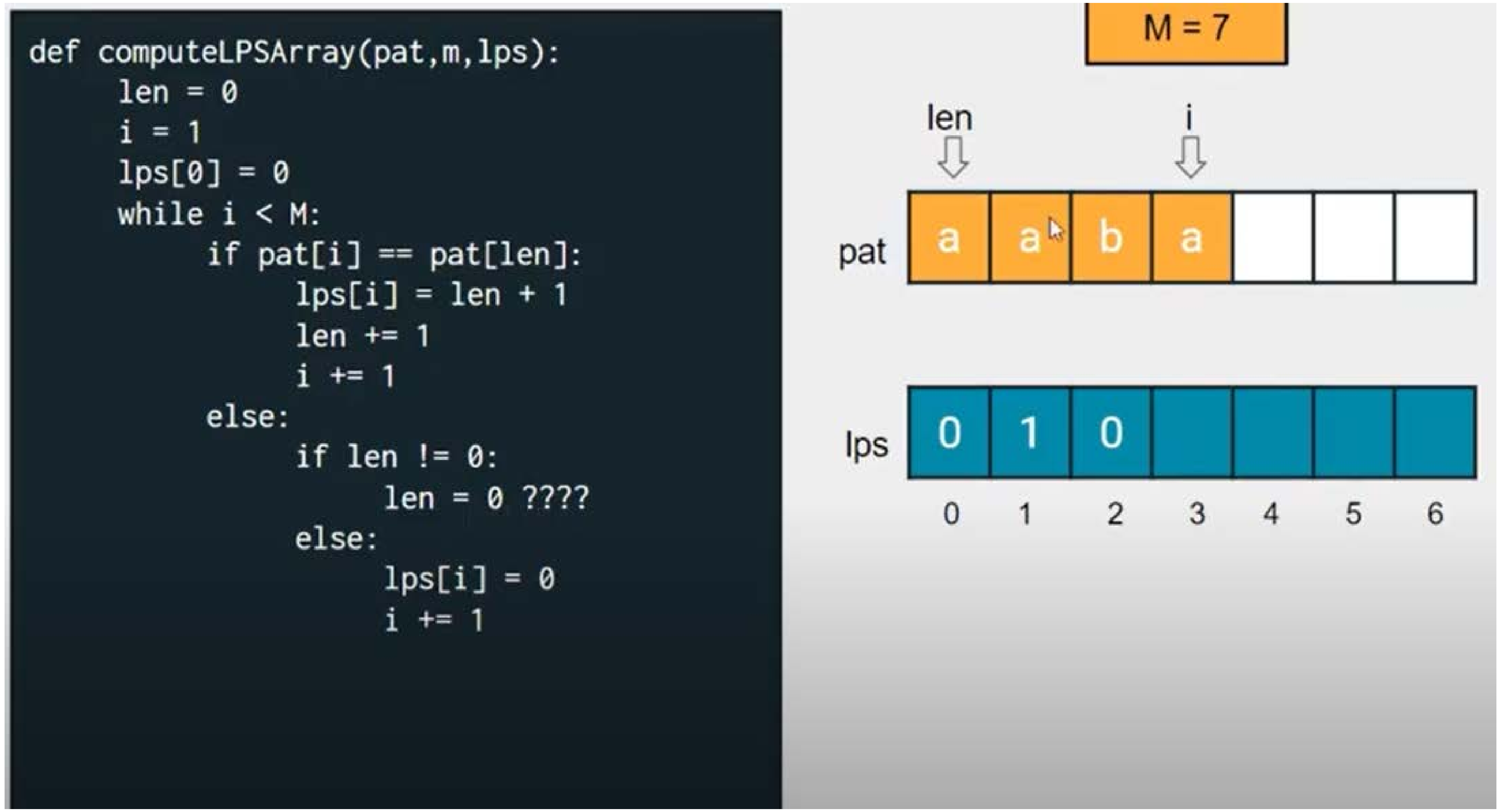
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            lps[i] = 0
            i += 1
```

تابع شکست در الگوریتم KMP



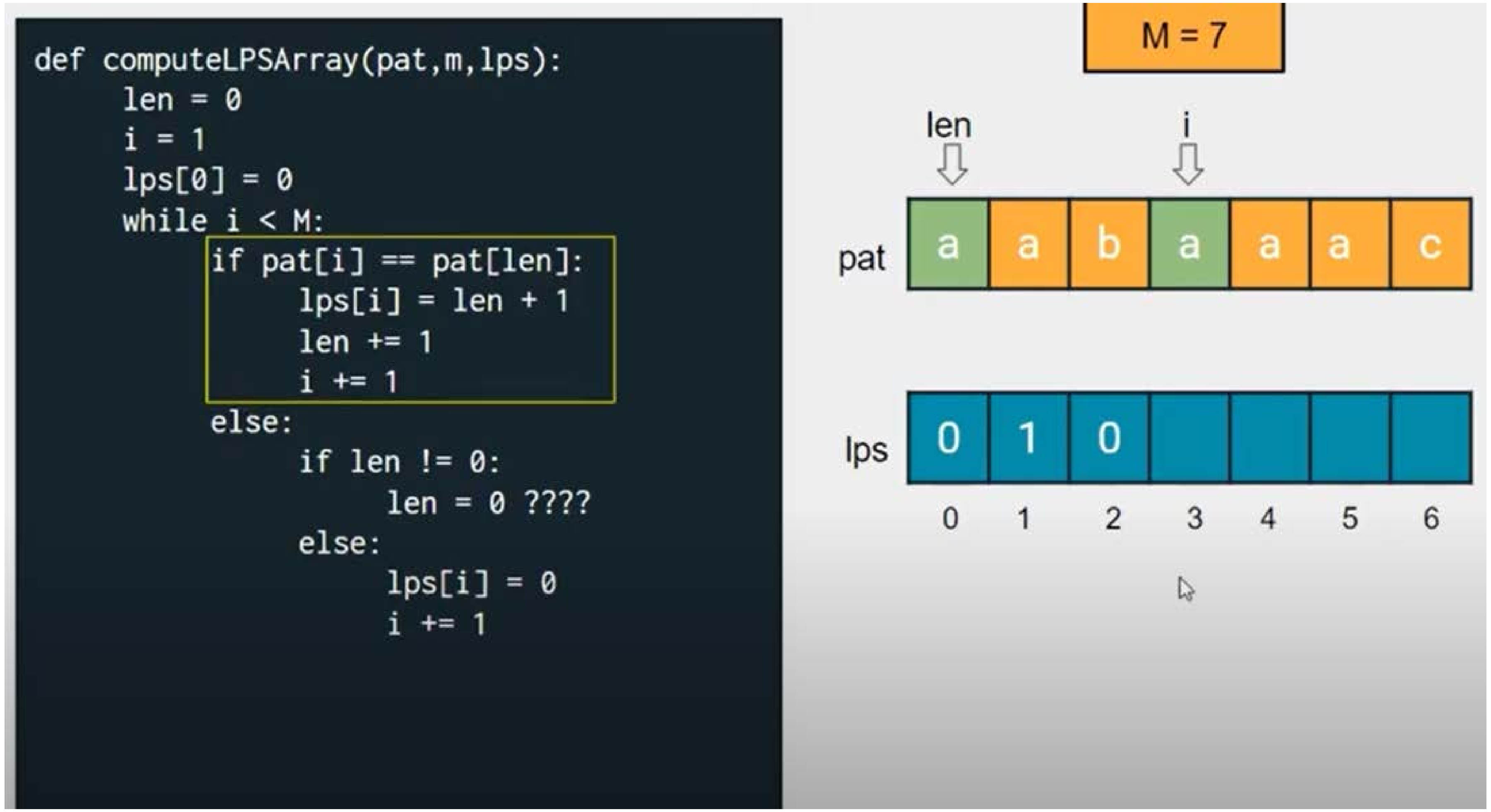
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:

            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



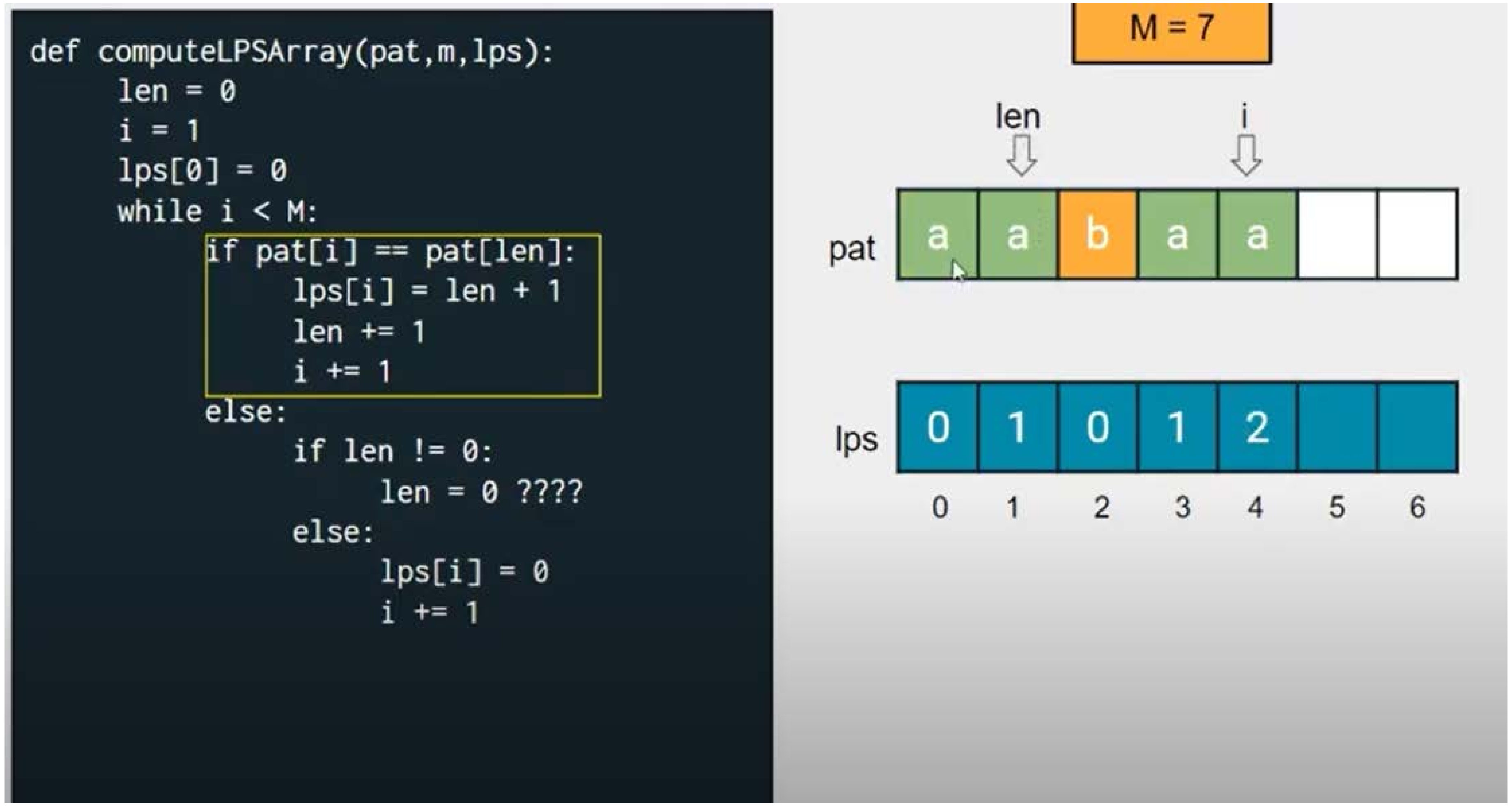
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



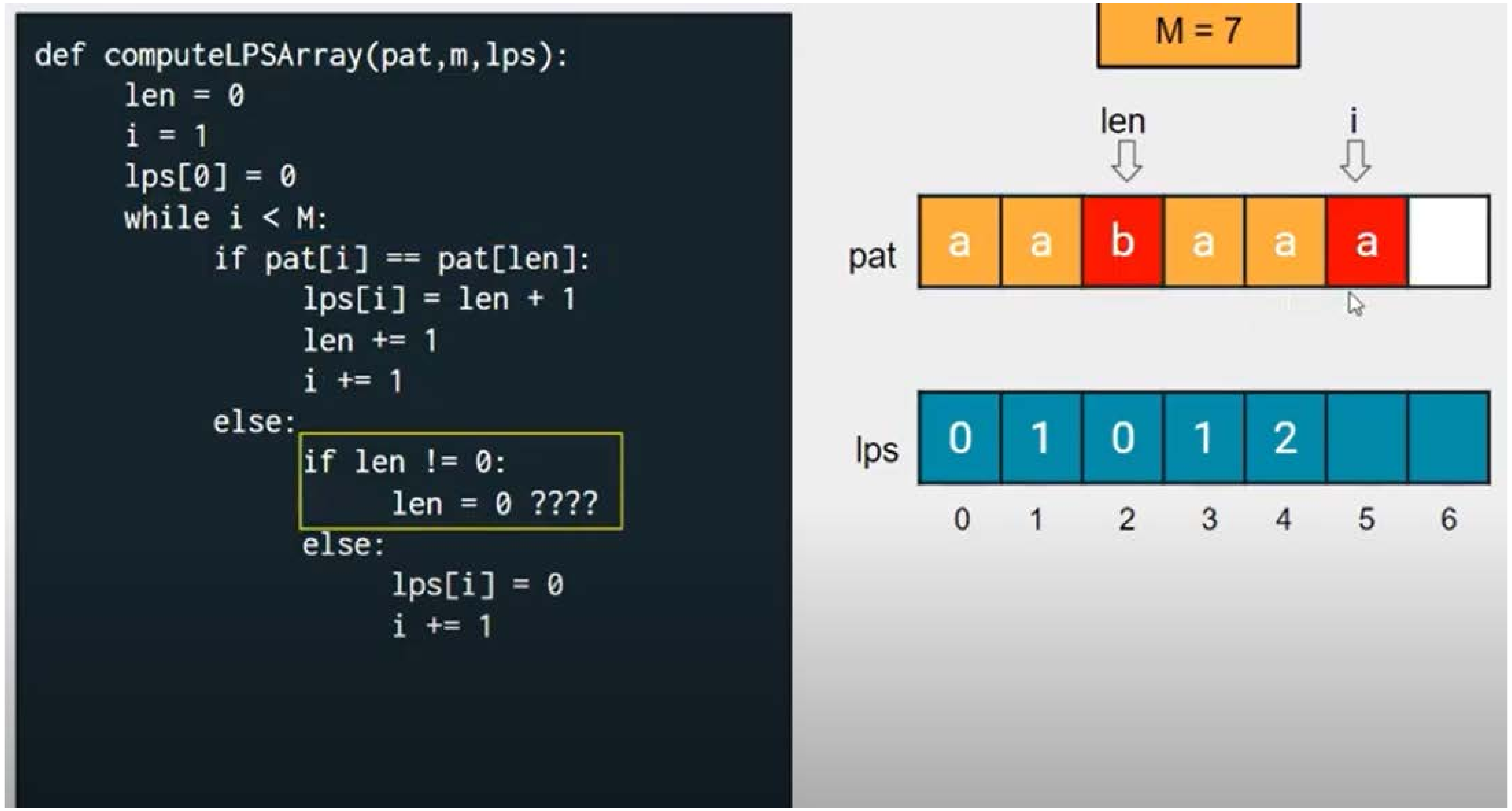
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



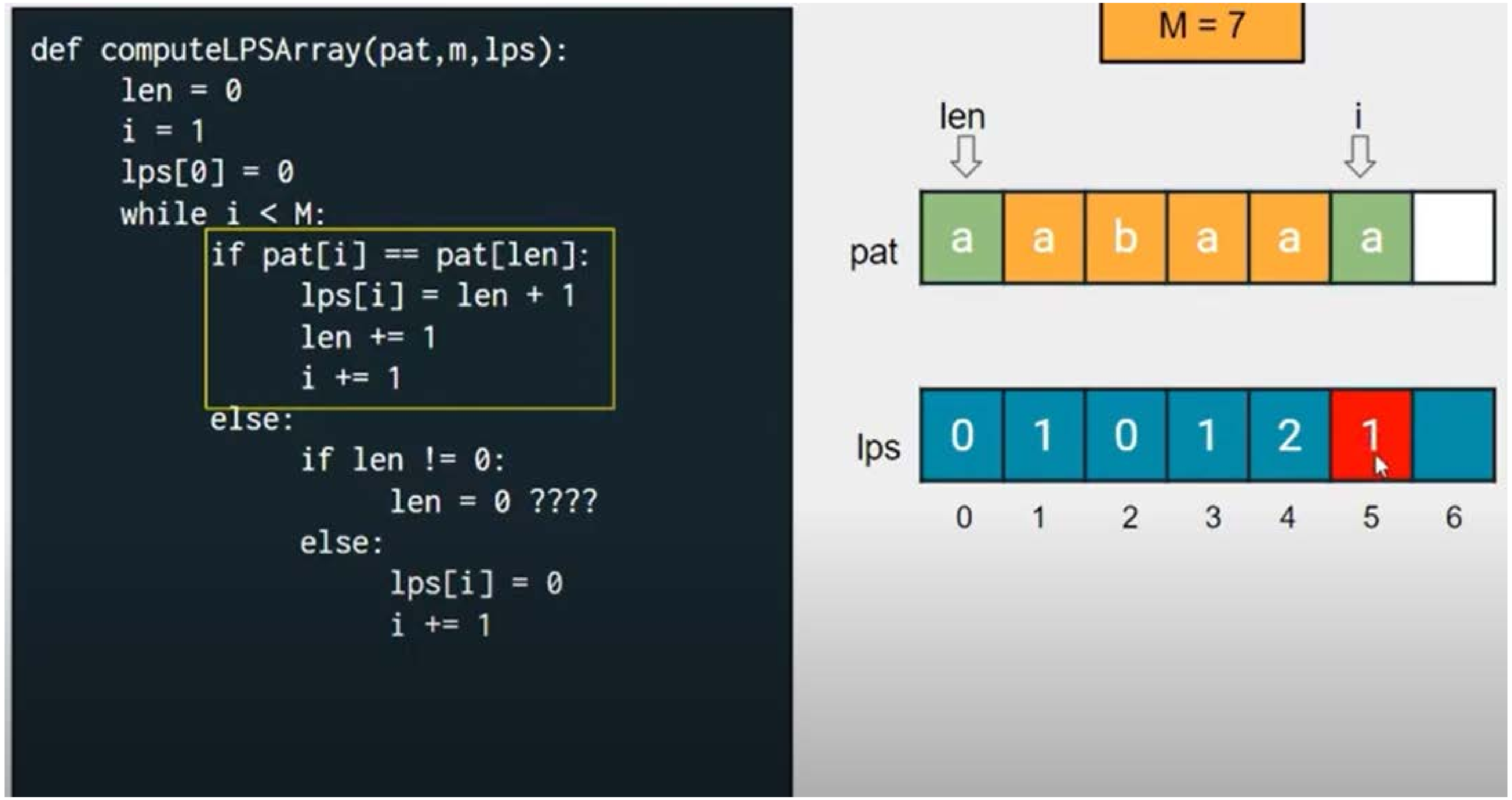
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



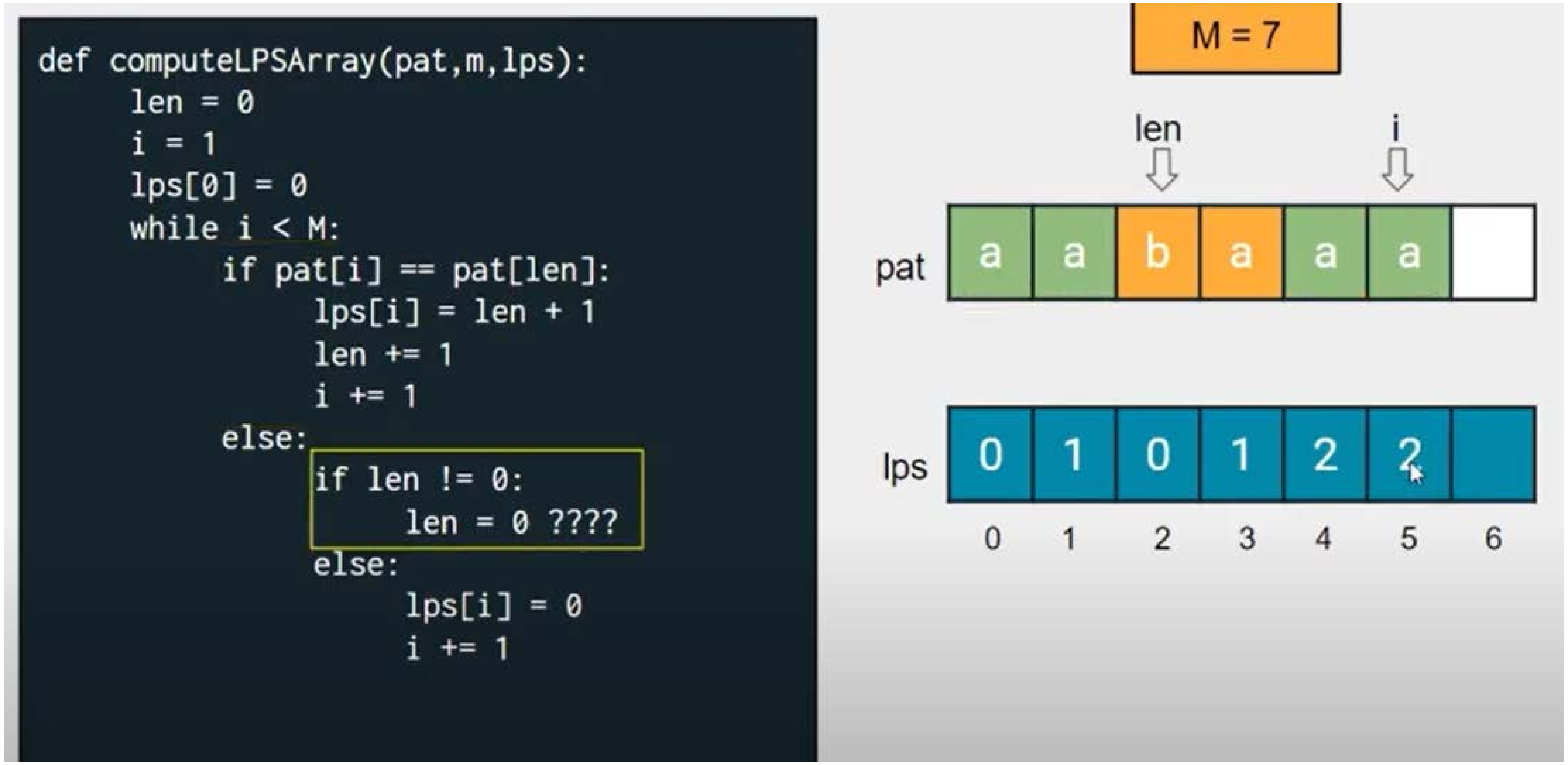
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



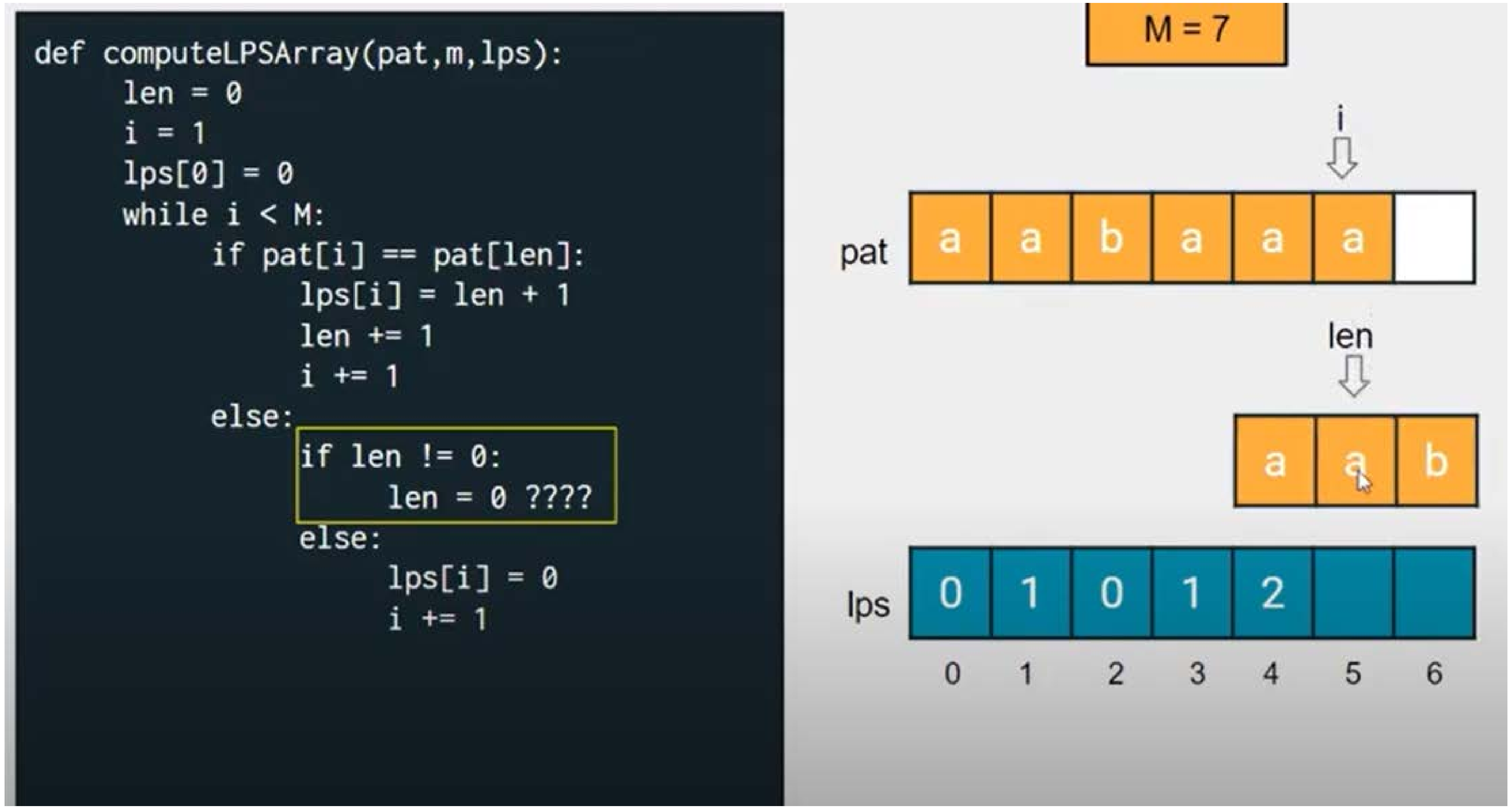
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



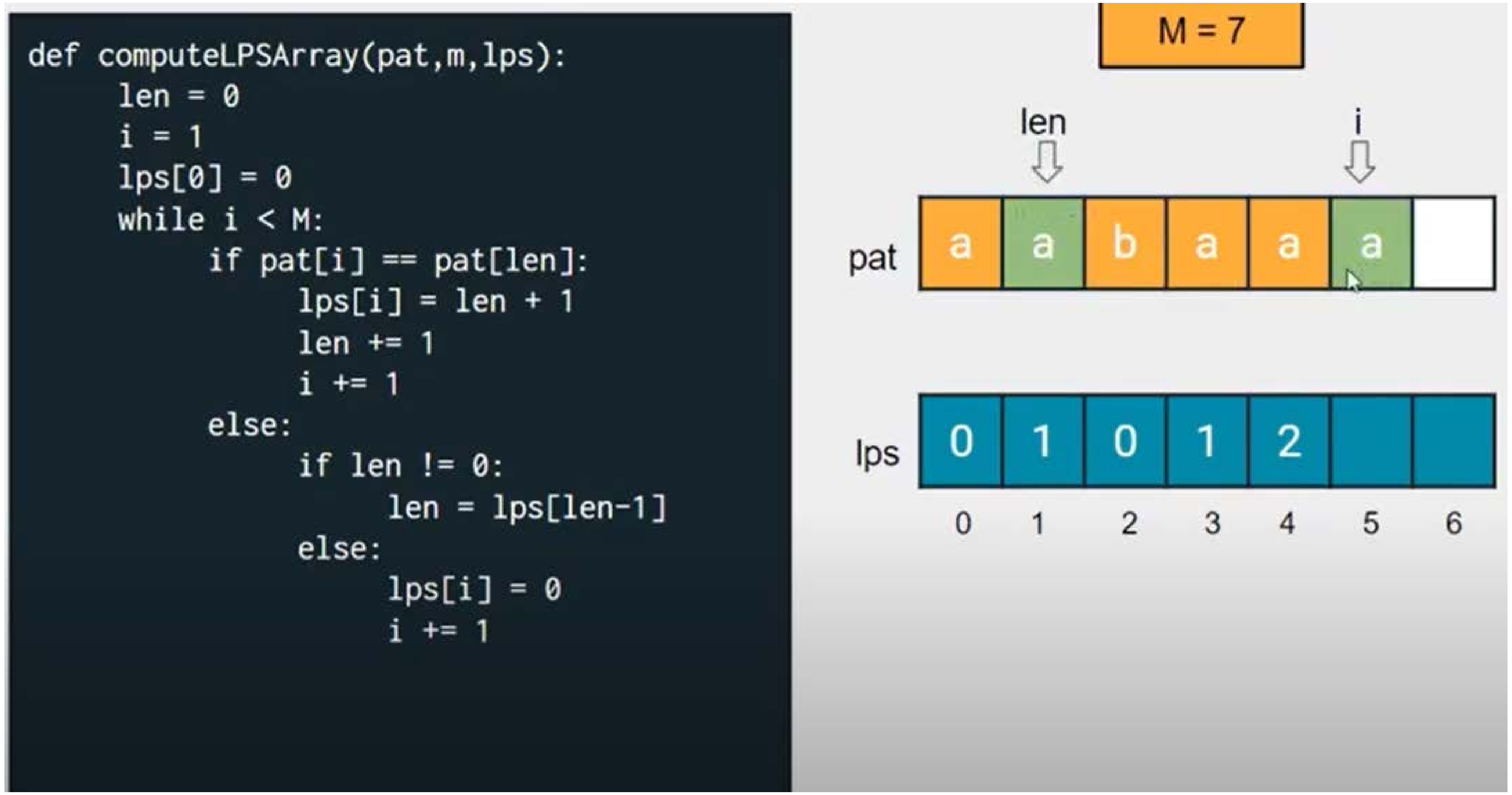
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



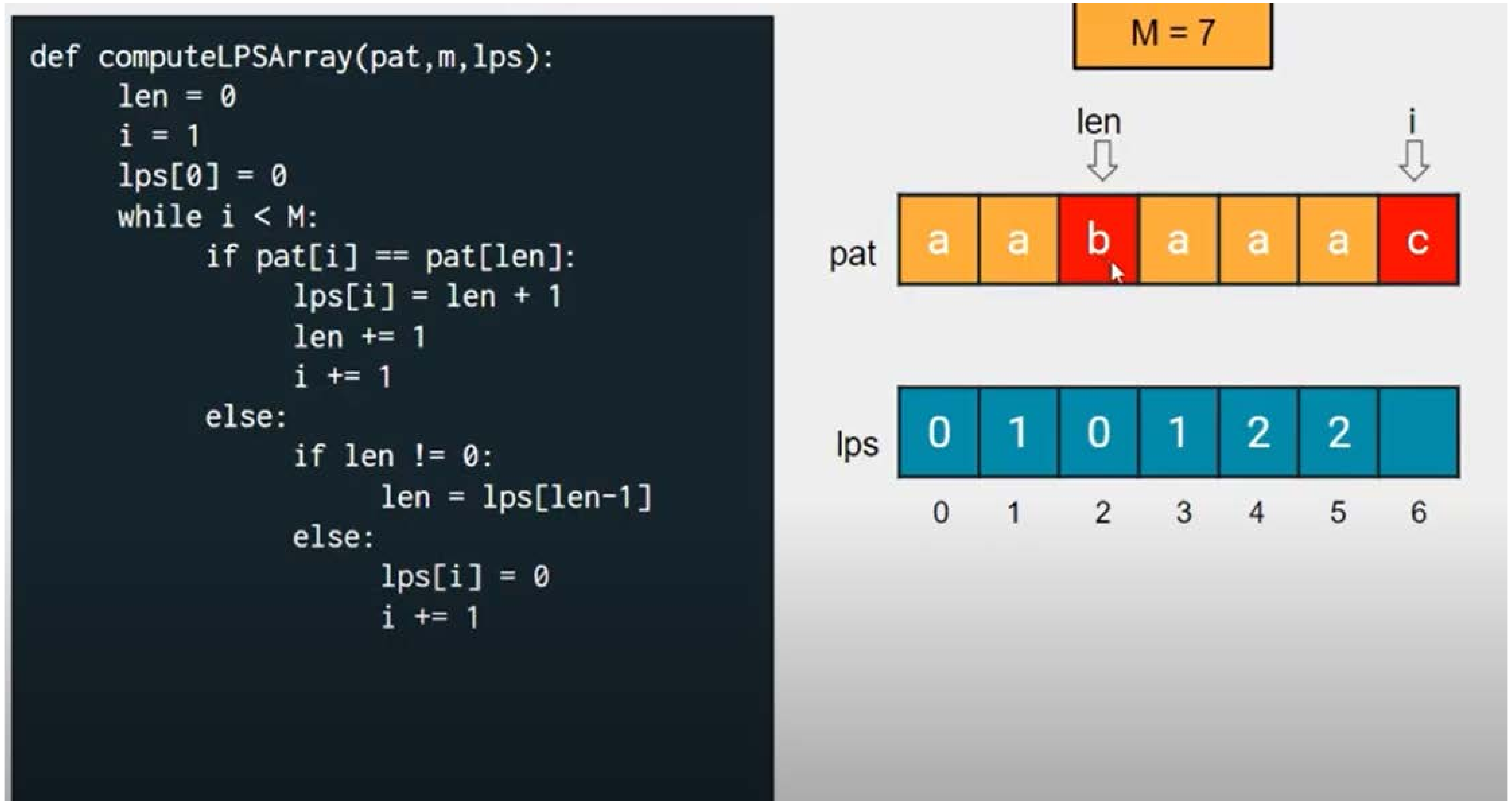
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = 0 ????
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



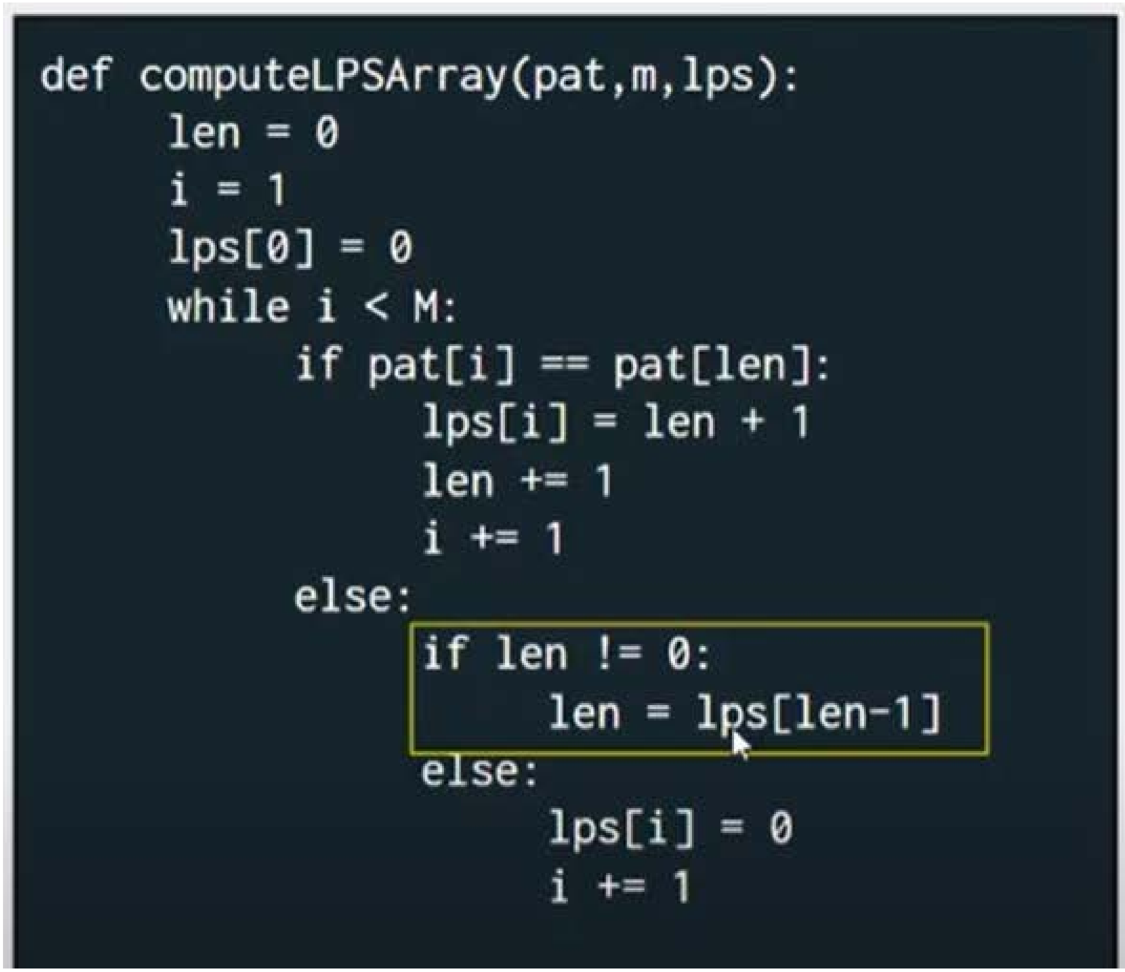
```
def computeLPSArray(pat,m,lps):
        len = 0
        i = 1
        lps[0] = 0
        while i < M:
                if pat[i] == pat[len]:
                        lps[i] = len + 1
                        len += 1
                        i += 1
                else:
                        if len != 0:
                                len = lps[len-1]
                        else:
                                lps[i] = 0
                                i += 1
```

تابع شکست در الگوریتم KMP

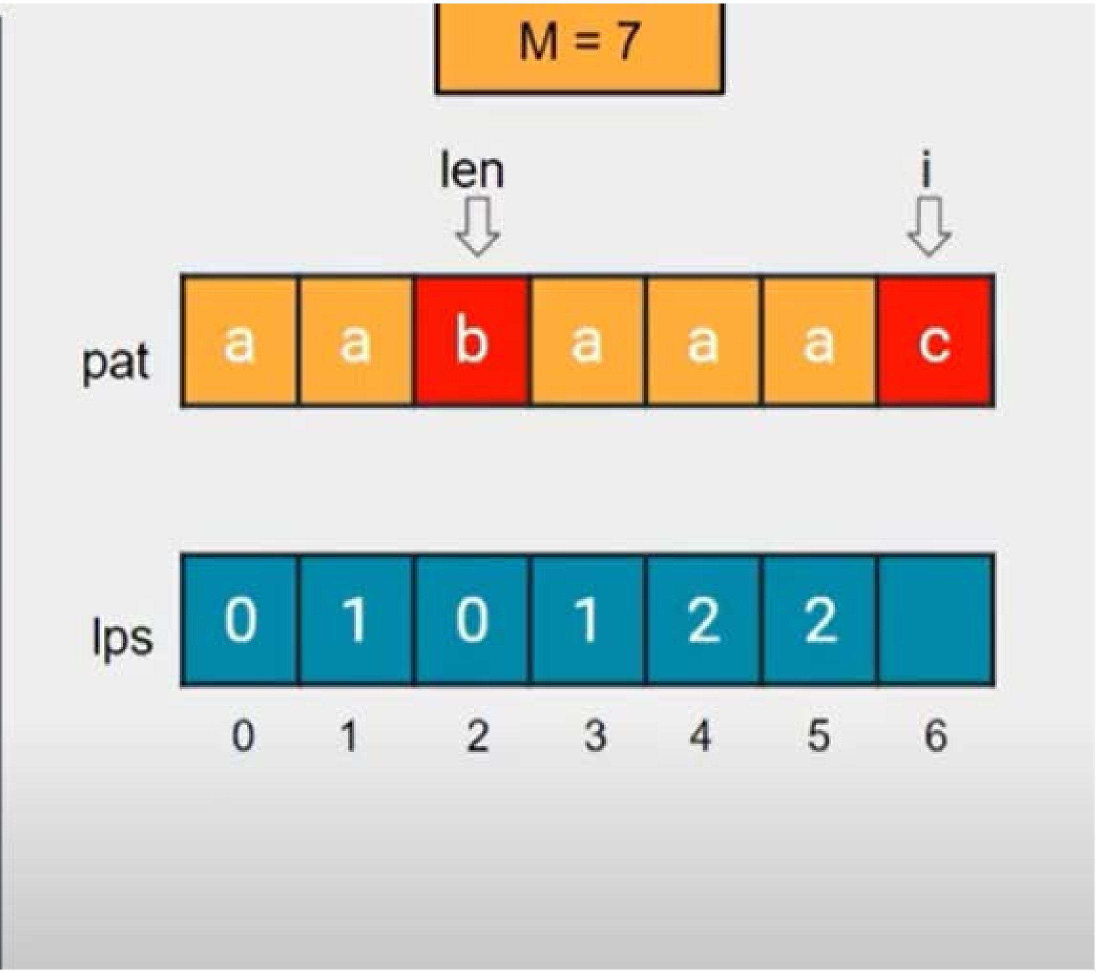


```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

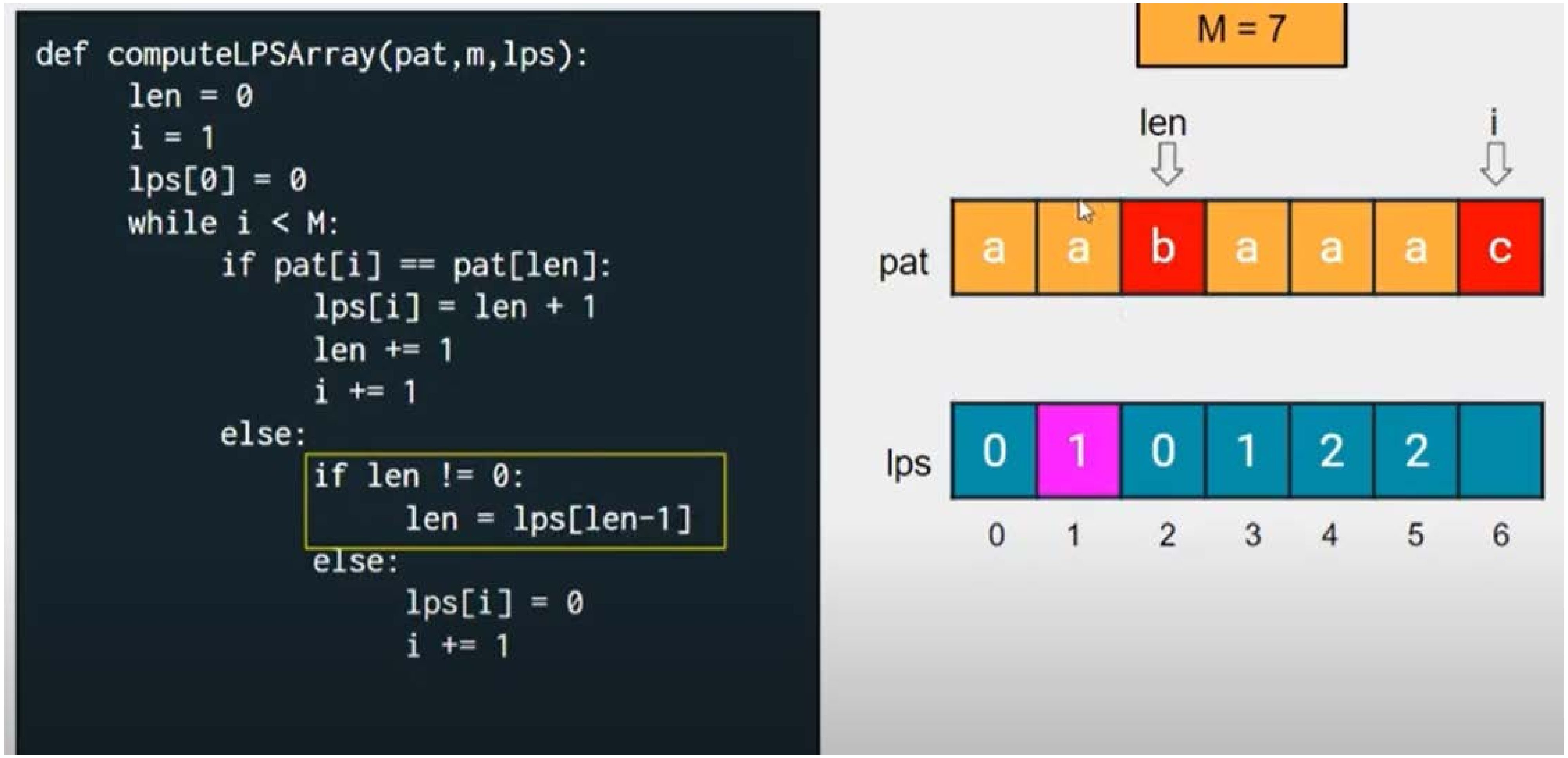
تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

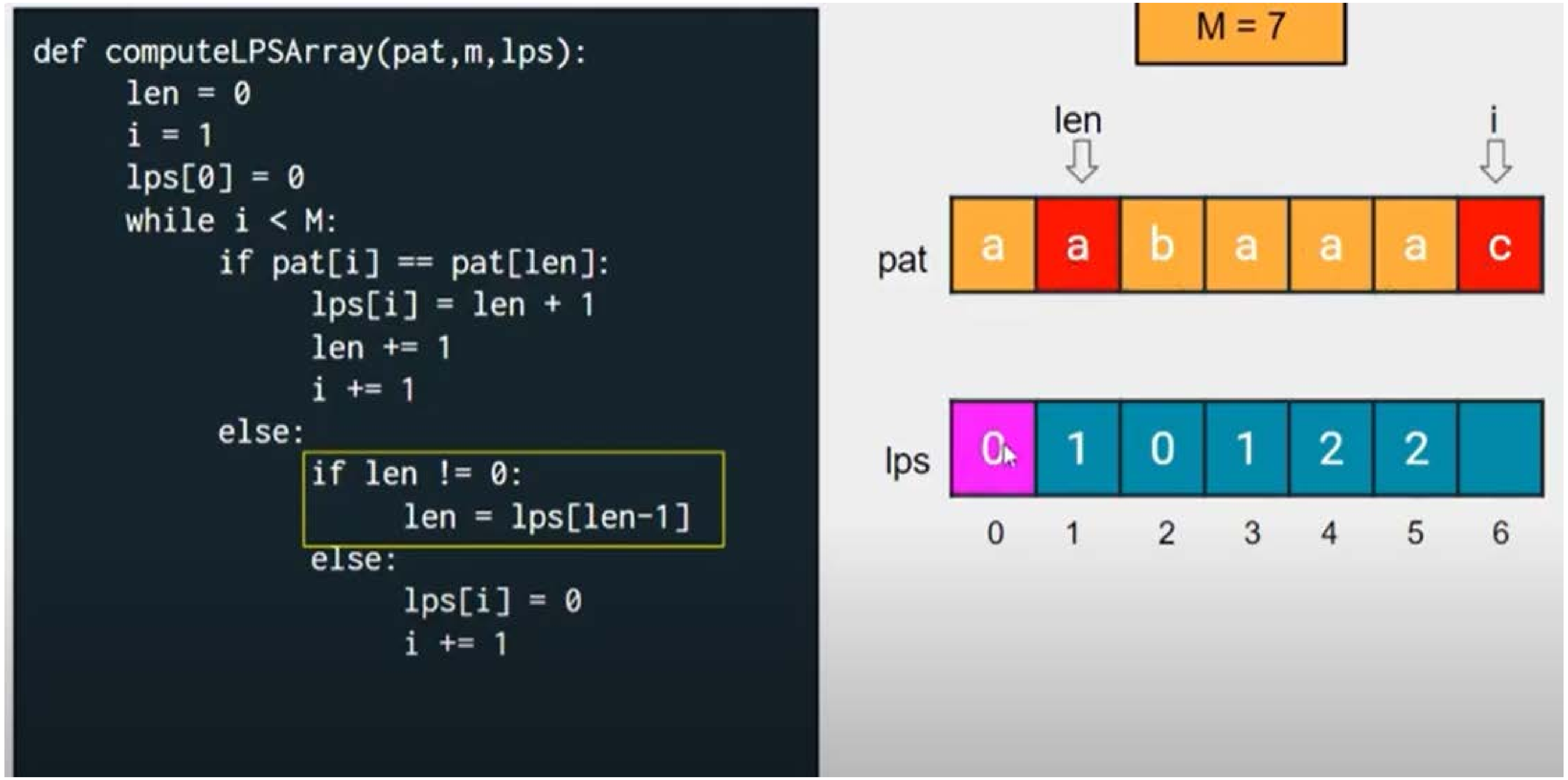


تابع شکست در الگوریتم KMP



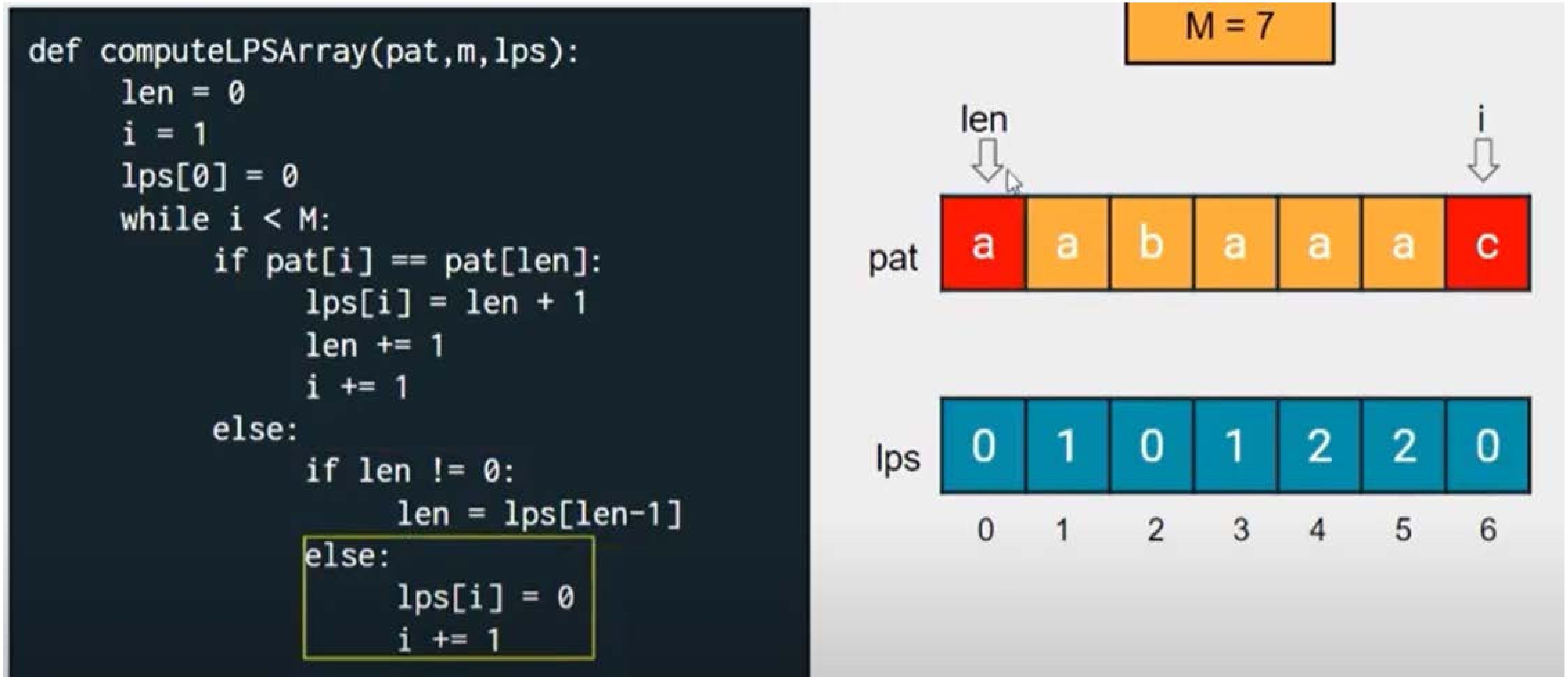
```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

تابع شکست در الگوریتم KMP



```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

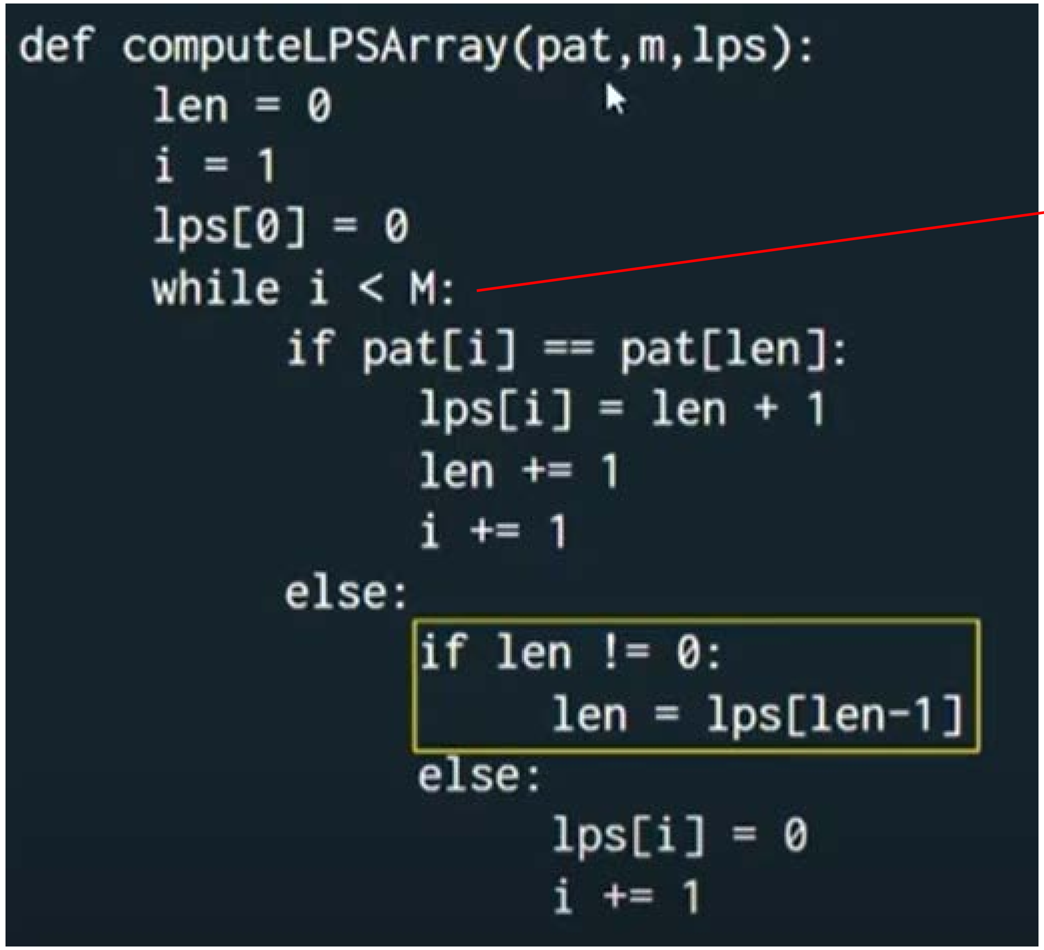
تحلیل پیچیدگی الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
  //Determine if pat is a substring of s
  int PosP=0, PosS=0;
  int  lengthP= pat.length(),  lengthS= length();
  while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {       //character match
                  PosP++;  PosS++;
          }
          else
                if ( PosP == 0) PosS++;
                else PosP=  pat.f [ PosP -1 ] ;
   if ( PosP<lengthP)  return -1;
   else  return PosS- lengthp;
}   // end of FastFind
```

$O(\text{ lengthS })$

اگر تابع شکست محاسبه شده باشد

تحلیل پیچیدگی تابع شکست



```
def computeLPSArray(pat,m,lps):
    len = 0
    i = 1
    lps[0] = 0
    while i < M:
        if pat[i] == pat[len]:
            lps[i] = len + 1
            len += 1
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
```

O(length)

تحلیل پیچیدگی الگوریتم کنوث-موریس-پرات KMP

```
int string::FastFind (String pat)
{
 //Determine if pat is a substring of s
 int PosP=0, PosS=0;
 int  lengthP= pat.length(),  lengthS= length();
 while  (( PosP< lengthP) && (PosS < lengthS))
          if (pat.str [PosP] == str [PosS] ) {       //character match
                 PosP++;  PosS++;
          }
          else
                    if ( PosP == 0) PosS++;
                    else PosP=  pat.f [ PosP -1 ] ;
 if ( PosP<lengthP)  return -1;
 else  return PosS- lengthp;
}   //  end of FastFind
```

O(lengthS + LengthP)

If $lengthS \gg LengthP$:
O(lengthS)