به نام خدا

ساختمان داده ها

جلسه پنجم

دانشگاه صنعتی همدان

گروه مهندسی کامپیوتر

نیم سال دوم 98-1397

تطابق الگو (Pattern Matching)

تطبیق الگو مساله ای است که تعیین می کند یک الگوی رشته داده شده P در متن رشته S وجود دارد یا خیر؟

- ✓ بدترین روش برای این مسئله روش ترتیبی است که الگوی P را با هر یک از زیر رشته های S مقایسه می کند.
- ✓ این روش دارای زمان محاسباتی O(m.n) خواهد بود. که mطول P و n طول S می باشد.
- ✓ الگوریتم ایده آل برای این روش دارای مرتبهO(m+n) می باشد.

The Knuth-Morris-Pratt Algorithm

- الگوریتم کنوث، موریس و پرات به این صورت عمل می کند که اگر در جستجو با عدم تطابق برخورد کند از اطلاعات داخل الگو استفاده می کند.
- فرض کنید رشته $s=s_0s_1\ldots s_{m-1}$ مورد نظر است و می خواهیم تعیین کنیم آیا تطابقی که از s_i شروع بشود وجود دارد؟

s= a b ? ? ?

pat= a b c a b a b

تابع شکست (fail)

- اگر $p_0p_1.....p_{n-1}$ یک الگو باشد تابع شکست بصورت زیر تعریف می شود:

$$f(i)=\begin{cases} k \text{ بزرگترین مقدار} & p_0p_1.....p_k = p_{j-k}p_{j-k+1}.... p_j \text{ به طوری که} \\ -1 & \end{cases}$$

J	0	1	2	3	4	5	6	7	8	9
pat	a	b	c	a	b	c	a	c	a	b
f	-1	-1	-1	0	1	2	3	-1	0	1

```
Void string:: fail()
{ int Lengthp= Length;
f[0]=-1;
for (int j=1;j<Length; j++){
int i=f[j-1];
While((*(str + j)!= *(str+i+1)) &&( i>=0)) i=f[i];
if (*(str + j)== *(str+i+1)) f[j]=i+1;
else f[j]=-1;
}
}
```

```
int string::FastFind(string pat){
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]+1;
} //end while
if (PosP < Length) return -1;else return PosS-
   LengthS;}
```

Illustration: given a String ‘S’ and pattern ‘p’ as follows:

S

b	a	c	b	a	b	a	b	a	b	a	c	a	c	a

p

a	b	a	b	a	c	a

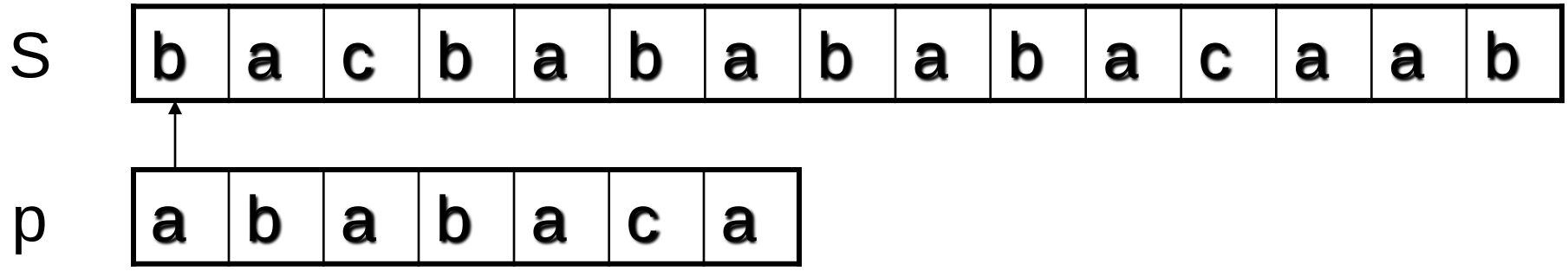
Let us execute the KMP algorithm to find whether ‘p’ occurs in ‘S’.

q	0	1	2	3	4	5	6
p	a	b	a	b	a	c	a
f	-1	-1	0	1	2	-1	0

Initially: n = size of S = 15;
m = size of p = 7

Step 1: i = 0, q = 0

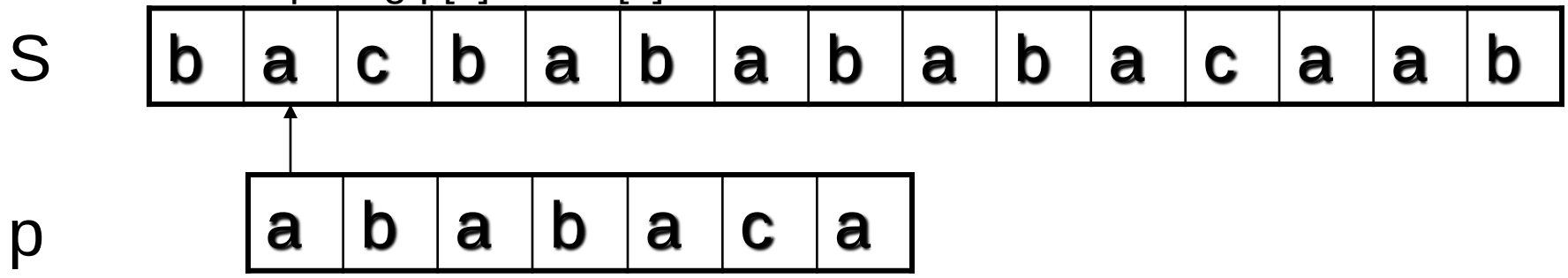
comparing p[0] with S[0]



P[1] does not match with S[1]. ‘p’ will be shifted one position to the right.

Step 2: i = 1, q = 0

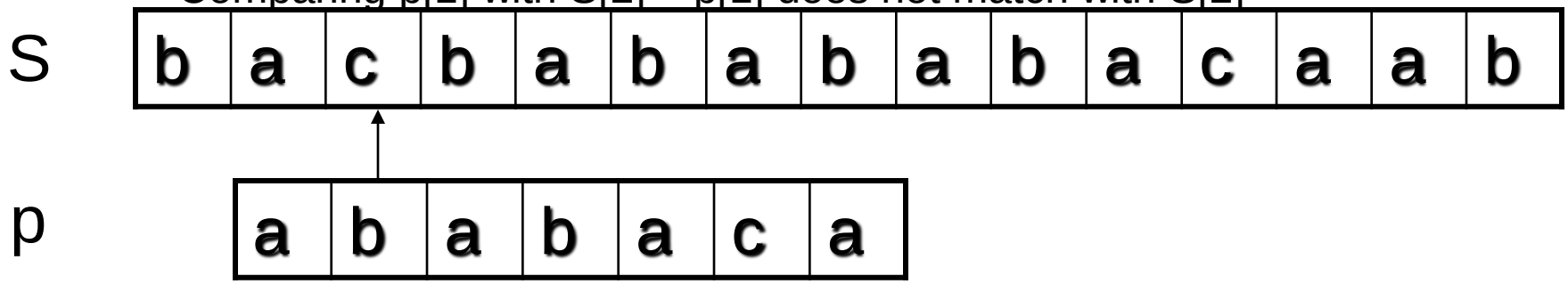
comparing p[0] with S[1]



P[0] matches S[1]. Since there is a match, p is not shifted.

Step 3: i = 2, q = 1

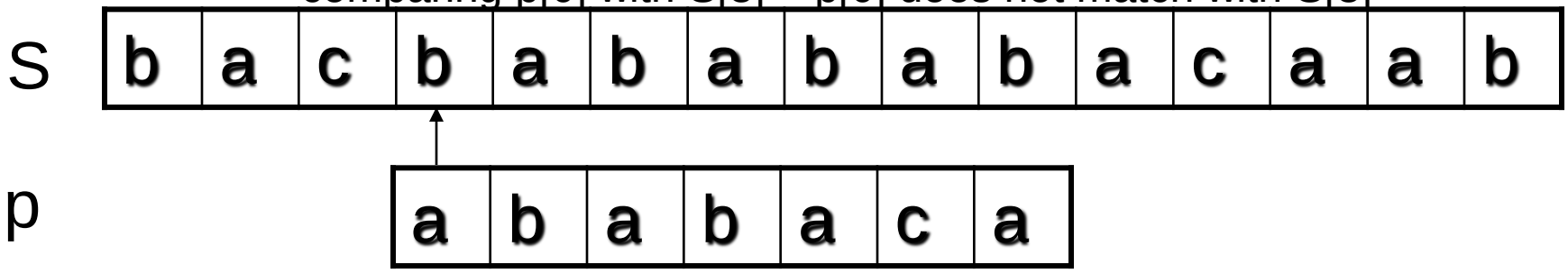
Comparing p[1] with S[2] p[1] does not match with S[2]



Backtracking on p, comparing p[0] and S[2]

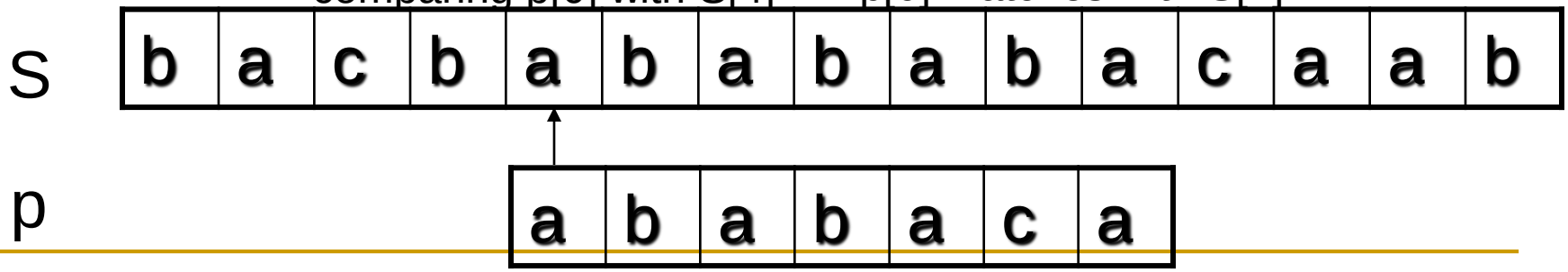
Step 4: i = 3, q = 0

comparing p[0] with S[3] p[0] does not match with S[3]



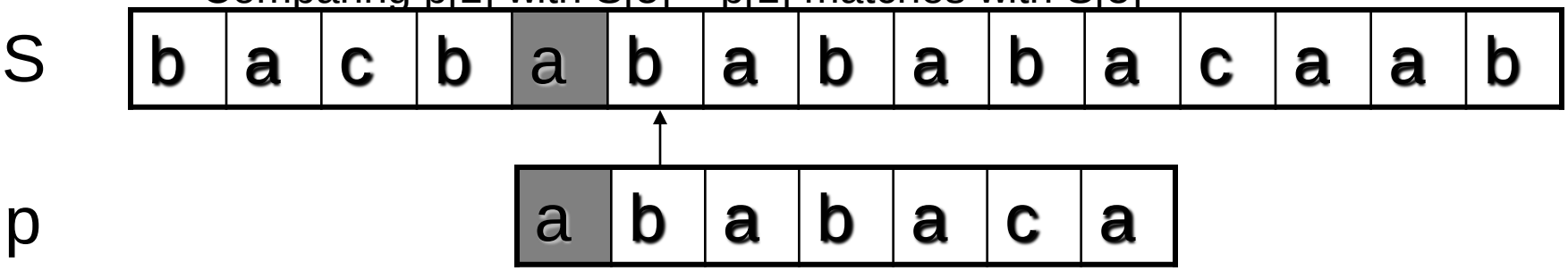
Step 5: i = 4, q = 0

comparing p[0] with S[4] p[0] matches with S[4]



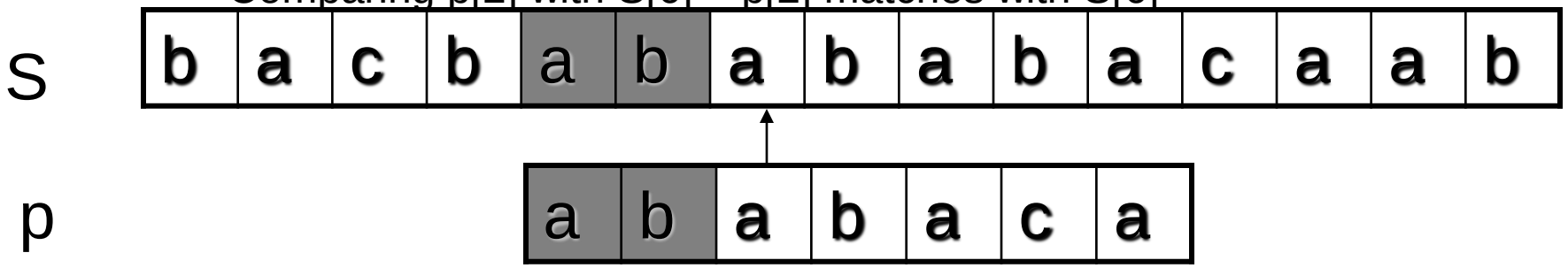
Step 6: i = 5, q = 1

Comparing p[1] with S[5] p[1] matches with S[5]



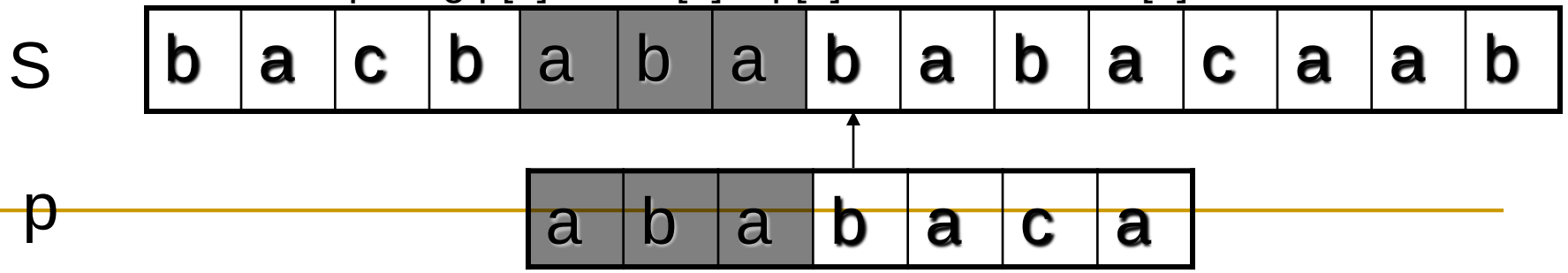
Step 7: i = 6, q = 2

Comparing p[2] with S[6] p[2] matches with S[6]



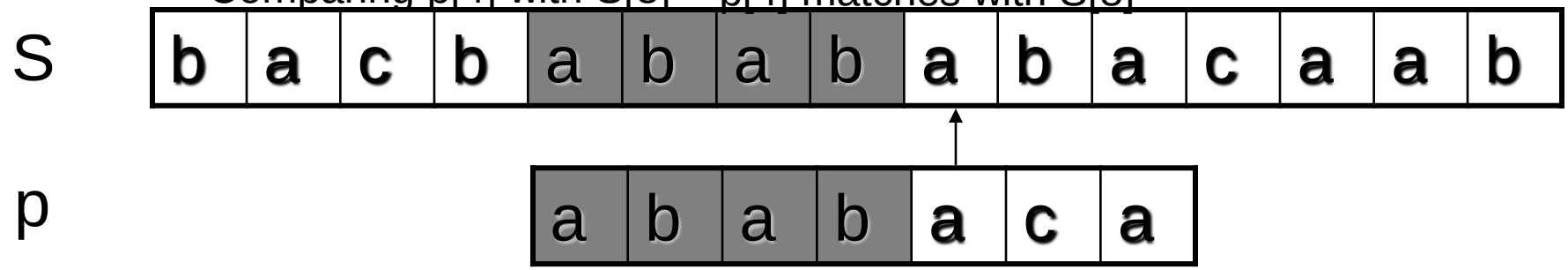
Step 8: i = 7, q = 3

Comparing p[3] with S[7] p[3] matches with S[7]



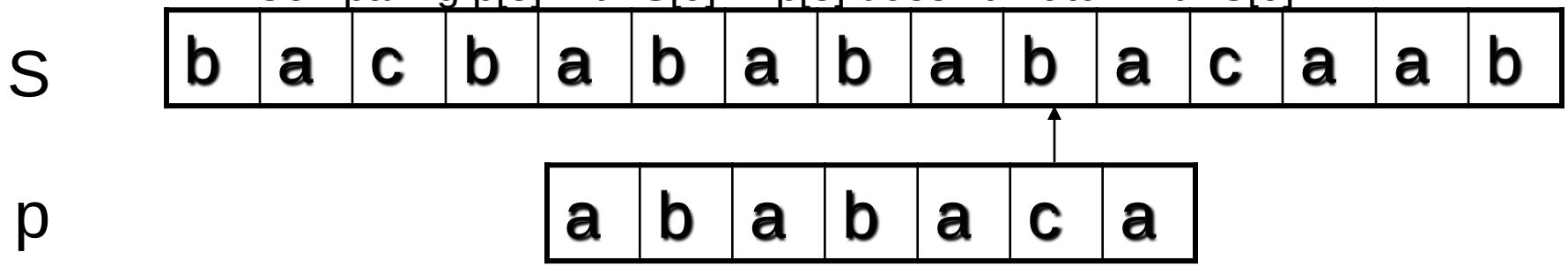
Step 9: i = 8, q = 4

Comparing p[4] with S[8] p[4] matches with S[8]



Step 10: i = 9, q = 5

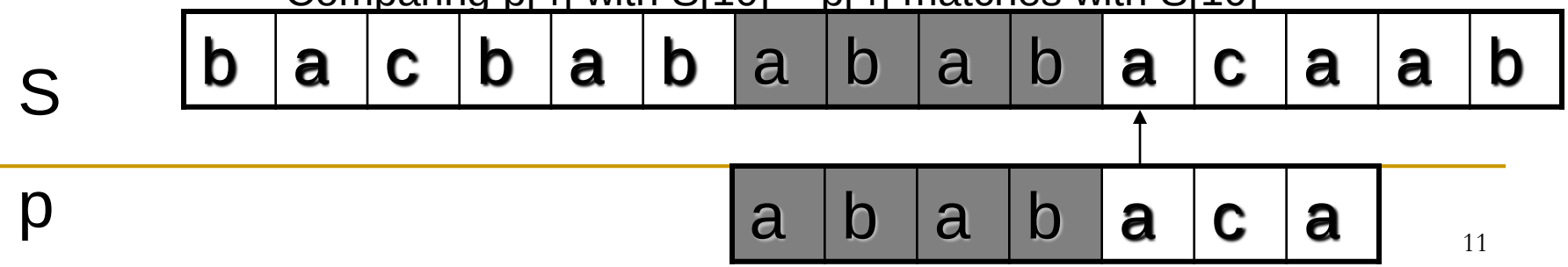
Comparing p[5] with S[9] p[5] doesn't match with S[9]



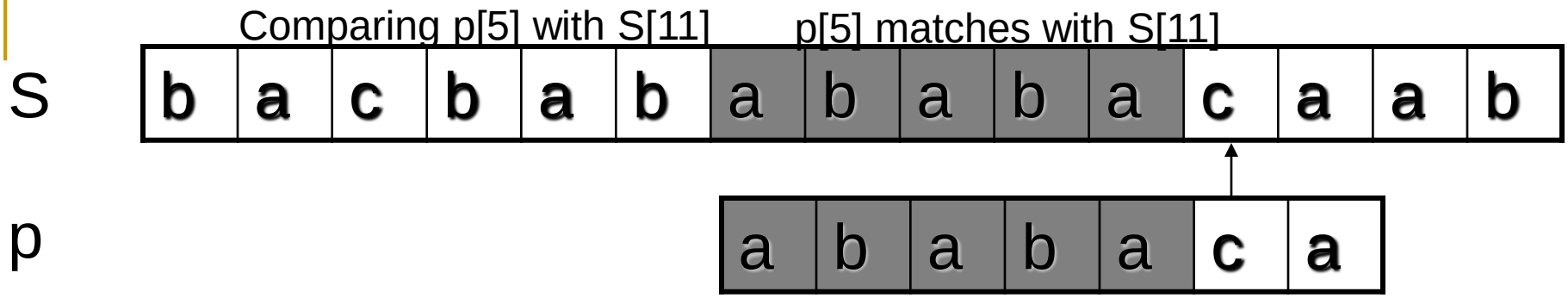
Backtracking on p, comparing p[3] with S[9] because after mismatch q = f[4] = 2

Step 11: i = 10, q = 4

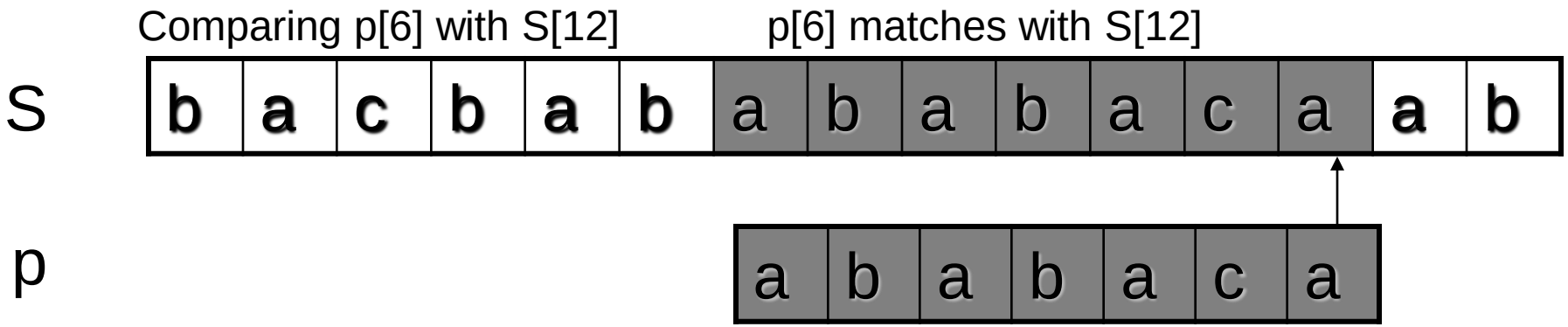
Comparing p[4] with S[10] p[4] matches with S[10]



Step 12: i = 11, q = 5



Step 13: i = 12, q = 6



Pattern ‘p’ has been found to completely occur in string ‘S’. The total number of shifts that took place for the match to be found are: i – m = 12- 6 = 6 shifts.

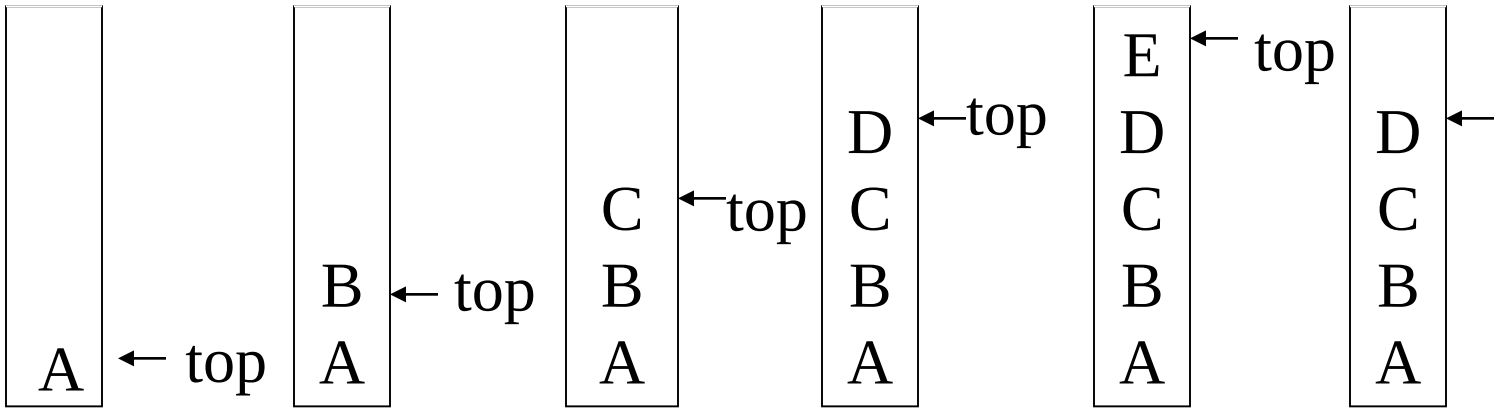
فصل سوم

پشته و صف

Stack & Queue

پشته

- پشته ساختمان داده ای است که داده ها را به ترتیب خاصی ذخیره می کند.
- در پشته اخرین عضوی که وارد می شود اولین عضوی است که خارج می شود.LAST IN FIRST OUT (LIFO)
- عنصر بالای پشته را top می گویند.


A
top
B
A
top
C
B
A
top
D
C
B
A
top
E
D
C
B
A
top
D
C
B
A

پشته ADT

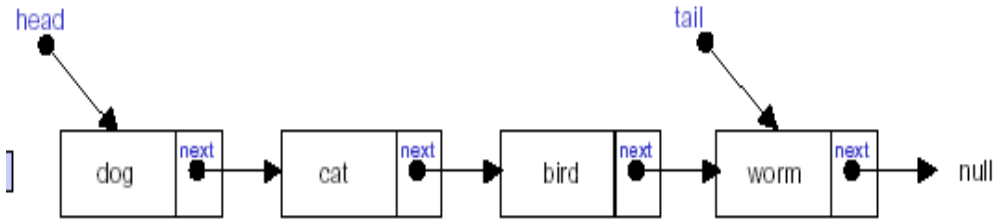
objects: a finite ordered list with zero or more elements.
methods:
for all *stack* ∈ *Stack*, *item* ∈ *element*, *max_stack_size* ∈ positive integer
Stack createS(*max_stack_size*) ::=
create an empty stack whose maximum size is *max_stack_size*
Boolean isFull(*stack, max_stack_size*) ::=
if (number of elements in *stack* == *max_stack_size*)
return TRUE
else return FALSE
Stack push(*stack, item*) ::=
if (IsFull(*stack*)) *stack_full*
else insert *item* into top of *stack* and **return**
Boolean isEmpty(*stack*) ::=
if(*stack* == CreateS(*max_stack_size*))
return TRUE
else return FALSE
Element pop(*stack*) ::=
if(IsEmpty(*stack*)) **return**
else remove and return the *item* on the top of the stack.

پیاده سازی پشته ها

با چندین روش می توان آنها را پیاده سازی کرد:

- آرایه ها

- لیستهای پیوندی (در قسمت لیستهای پیوندی بررسی می شود)



پیاده سازی پشته ها به روش ارایه – نمایش داده ها

- برای اینکه بتوانیم از پشته ها برای انواع داده های مختلف استفاده کنیم از قالبها (template) استفاده می کنیم.

```
private:
     int top;
     KeyType *stack;
     int MaxSize;
```

سازنده کلاس پشته

```
template <class KeyType>
Stack<KeyType>::Stack (int MaxStackSize):MaxSize (MaxStackSize)
{
    stack = new KeyType[MaxSize];
    top = -1;
}
```

پیاده سازی اعمال روی پشته ها

```
template <class KeyType>
inline Boolean  Stack<KeyType>::IsFull()
{
    if (top == MaxSize -1) return TRUE;
    else return FALSE;
}

template <class KeyType>
inline Boolean  Stack<KeyType>::IsEmpty()
{
    if (top == -1) return TRUE;
    else return FALSE;
}
```

پیاده سازی اعمال روی پشته ها

تابع افزودن یک عنصر به پشته

```
template <class KeyType>
void Stack<KeyType>::Add (const KeyType& x)
// add x to the stack
{
    if (IsFull()) StackFull();
    else stack[++top] = x;
}
```

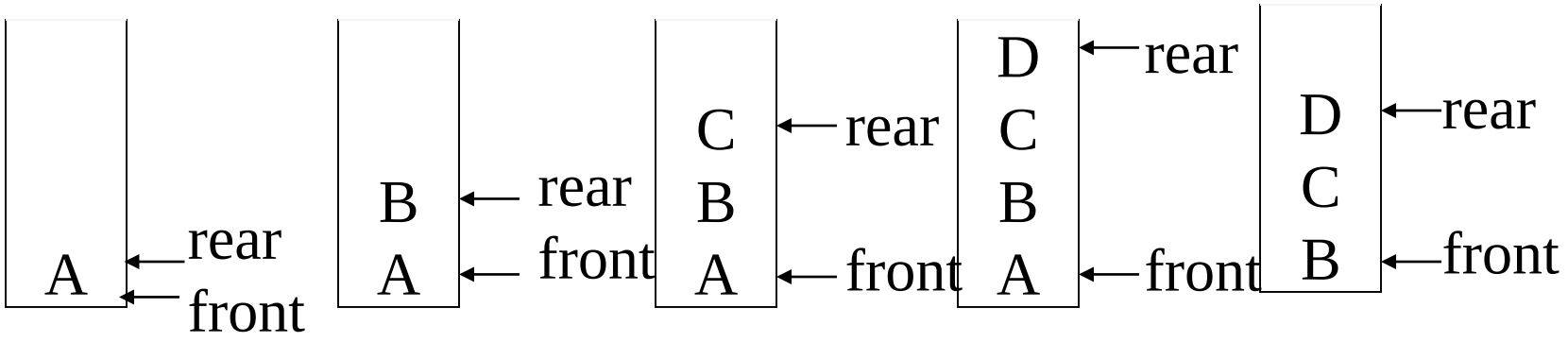
تابع حذف یک عنصر از پشته

```
template <class KeyType>
KeyType* Stack<KeyType>::Delete (KeyType& x)
// remove and return top element from stack
{
    if (IsEmpty()) {StackEmpty(); return 0;}
    x = stack[top--];
    return &x;
}
```

```
void main()
{
    Stack<int> s;
    int x;
    s.Add(5);
    s.Add(7);
    s.Delete(x);
    s.Add(9);
    s.Add(10);
    s.Delete(x);
    s.Delete(x);
    s.Delete(x);
}
```

صف (Queue)

- صف یک لیست ترتیبی است که تمام درج ها از یک سمت و حذف ها از سمت دیگر انجام می گیرد.
- بنابر این به صف FIFO (First In First Out) می گویند.
- اولین عضو ورودی اولین عضوی است که خارج می شود.
- عنصر ابتدای صف را Front و انتهای صف را Rear می گویند.


A
rear
front
B
A
rear
front
C
B
A
rear
front
D
C
B
A
rear
front
D
C
B
rear
front

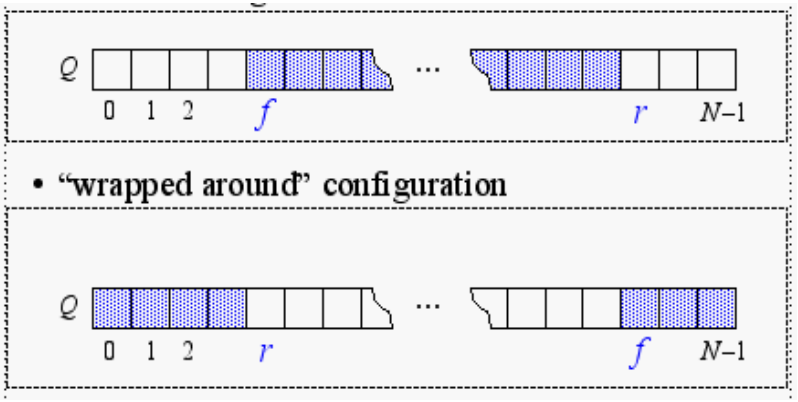
صف ADT

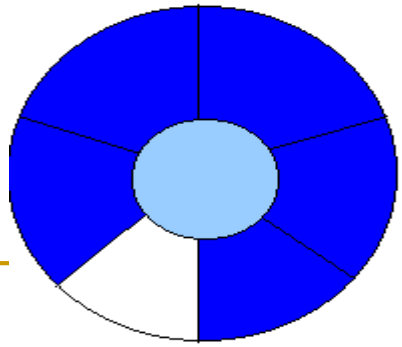
objects: a finite ordered list with zero or more elements.
methods:
for all *queue* ∈ *Queue*, *item* ∈ *element*,
max_ queue_ size ∈ positive integer
Queue createQ(*max_queue_size*) ::=
create an empty queue whose maximum size is
max_queue_size
Boolean isFullQ(*queue, max_queue_size*) ::=
if(number of elements in *queue* == *max_queue_size*)
return *TRUE*
else return *FALSE*
Queue Enqueue(*queue, item*) ::=
if (IsFullQ(*queue*)) *queue_full*
else insert *item* at rear of *queue* and return *queue*
Boolean isEmptyQ(*queue*) ::=
if (*queue* ==CreateQ(*max_queue_size*))
return *TRUE*
else return *FALSE*
Element dequeue(*queue*) ::=
if (IsEmptyQ(*queue*)) **return**
else remove and return the *item* at front of queue.

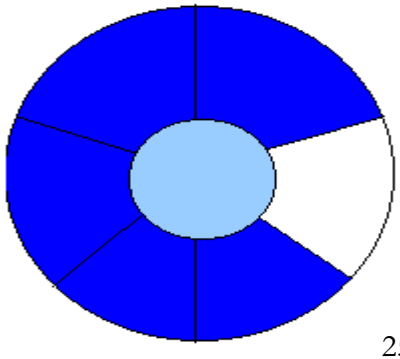
پیاده سازی صف

به چندین روش می توان صف را پیاده کرد:

- استفاده از ارایه ها
  - استفاده از ارایه با دو متغیر front و rear
  - استفاده از ارایه با یک متغیر rear (front همیشه صفر است)
  - استفده از ارایه به صورت حلقوی
- استفاده از لیستهای پیوندی







ساختار داده ای صف

```
private:
    int front;
    int rear;
    KeyType *queue;
    int MaxSize;
```

سازنده صف

```
template <class KeyType>
Queue<KeyType>::Queue (int MaxQueueSize) :
MaxSize(MaxQueueSize)
{
    queue = new KeyType[MaxSize];
    front = rear = -1;
}
```

پیاده سازی اعمال صف

تابع بررسی پر بودن صف

```
template <class KeyType>
inline Boolean  Queue<KeyType>::IsFull()
{
    if (rear == MaxSize -1) return TRUE;
    else return FALSE;
}
```

تابع بررسی خالی بودن صف

```
template <class KeyType>
inline Boolean  Queue<KeyType>::IsEmpty()
{
    if (front == rear) return TRUE;
    else return FALSE;
}
```

پیاده سازی اعمال صف

تابع اضافه کردن یک عنصر به صف

```
template <class KeyType>
void Queue<KeyType>::Add (const KeyType& x)
// add x to the queue
{
    if (IsFull()) QueueFull();
    else queue[++rear] = x;
}
```

تابع حذف کردن یک عنصر از صف

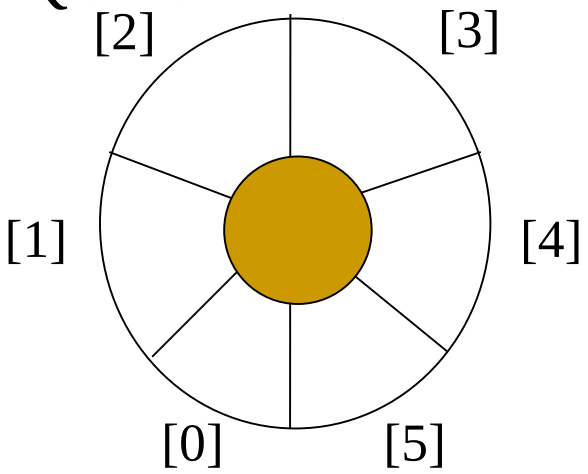
```
template <class KeyType>
KeyType* Queue<KeyType>::Delete (KeyType& x)
// remove and return front element from queue
{
    if (IsEmpty()) {QueueEmpty(); return 0;}
    x = queue[++front];
    return &x;
}
```

```
void main()
{
     Queue<int> s(2);
     int x;
     s.Add(5);
     s.Add(7);
     s.Delete(x);
     s.Add(9);
     s.Add(10);
     s.Delete(x);
     s.Delete(x);
     s.Delete(x);
}
```

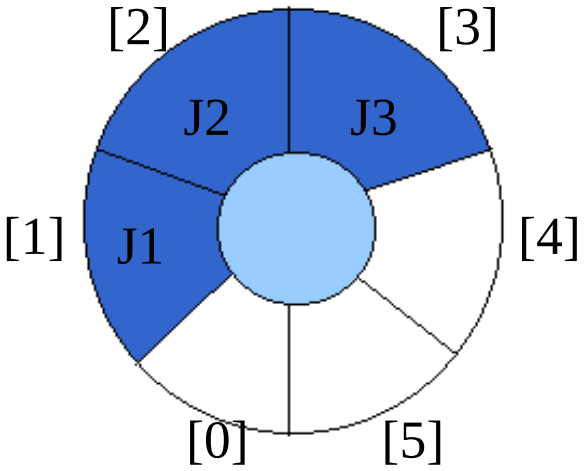
معایب این روش

- در این روش اعضا به ترتیب وارد صف می شوند و به ترتیب از انتهای صف حذف می شوند ولی فضای عناصری که حذف می شوند دوباره استفاده نمی شوند و صف بعد از مدتی دیگر فضای خالی نخواهد داشت:
- دو روش برای حل این معضل وجود دارد
  - شیفت دادن عناصر
  - استفاده از صف دایره ای

EMPTY QUEUE

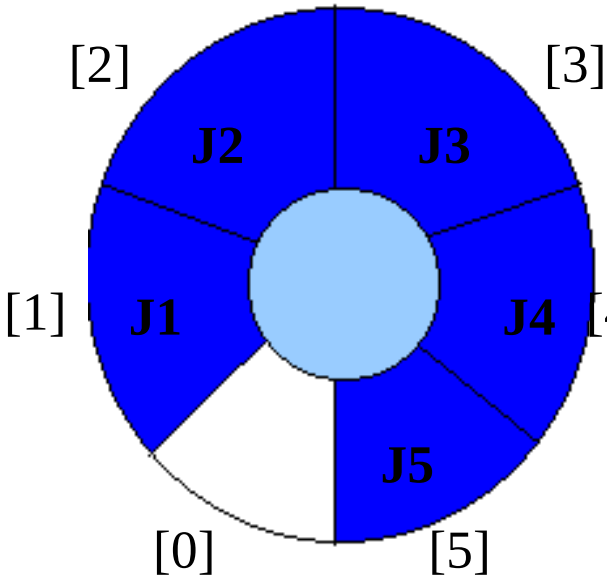


front = 0
rear = 0



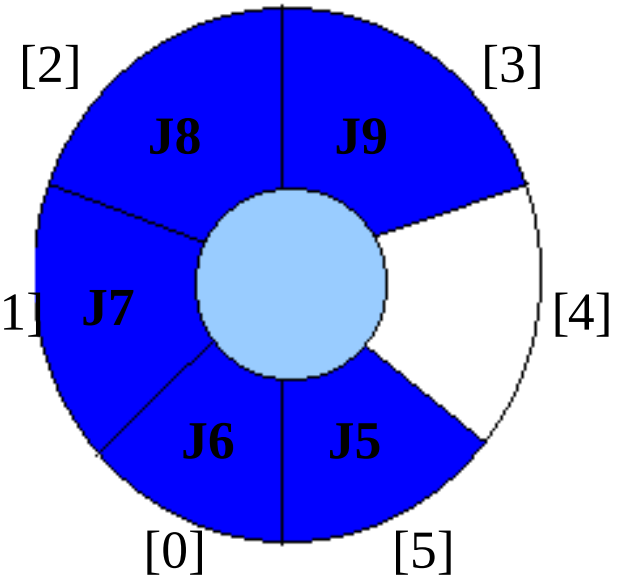
front = 0
rear = 3

FULL QUEUE



front =0
rear = 5

FULL QUEUE



front =4
rear =3

پیاده سازی صف حلقوی

```
private:
    int front;
    int rear;
    KeyType *queue;
    int MaxSize;
```

سازنده کلاس

```
template <class KeyType>
Queue<KeyType>::Queue (int MaxQueueSize) :
MaxSize(MaxQueueSize)
{
    queue = new KeyType[MaxSize];
    front = rear = 1;
}
```

توابع پر بودن و خالی بودن صف

```
template <class KeyType>
Boolean  Queue<KeyType>::IsFull()
{
    if (rear == MaxSize -1) return TRUE;
    else return FALSE;
}

template <class KeyType>
Boolean  Queue<KeyType>::IsEmpty()
{
    if (front == rear) return TRUE;
    else return FALSE;
}
```

حذف و اضافه کردن در صف حلقوی

```
template <class KeyType>
void Queue<KeyType>::Add (const KeyType& x)
// add x to the queue
{
    int k = (rear + 1) % MaxSize;
    if (front == k) QueueFull();
    else queue[rear = k] = x;
}

template <class KeyType>
KeyType* Queue<KeyType>::Delete (KeyType& x)
// remove and return top element from queue
{
  if (front == rear) {QueueEmpty(); return 0;}
    x = queue[++front %= MaxSize];
    return &x;
}
```

```
void main()
{
    Queue<int> s(2);
    int x;
    s.Add(5);
    s.Add(7);
    s.Delete(x);
    s.Add(9);
    s.Add(10);
    s.Delete(x);
    s.Delete(x);
    s.Delete(x);
}
```