ساختمان داده‌ها و الگوریتم‌ها

(ریاضی)

نیمسال اول ۱۳۹۰-۱۳۸۹

فصل دوم: ساختمان داده‌های پایه‌ای

آرایه‌ها

ساختمان داده‌ها یا ساختار داده‌ها در حالت کلی به دو دسته خطی و غیرخطی تقسیم می‌شوند. ساختمان داده‌ای را خطی گویند، هرگاه عناصر آن تشکیل یک دنباله دهند. به بیان دیگر یک لیست خطی باشد. برای نمایش ساختمان داده خطی در حافظه دو روش اساسی وجود دارد. یکی از این روش‌ها عبارت است از داشتن رابطه خطی بین عناصری که به‌وسیله خانه‌های متوالی حافظه نمایش داده می‌شود. این ساختارهای خطی آرایه‌ها نام دارد که موضوع اصلی این فصل را تشکیل می‌دهند.

روش دیگر عبارت است از داشتن رابطه خطی بین عناصری که به‌وسیله اشاره‌گرها یا پیوندها نمایش داده می‌شود. این ساختارهای خطی لیستهای پیوندی نام دارد که موضوع اصلی مطالب فصل‌های بعدی را تشکیل می‌دهد.

عملیات معمول بر روی یک ساختار خطی

(الف) پیمایش: رویت کردن همه عناصر داخل لیست را پیمایش گویند.

(ب) جستجوکردن: پیداکردن مکان یک عنصر با یک مقدار داده شده یا رکورد با یک کلید معین را جستجو گویند.

(ج) اضافه کردن: افزودن یک عنصر جدید به لیست را اضافه کردن، گویند.

(د) حذف کردن: حذف یک عنصر از لیست را حذف کردن، گویند.

(ه)مرتب کردن: تجدید آرایش عناصر با یک نظم خاص را مرتب کردن، گویند.

(و) ادغام کردن: ترکیب دو لیست در یک لیست را ادغام کردن، گویند.

۱-۲ مفهوم نوع داده مجرد (Abstract Data Type)

۱-۲ مفهوم نوع داده مجرد (Abstract Data Type)

تعریف نوع داده: نوع داده مجموعه‌ای از انواع داده مقصد (object) و عملگرهایی است که بر روی این نوع داده‌ها عمل می‌کنند.

۱-۲ مفهوم نوع داده مجرد (Abstract Data Type)

تعریف نوع داده: نـوع داده مجموعـه‌ای از انـواع داده مقـصد (object) و عملگرهـایی است که بر روی این نوع داده‌ها عمل می‌کنند.

برای مثال نوع داده صحیح شامل داده‌های مقصود بـین (Max int.....+۱,۰,-۱,...) بوده و عملگرهای حسابی +,۰,*,/ و غیره روی آنها عمل می‌کنند.

۱-۲ مفهوم نوع داده مجرد (Abstract Data Type)

تعریف نوع داده: نوع داده مجموعه‌ای از انواع داده مقصد (object) و عملگرهایی است که بر روی این نوع داده‌ها عمل می‌کنند.

برای مثال نوع داده صحیح شامل داده‌های مقصود بین (Max int.....+۱,۰,-۱,...) بوده و عملگرهای حسابی +,۰,*,/ و غیره روی آنها عمل می‌کنند.

تعریف نوع داده مجرد (ADT): نوع داده مجرد، یک نوع داده است که در ساختار آن نیاز به دانستن مشخصات داده‌های مقصود و مشخصات اعمالی (عملگرها) که بر روی آنها انجام می‌شود، می‌باشد و در آن قسمت نمایش مقصودها و پیاده‌سازی عملگر از یکدیگر متمایز می‌باشند.

۲-۲ آرایه‌ها

آرایه، لیستی از n عنصر یا مجموعه‌ای متناهی، از عناصر داده‌ای هم نوع می‌باشد (یعنی عناصر داده‌ای از یک نوع هستند) به‌طوری که:

الف) به عناصر آرایه به ترتیب و یا مستقیم (تصادفی) و به کمک یک مجموعه از اندیس‌ها می‌توان دسترسی پیدا کرد.

ب) عناصر آرایه به ترتیب در خانه‌های متوالی حافظه ذخیره می‌شوند.

۲-۲ آرایه‌ها

آرایه، لیستی از n عنصر یا مجموعه‌ای متناهی، از عناصر داده‌ای هم نوع می‌باشد (یعنی عناصر داده‌ای از یک نوع هستند) به‌طوری که:

الف) به عناصر آرایه به ترتیب و یا مستقیم (تصادفی) و به کمک یک مجموعه از اندیس‌ها می‌توان دسترسی پیدا کرد.

ب) عناصر آرایه به ترتیب در خانه‌های متوالی حافظه ذخیره می‌شوند.

۲-۳ آرایه به‌عنوان داده انتزاعی (Abstract Data Type)

۲-۲ آرایه‌ها

آرایه، لیستی از n عنصر یا مجموعه‌ای متناهی، از عناصر داده‌ای هم نوع می‌باشد (یعنی عناصر داده‌ای از یک نوع هستند) به‌طوری که:

الف) به عناصر آرایه به ترتیب و یا مستقیم (تصادفی) و به کمک یک مجموعه از اندیس‌ها می‌توان دسترسی پیدا کرد.

ب) عناصر آرایه به ترتیب در خانه‌های متوالی حافظه ذخیره می‌شوند.

۲-۳ آرایه به‌عنوان داده انتزاعی (Abstract Data Type)

۱. مجموعه عناصر

لیستی از مجموعه مرتب و متناهی که همه عناصر آن از یک نوع می‌باشند.

۲. عملیات اصلی

دستیابی مستقیم یا تصادفی به هر عنصر آرایه به‌طوریکه بتوان عمل ذخیره و بازیابی را انجام داد.

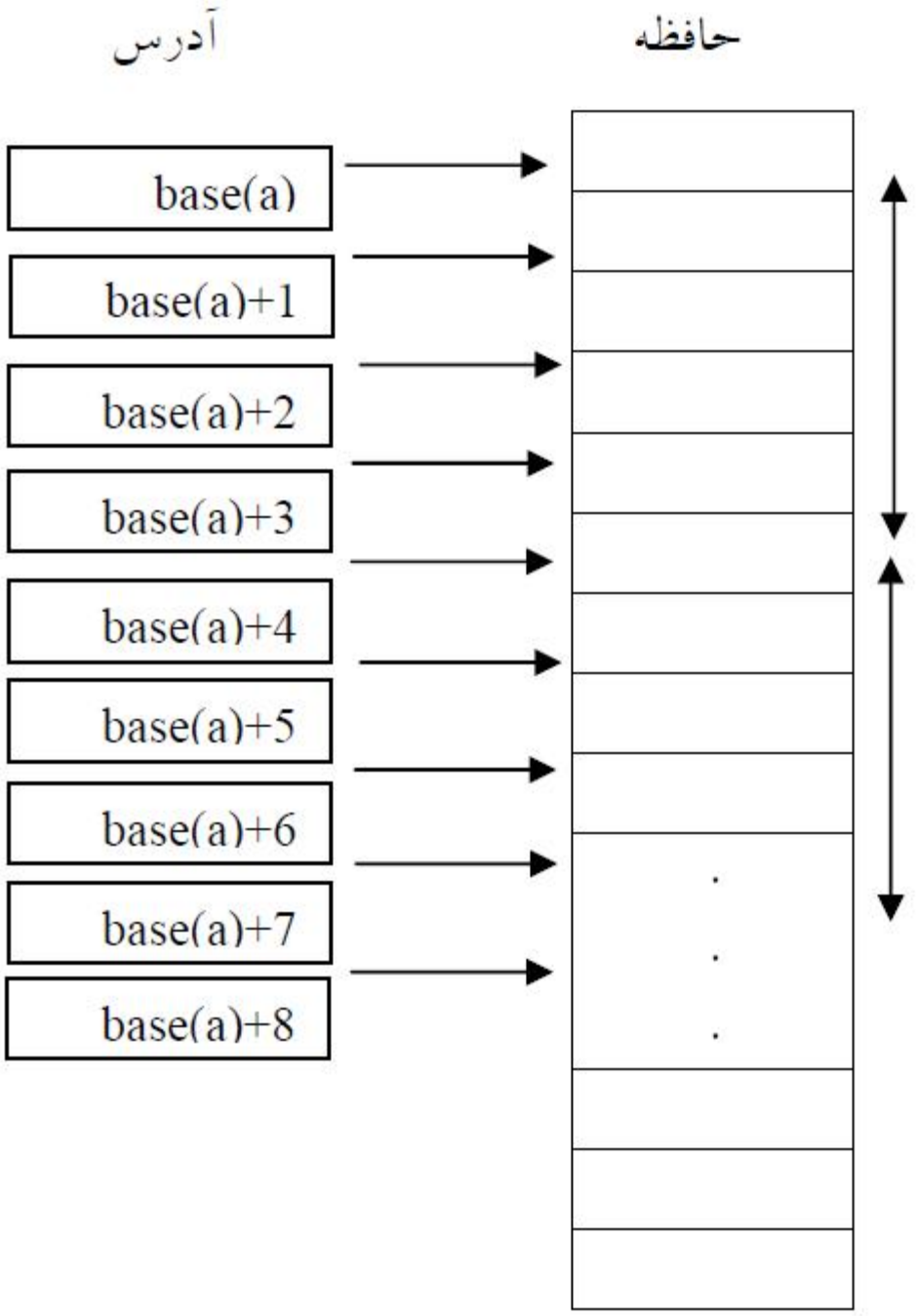
آرایه‌های یک بعدی:

آرایه‌های یک بعدی:

نمایش آرایه‌های یک بعدی:

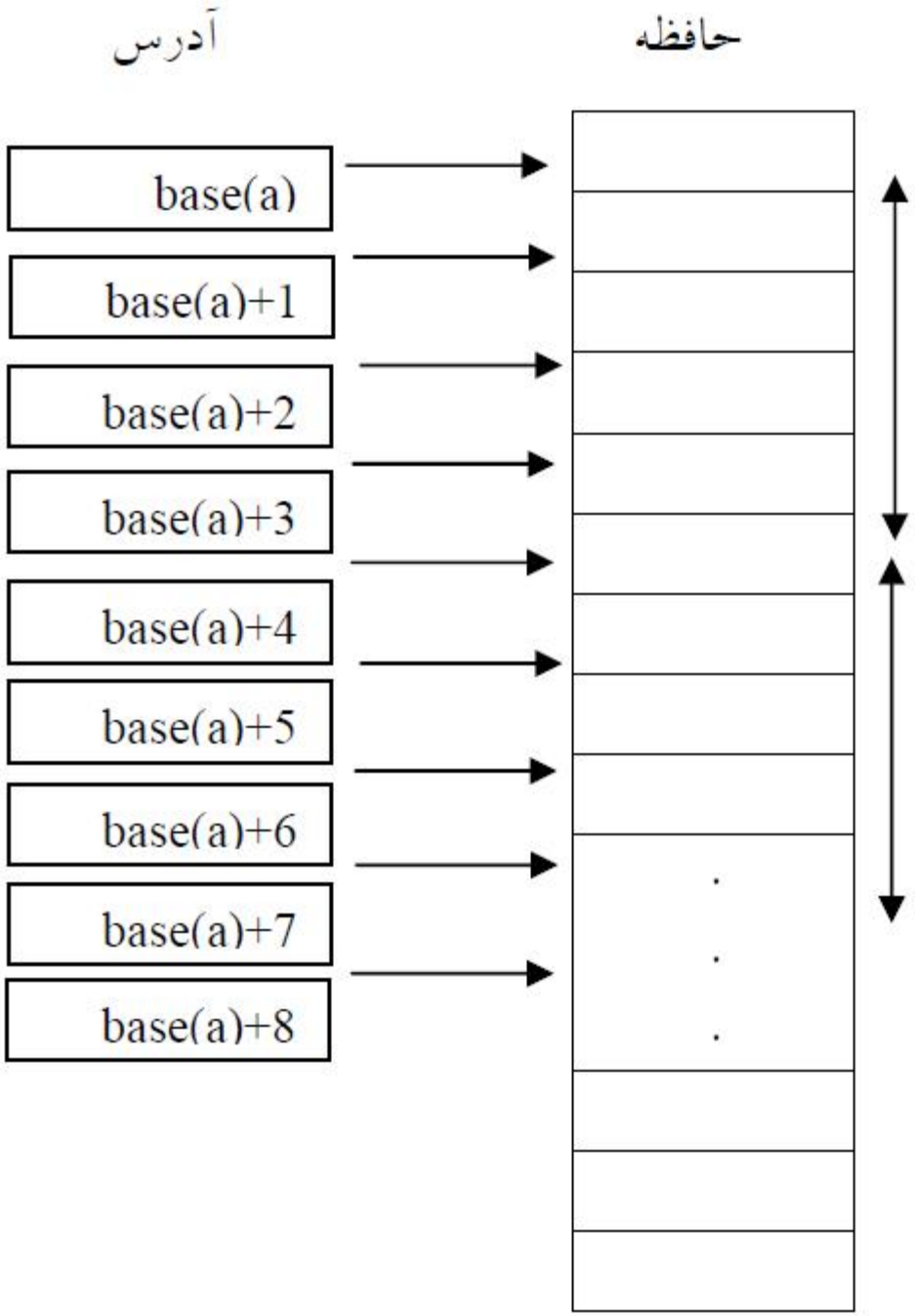
آرایه‌های یک بعدی:

نمایش آرایه‌های یک بعدی:



آرایه‌های یک بعدی:

نمایش آرایه‌های یک بعدی:



Loc(i)=base(a)+i*size

مثال از کاربردها:

مثال از کاربردها:

عنوان الگوریتم	الگوریتم جستجوی ترتیبی
ورودی	A آرایه‌ای از عناصر، n تعداد عناصر، x عنصر مورد جستجو.
خروجی	اندیس عنصر مورد جستجو در صورت وجود.

```
int   Seq_Search (elementtype a[ ] ,  int    n, elementtype   item)
{
      int    i;
      for (i=0 ;  i < n  ;  i++)
          if (a[ i ] == item)
              return (i);
       return (-1);
}
```

مثال از کاربردها:

عنوان الگوریتم	الگوریتم جستجوی ترتیبی
ورودی	A آرایه‌ای از عناصر، n تعداد عناصر، x عنصر مورد جستجو.
خروجی	اندیس عنصر مورد جستجو در صورت وجود.

```
int   Seq_Search (elementtype a[ ] ,  int    n, elementtype   item)
{
      int    i;
      for (i=0 ;  i < n  ;  i++)
           if (a[ i ] == item)
                return (i);
       return (-1);
}
```

- محاسبه زمان و پیچیدگی الگوریتم جستوی خطی

مثال از کاربردها:

مثال از کاربردها:

عنوان الگوریتم	الگوریتم جستجوی دودوئی
ورودی	آرایه n عنصری مرتب شده صعودی a و item که باید جستجو شود.
خروجی	اگر عنصر item پیدا شود flag=1 و موقعیت آن را بر می‌گرداند در غیر این‌صورت flag=0 و مقدار ۱- را بر می‌گرداند..

```
int   Binary_Search (elementtype a[ ] ,  int    n, elementtype   item)
{
       int  mid , flag = 0 , first = 0 , last = n-1 ;
      while (first < = last  &&  ! flag)
      {
           mid = (first + last)/2 ;
           if (item <a[mid])
                 last = mid -1 ;
          else  if (item > a [mid])
                 first = mid + 1 ;
          else
           {
                 flag = 1 ;
                 return mid ;
           }
       }// End While
       retrurn -1 ;
}
```

مثال از کاربردها:

عنوان الگوریتم	الگوریتم جستجوی دودوئی
ورودی	آرایه n عنصری مرتب شده صعودی a و item که باید جستجو شود.
خروجی	اگر عنصر item پیدا شود flag=1 و موقعیت آن را بر می‌گرداند در غیر اینصورت flag=0 و مقدار ۱- را بر می‌گرداند..

```
int   Binary_Search (elementtype a[ ] ,  int    n, elementtype   item)
{
       int  mid , flag = 0 , first = 0 , last = n-1 ;
      while (first < = last  &&  ! flag)
      {
           mid = (first + last)/2 ;
           if (item <a[mid])
                 last = mid -1 ;
          else  if (item > a [mid])
                 first = mid + 1 ;
          else
           {
                 flag = 1 ;
                 return mid ;
           }
       }// End While
       retrurn -1 ;
}
```

- پیچیدگی الگوریتم جستجوی دودوئی

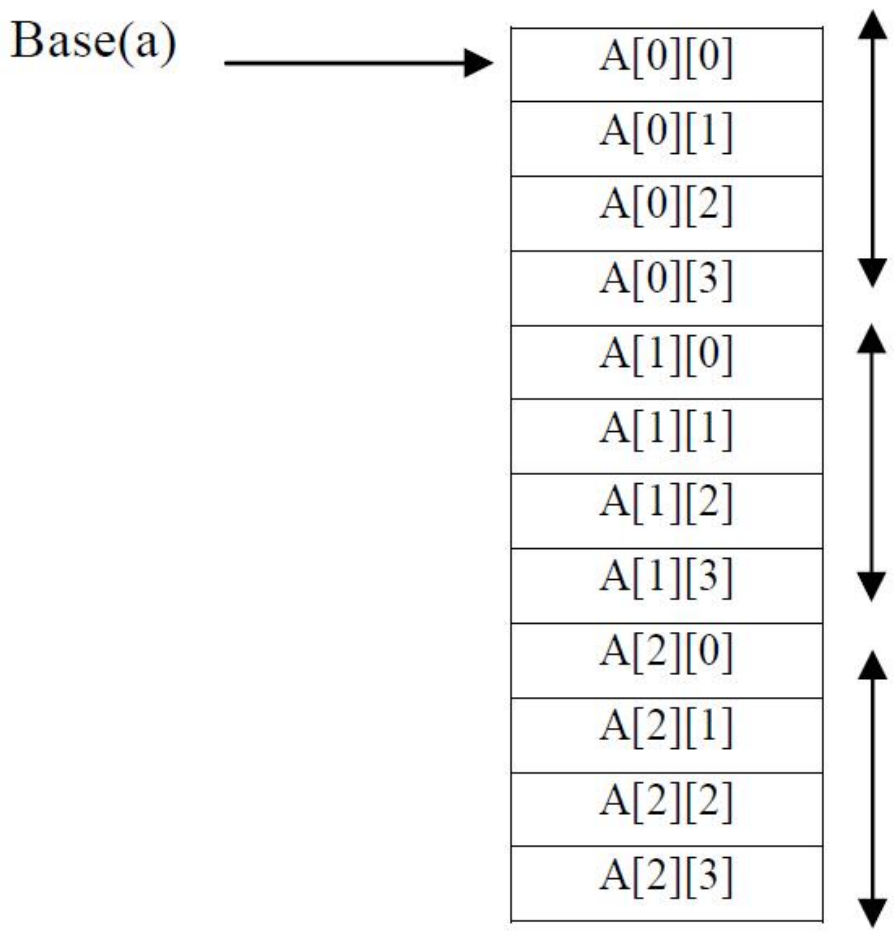
آرایه‌های دو بعدی:

$$\begin{bmatrix} A[۰][۰] & A[۰][۱] & A[۰][۲] \\ A[۱][۰] & A[۱][۱] & A[۱][۲] \\ A[۲][۰] & A[۲][۱] & A[۲][۲] \end{bmatrix}$$

آرایه‌های دو بعدی:

$$\begin{bmatrix} A[۰][۰] & A[۰][۱] & A[۰][۲] \\ A[۱][۰] & A[۱][۱] & A[۱][۲] \\ A[۲][۰] & A[۲][۱] & A[۲][۲] \end{bmatrix}$$

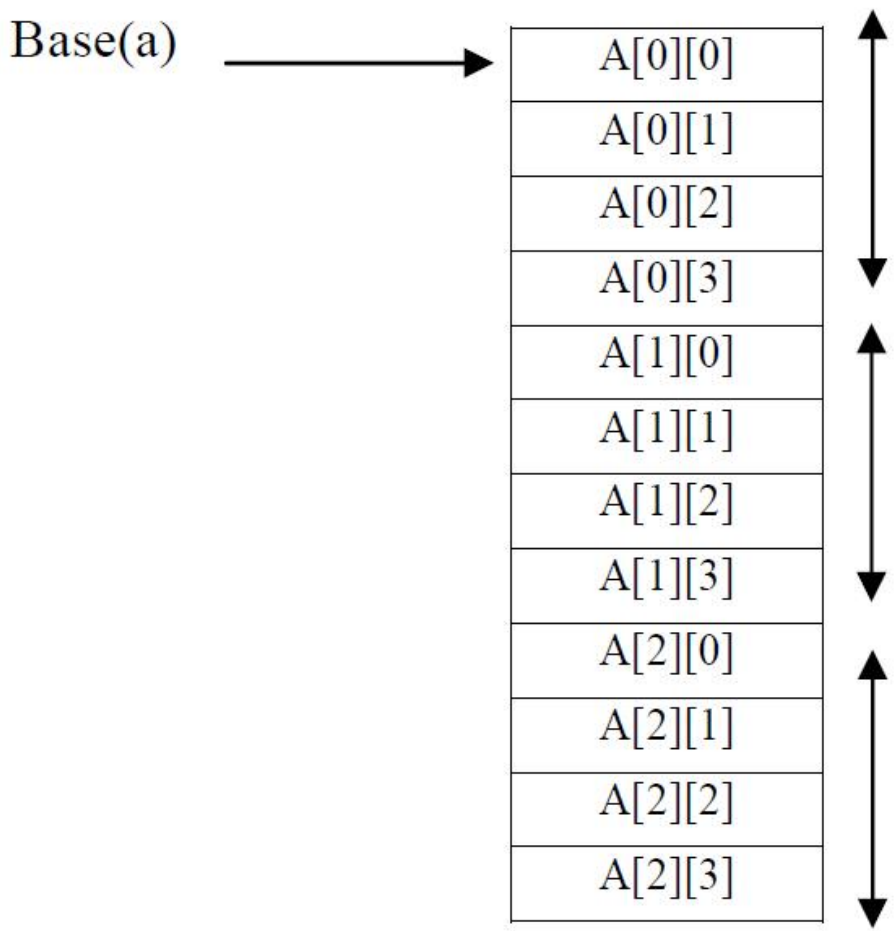
نحوه ذخیره‌سازی آرایه‌های دو بعدی:



آرایه‌های دو بعدی:

$$\begin{bmatrix} A[۰][۰] & A[۰][۱] & A[۰][۲] \\ A[۱][۰] & A[۱][۱] & A[۱][۲] \\ A[۲][۰] & A[۲][۱] & A[۲][۲] \end{bmatrix}$$

نحوه ذخیره‌سازی آرایه‌های دو بعدی:



$$\text{محل}[A[i][j]] = base(A) + (i * n + j) * size$$

ماتریس‌های تُنُک (Sparse):

ماتریس‌های تُنُک (Sparse):

ماتریسی که بیشتر درایه‌های آن صفر باشد.

ماتریس‌های تُنُک (Sparse):

ماتریسی که بیشتر درایه‌های آن صفر باشد.

بنابراین به‌نظر می‌رسد که لازم نباشد عناصر صفر ماترس در حافظه ذخیره شوند. لذا نمایش معمولی ماتریس‌ها برای نمایش یک ماتریس اسپارس مناسب نیست، بلکه باید نمایش دیگری را درنظر گرفت. در این نمایش فقط شماره سطر، شماره ستون و خود مقدار مربوط به عنصر غیر صفر باید نگهداری شود. ماتریس اسپارس را می‌توان در یک ماتریس که دارای سه ستون است ذخیره کرد، که در آن، ستون اول حاوی شماره سطر مقدار غیر صفر، ستون دوم حاوی شماره ستون مقدار غیر صفر و ستون سوم حاوی خود مقدار غیر صفر می‌باشد. در سطر اول ماتریس، مشخصات کلی ماتریس اسپارس را می‌نویسیم. اگر تعداد عناصر غیرصفر ماتریس اصلی n باشد، آنگاه نمایش ماتریس اسپارس را برای n+1 سطر خواهد بود.

ماتریس‌های تُنُک (Sparse):

$$A=\begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix} \rightarrow$$

(الف) ماتریس اسپارس

تعداد مقادیر غیر صفر

تعداد ستون

تعداد سطرها

3	5	3
0	1	3
1	0	6
2	3	2

(ب) نمایش ماتریس اسپارس

4*3

ماتریس‌های تُنُک (Sparse):

$$A=\begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix} \rightarrow$$

(الف) ماتریس اسپارس

تعداد مقادیر غیر صفر

تعداد ستون

تعداد سطرها

3	5	3
0	1	3
1	0	6
2	3	2

(ب) نمایش ماتریس اسپارس

4*3

نمایش اسپاس موجب صرفه‌جوئی در حافظه مصرفی می‌شود. اگر تعداد عناصر صفر زیاد باشد و ماتریس نیز بزرگ باشد، این صرفه‌جوئی در حافظه چشمگیر خواهد بود.

ماتریس‌های تُنُک (Sparse):

$$A=\begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix} \rightarrow$$

(الف) ماتریس اسپارس

تعداد مقادیر غیر صفر

تعداد ستون

تعداد سطرها

3	5	3
0	1	3
1	0	6
2	3	2

(ب) نمایش ماتریس اسپارس

4*3

آیا دسترسی تصادفی دارد؟

ترانهاده ماتریس تُنُک :

منظور از ترانهاده ماتریس، ماتریس دیگری است که جای سطر و ستون‌های آن عوض شده باشد. الگوریتم زیر روش پیداکردن ترانهاده ماتریس اسپارس که به‌صورت جدول ایندکسی با ابعاد $(N+1)\times 3$ (N تعداد عناصر غیرصفر) ذخیره شده است را پیدا می‌کند.

فقط جابجایی سطر و ستون کافی نیست؟

الگوریتم‌های تطابق الگو (Pattern Matching)

منظور از تطبیق الگو همانطور که در بالا اشاره کردیم، مسئله‌ای است که تعیین می‌کند یک الگوی رشته داده شده P در متن رشته‌ای T وجود دارد یا خیر. در الگوریتهای ارائه شده، فرض خواهیم کرد که همواره طول P کوچک‌تر از طول T باشد.

الگوریتم‌های تطابق الگو (Pattern Matching)

- **الگوریتم اول تطبیق الگو**

الگوریتم اول تطبیق الگو، الگوریتم ساده و غیرکارآمدی است که در آن الگوی داده شده P با همه زیررشته‌های T مقایسه می‌شود. عمل مقایسه با حرکت از چپ به راست رشته T انجام می‌شود تا به یک تطبیق با P برسد.

فرض کنید که:

$$W_K = \text{Subtring}(T, K, \text{Length}(P))$$

که در آن W_K زیررشته‌ای از T با طول P و با شروع از K امین کاراکتر T می‌باشد. ابتدا P را کاراکتر به کاراکتر با اولین زیررشته یعنی W_1 مقایسه می‌کنیم. اگر تمام کاراکترها مساوی باشند، در اینصورت $P=W_1$ و در نتیجه P در T ظاهر شده است. از سوی دیگر اگر به این نتیجه برسیم که یک کاراکتر P دارای متناظر در W_1 نیست در این‌صورت $P \neq W_1$ خواهد بود و به سراغ زیررشته بعدی یعنی W_2 می‌رویم. اعمال بالا را ادامه می‌دهیم تا زمانیکه، عمل مقایسه متوقف شود.

این روش دارای زمان محاسباتی $O(n.m)$ می‌باشد که در آن n طول زیررشته P و m طول رشته T می‌باشد.

الگوریتم‌های تطابق الگو (Pattern Matching)

- **الگوریتم دوم تطبیق الگو**

الگوریتم دوم تطبیق الگو، الگوریتمی است که در این الگوریتم وقتی طول الگو (P) بزرگ‌تر از تعداد کاراکترهای باقیمانده در رشته T می‌باشد از آن خارج می‌شود، و از تست اولین و آخرین کاراکتر P و T قبل از تست بقیه کاراکترها استفاده می‌کند.

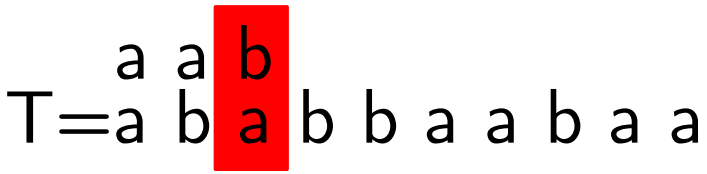
الگوریتم بدین صورت کار می‌کند که، در رشته T از ابتدا با استفاده از اندیس start به اندازه تعداد کاراکترهای الگو(P) انتخاب می‌شود و کاراکتر آخر انتخاب شده در T (Endmatch) با کاراکتر آخر الگو مقایسه می‌شود و در صورت مساوی بودن، سایر کاراکترها بررسی می‌شوند و در غیراینصورت اندیس Start یک مکان به جلو حرکت می‌کند و از آن مکان به اندازه کاراکترهای الگو انتخاب می‌شود. مراحل بالا را تا زمانیکه مسئله حل نشده تکرار می‌کنیم.

الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : P=a a b T=a b a b b a a b a a

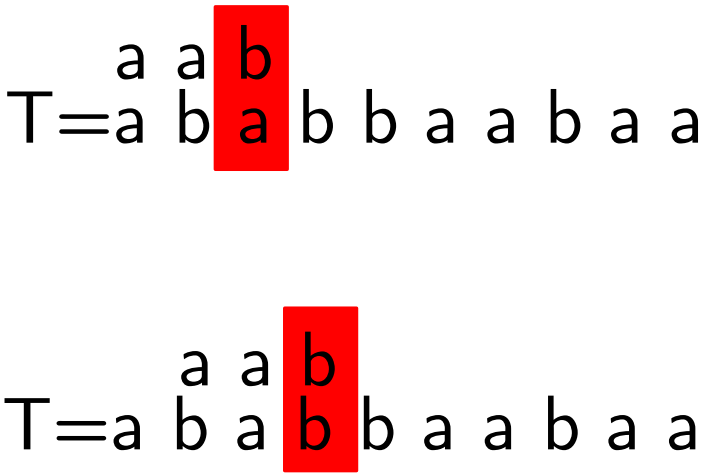
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



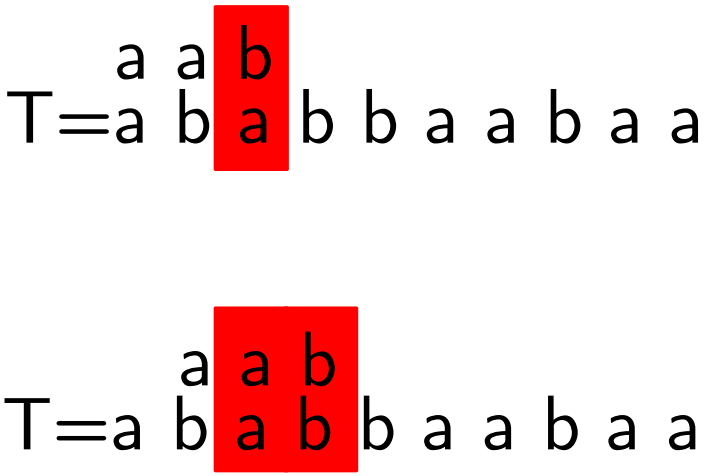
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



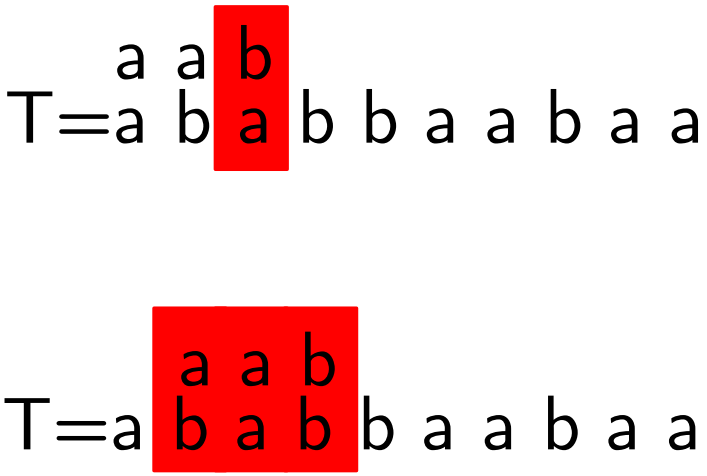
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



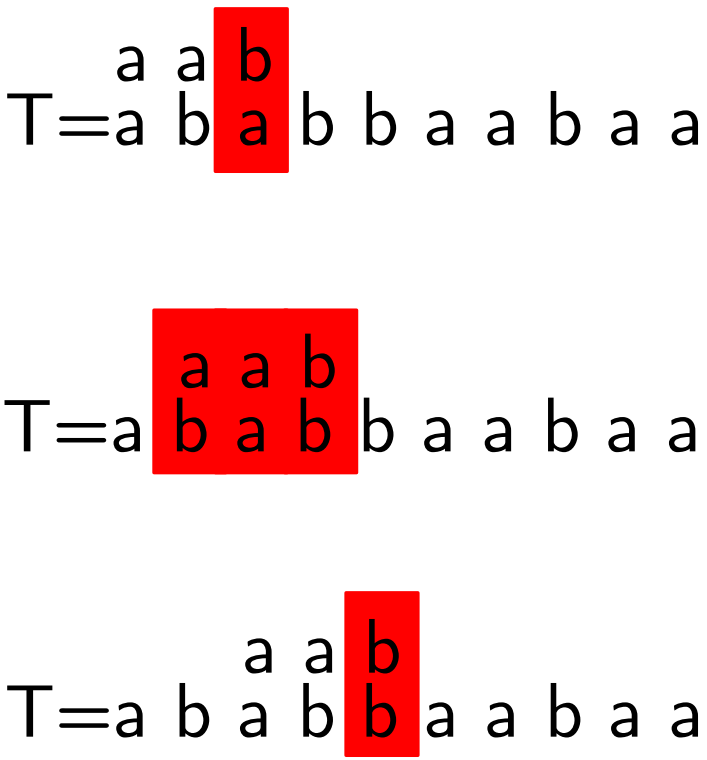
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



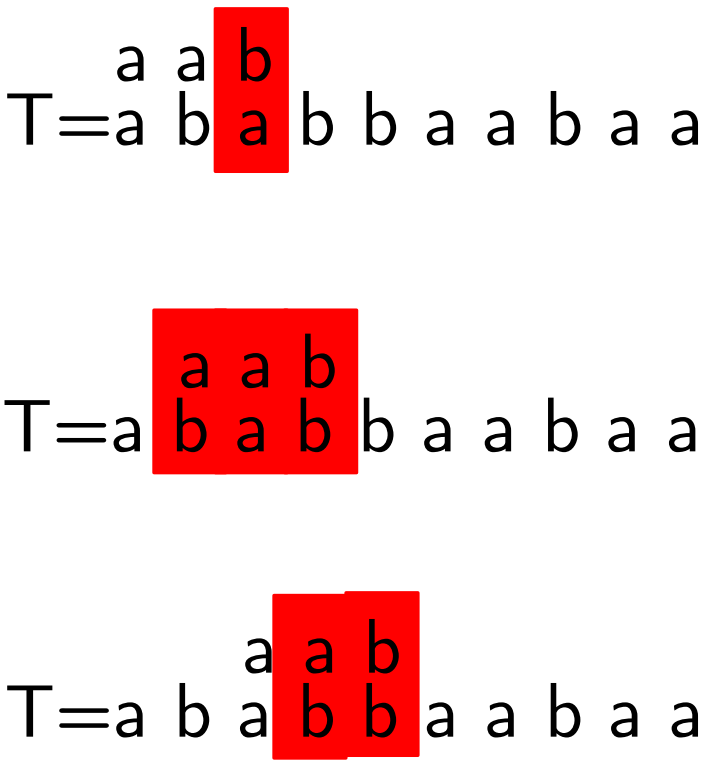
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



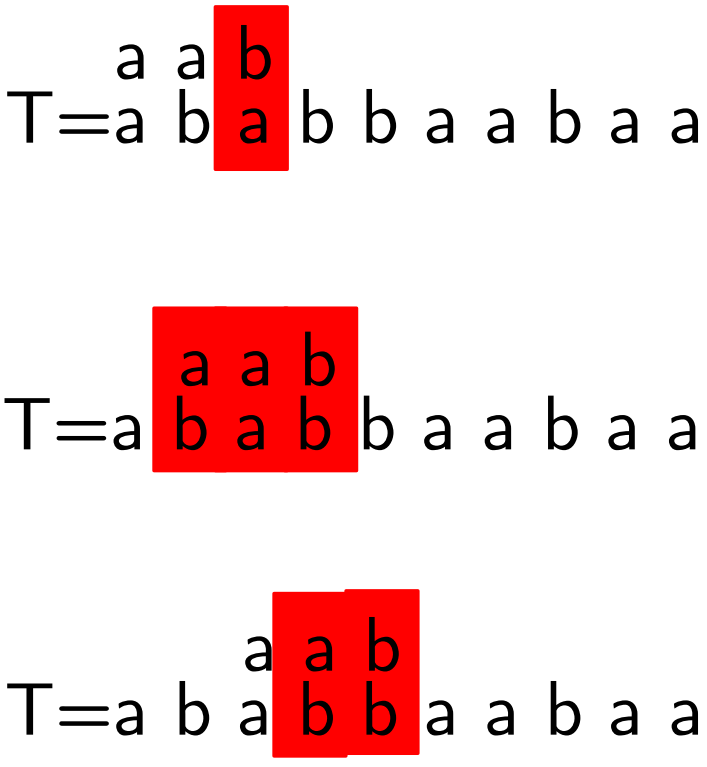
الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



الگوریتم‌های تطابق الگو (Pattern Matching)

مثال : T=a b a b b a a b a a P=a a b



سرعت پردازش در این روش به‌طور متوسط نسبت به روش ترتیبی بیشتر است با این حال در بدترین حالت زمان اجرا هنوز O(n.m) است.

الگوریتم‌های تطابق الگو (Pattern Matching)

- **الگوریتم سوم تطابق الگو**

به‌صورت ایده‌ال الگوریتمی مورد قبول است که در زمان:

O(strlen(string)+strlen(substring)) یعنی (طول الگو + طول رشته) O

کار کند. این زمان بهترین حالت برای این مسئله می‌باشد.

در بدترین حالت باید حداقل یکبار تمام کاراکترهای T و P را تست نمود.

می‌خواهیم رشته‌ای را برای یافتن یک الگو جستجو کنیم، بدون اینکه به عقب برگردیم.

بنابراین اگر یک عدم تطابق رخ دهد، می‌خواهیم که از اطلاعات خود در مورد کاراکترها و موقعیت آن در الگو یعنی محلی که عدم تطبیق روی داده است، برای تعیین جایی که باید جستجو را ادامه دهیم، استفاده کنیم.

پایان فصل سوم