ساختمان داده‌ها

دکتر امین گلزاری اسکوئی

a.golzari@azaruniv.ac.ir

a.golzari@tabrizu.ac.ir

https://github.com/Amin-Golzari-Oskouei

دانشگاه شهید مدنی آذربایجان

پاییز ۱۴۰۲

فصل ۳
آرایه

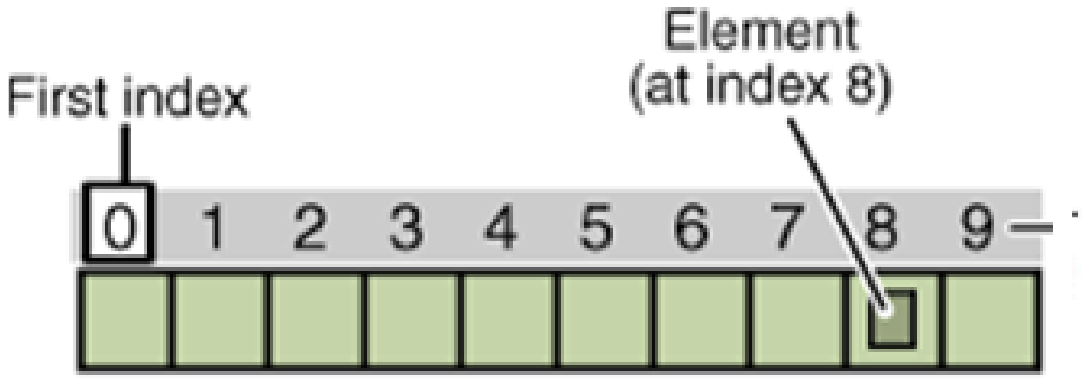
مطالب این فصل

- تعریف آرایه
- نحوه ذخیره‌سازی آرایه در رایانه

تعریف آرایه

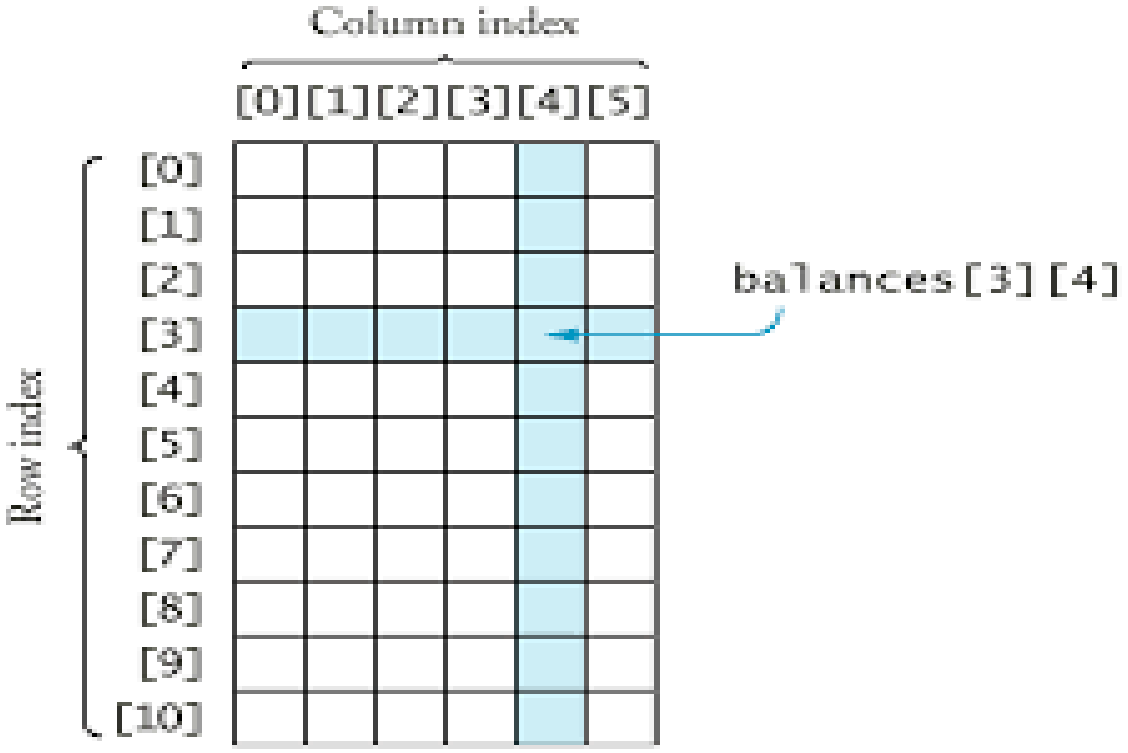
آرایه مجموعه‌ای از داده‌های هم نوع است که تحت یک نام معرفی شده و برای دسترسی به هر عنصر آن از اندیس مشخصی استفاده می‌شود.

```
int  a[10];
```



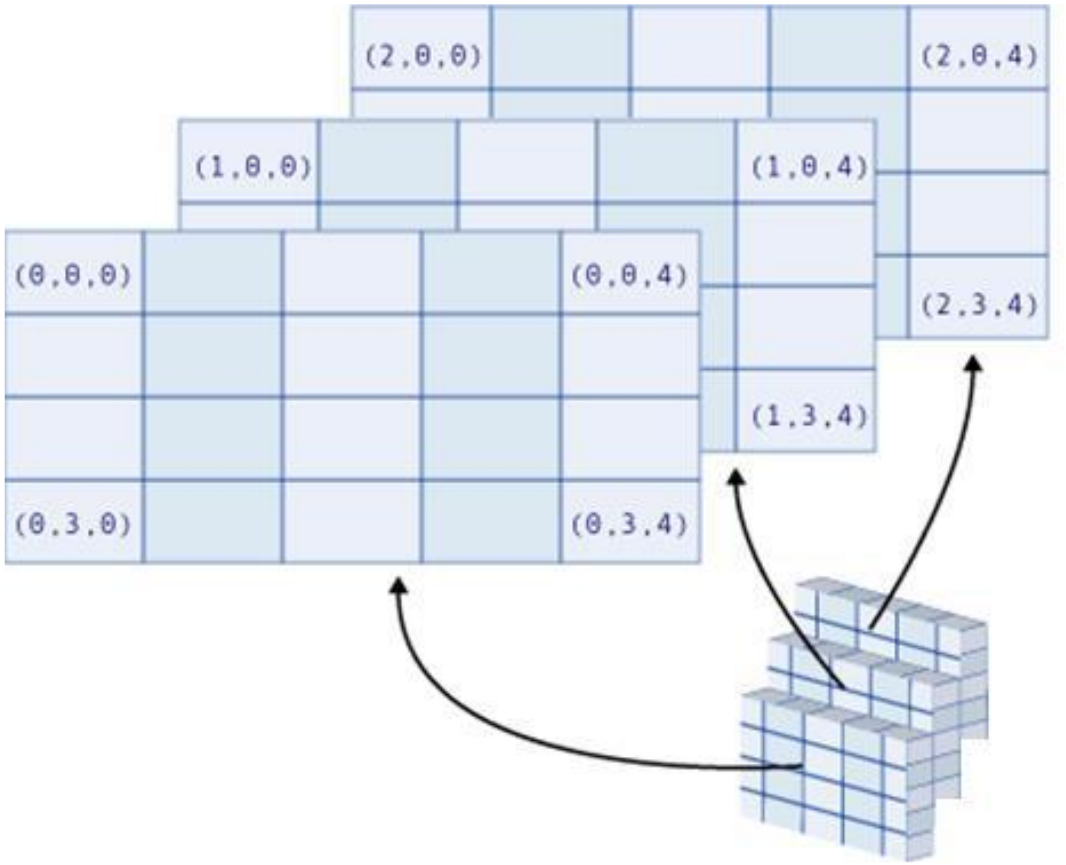
آرایه دو بعدی

```
int  balances[11][6];
```



آرایه ۳ بعدی

```
int  c [3][4][5];
```



نحوه ذخیره عناصر آرایه در حافظه

عناصر آرایه در حافظه به صورت پشت سر هم قرار می گیرند که موجب سریع شدن سرعت دسترسی به عناصر آرایه می شود. با فرض اینکه عنصر اول آرایه در آدرس آلفا حافظه ذخیره شود و هر عنصر آرایه به اندازه W بایت فضا اشغال نماید، محل هر عنصر آرایه در حافظه به کمک روابط زیر محاسبه می شود. فرض شده که عناصر آرایه به صورت سطر به سطر در حافظه ذخیره شده است:

$x[l..u]$

$loc(x[i]): \quad \alpha + (i-l) \times w$

$x[l_1..u_1, l_2..u_2]$

$loc(x[i,j]): \quad \alpha + [(i-l_1) \times (u_2 - l_2 + 1) + (j - l_2)] \times w$

$x[l_1..u_1, l_2..u_2, l_3..u_3]$

$loc(x[i,j,k]):$

$\alpha + [(i-l_1) \times (u_2 - l_2 + 1) \times (u_3 - l_3 + 1) + (j - l_2) \times (u_3 - l_3 + 1) + (k - l_3)] \times w$

مثال

آدرس عنصر A[1][2] در آرایه A[3][4] را محاسبه کنید.

فرض :

عناصر آرایه از نوع داده ۸ بایتی است .

آدرس شروع آرایه ۱۰ است.

حل:

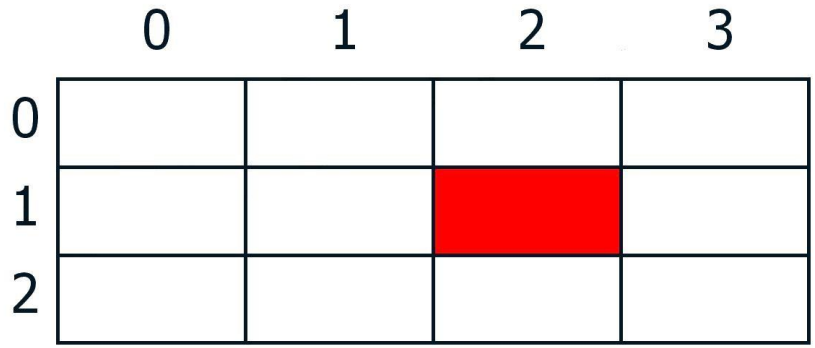
محدوده اندیس ها به صورت A[0..2][0..3] باشد.

$$\alpha + [(i - l_1) \times (u_2 - l_2 + 1) + (j - l_2)] \times w$$

$$10 + [(1 - 0) \times 4 + (2 - 0)] \times 8 = 58$$

مثال

A[3][4]



$$10 + [(1-0) \times 4 + (2-0)] \times 8 = 58$$

مثال

آدرس عنصر A[3][4][2] در آرایه A[20][10][5] را محاسبه کنید.

فرض :

عناصر آرایه از نوع داده ۲بایتی است .

آدرس شروع آرایه صفر است.

حل:

محدوده اندیس ها به صورت A[0..19][0..9][0..4] باشد.

$$\alpha + [(i - l_1) \times (u_2 - l_2 + 1) \times (u_3 - l_3 + 1) + (j - l_2) \times (u_3 - l_3 + 1) + (k - l_3)] \times w$$

$$0 + [(3 - 0) \times 10 \times 5 + (4 - 0) \times 5 + (2 - 0)] \times 2$$

$$= 0 + [150 + 20 + 2] \times 2 = 344$$

جستجوی خطی در آرایه

تابع زیر مقدار x را در آرایه n عنصری a، به روش مقایسه با تک تک عناصر آرایه، جستجو می‌نماید. در صورت پیدا کردن، اندیس خانه حاوی x و در صورت پیدا نکردن، عدد -1 را بر می گرداند.

```
seqsearch (a[ ] , n , x )
{
    for ( i = 0 ; i <=n-1 ; i++ )
          if (a[i] == x)
              return  i ;
  return  -1 ;
}
```

در یک جستجوی ناموفق نیاز به n+1 عمل مقایسه داریم که در نتیجه زمان آن O(n) خواهد بود.

جستجوی دودویی

با فرض اینکه آرایه به طور صعودی مرتب شده باشد، عنصر مورد جستجو با عنصر وسط آرایه مقایسه می شود،
در صورت برابر بودن، پیدا شده است .
در غیر اینصورت، اگر از عنصر وسط آرایه بزرگتر باشد، مقایسه به طور بازگشتی در نیمه بالایی آرایه انجام می گیرد
و در صورت کوچکتر بودن از عنصر وسط، مقایسه به طور بازگشتی در نیمه پایینی آرایه انجام می شود.

مثال

مثال: پیدا کردن عدد ۱۲ در آرایه مرتب:

۱- مقایسه ۱۲ با عنصر وسط x[1..9] یعنی ۳۵

۲- مقایسه ۱۲ با عنصر وسط x[1..4] یعنی ۹ .

۳- مقایسه ۱۲ با عنصر وسطx[3..4] یعنی ۱۲.

بنابراین با ۳ مقایسه عدد ۱۲ را پیدا کردیم.

تابع جستجوی دودویی

تابع زیر مقدار x را در آرایه n عنصری مرتب A، به روش دودویی، جستجو می‌نماید. اگر x را پیدا کند، اندیس آن را در آرایه بر می گرداند و در صورت پیدا نکردن، عدد -1 را بر می گرداند.(در ابتدا : low=0 , high=n-1)

```
bsearch(a[ ] , x , low , high){
    while (low <= high)
    {
      mid = (low + high) / 2 ;
      if ( x < a[mid] )
          high = mid-1 ;
      else
          if ( x > a[mid] )
                low = mid+1 ;
          else   return  mid ;
    }
    return  -1 ;
}
```

جستجوی دودویی(بازگشتی)

```
bsearch (a[ ] , x , low , high ){
    if (low <=high )
    {
        mid = ( low+high ) / 2;
        if ( x < a[mid] )
            bsearch( a , x , low , mid-1 );
        else  if ( x > a[mid] )
                    bsearch (a , x , mid+1 , high );
                else
                    return  mid;
      }
   return  -1; }
```

رابطه بازگشتی : $T(n) = T(\frac{n}{2}) + 1$ جواب رابطه : $\lfloor \lg n \rfloor + 1$ مرتبه : $O(\lg n)$

جستجوی دودویی

حداکثر تعداد مقایسه ها برای پیدا کردن عنصری به روش جستجوی دودویی در یک آرایه :

با هزار عنصر : 10

با ده هزار عنصر : 14

با صد هزار عنصر : 17

با یک میلیون عنصر : 20

بنابراین برتری جستجوی دودویی به جستجوی خطی ، در آرایه با تعداد عناصر زیاد، بیشتر خود را نشان می دهد.

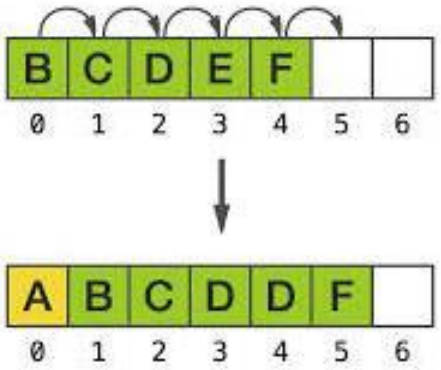
درج در آرایه

تابع insert ، مقدار x را در مکان k ام آرایه a، اضافه می‌کند.

فرض شده که آرایه a دارای n خانه است که m عنصر آن پر است : (m<n)

مثال: X='A',k=0,n=7,m=5

```
insert (a[ ] , m , k , x)
{
  for ( i = m-1 ; i >= k ; i-- )
       a[i+1] = a[i];
  a[k] = x;
}
```



تعداد شیفت مورد نیاز : $m - k$

حذف از آرایه

تابع زیر k امین عنصر آرایه a را حذف کرده و آن را در متغیر x قرار می‌دهد. (k<=n)

```
delete( a[ ] , n , k , x )
{
    x = a[k];
   for ( i = k ; i<n ; i++ )
       a[i] = a[i+1];
   a[i] = 0;
   return(x);
}
```

ماتریس

ماتریس های معروف عبارتند از:

۱- ماتریس اسپارس

۲- ماتریس مثلثی (پایین مثلثی و بالا مثلثی)

۳- ماتریس قطری (سه قطری ، پنج قطری و ...)

ماتریس اسپارس (خلوت)

ماتریسی که دارای تعداد نسبتا زیادی عنصر صفر باشد را ماتریس اسپارس(خلوت یا تنک)می نامند.
در این ماتریس، برای کاهش حافظه مصرفی و زمان اجرا، فقط عناصر غیرصفر ماتریس ذخیره می‌شوند.

مثال:

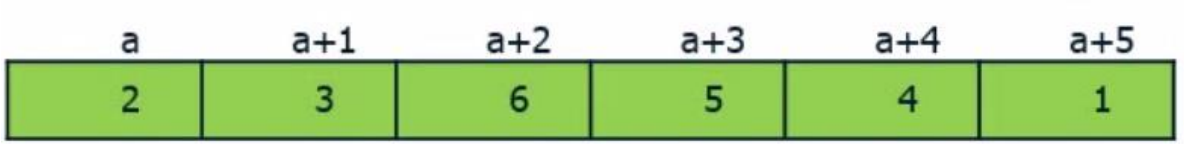
$$\begin{pmatrix} 0 & 0 & 2 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 3 & 4 & 2 \\ 1 & 3 & 2 \\ 2 & 2 & 5 \end{pmatrix}$$

ماتریس پایین مثلثی

ماتریسی که تمام عناصر بالای قطر اصلی آن صفر باشد را ماتریس پایین مثلثی می گویند. برای ذخیره این ماتریس، فقط عناصر غیر صفر ذخیره می شوند.

مثال:

$$\begin{bmatrix} 2 & 0 & 0 \\ 3 & 6 & 0 \\ 5 & 4 & 1 \end{bmatrix}$$



عنصر واقع در سطر i و ستون j ماتریس مثلثی در آدرس زیر ذخیره می شود:

$$a + \frac{i(i-1)}{2} + j - 1$$

ماتریس ۳ قطری

ماتریس سه قطری یک ماتریس مربعی می باشد که درایه های غیر صفر آن روی قطر اصلی و بلافاصله بالا و پائین قطر اصلی ظاهر می شوند. تعداد عناصر غیر صفر در این ماتریس برابر می باشد.

مثال:

$$\begin{bmatrix} 6 & 23 & 0 & 0 \\ 1 & 3 & 7 & 0 \\ 0 & 2 & 4 & 83 \\ 0 & 0 & 9 & 5 \end{bmatrix}$$

a	a+1	a+2	a+3	a+4	a+5	a+6	a+7	a+8	a+9
6	23	1	3	7	2	4	83	9	5

عنصر واقع در سطر i و ستون j ماتریس پایین مثلثی در آدرس حافظه زیر ذخیره می شود:

$$a + 2i + j - 3$$

تمرین

۱- ماتریس بالا مثلثی ماتریس مربعی است که عناصر پایین قطر اصلی آن برابر صفر هستند. اگر عناصر غیر صفر را به صورت ستونی در حافظه ذخیره کنیم، محل ذخیره عنصر واقع در سطر i و ستون j این ماتریس در حافظه را مشخص کنید.

۲- ماتریس پایین مثلثی ماتریس مربعی است که عناصر بالای قطر اصلی آن برابر صفر هستند. اگر عناصر غیر صفر را به صورت قطری با شروع از قطر اصلی در حافظه ذخیره کنیم، محل ذخیره عنصر واقع در سطر i و ستون j این ماتریس در حافظه را مشخص کنید.

۳- ماتریس پنج قطری ماتریس مربعی است که به جز عناصر قطر اصلی و دو قطر بالای آن و دو قطر پایین آن، سایر عناصر برابر صفر می باشند. اگر عناصر غیر صفر را به صورت سطری در حافظه ذخیره نماییم، محل ذخیره عنصر واقع در سطر i و ستون j این ماتریس در حافظه را مشخص کنید.

تشکر

سوال؟

a.golzari@azaruniv.ac.ir
a.golzari@tabrizu.ac.ir
https://github.com/Amin-Golzari-Oskouei