به نام خدا

ساختمان داده ها و الگوریتم ها

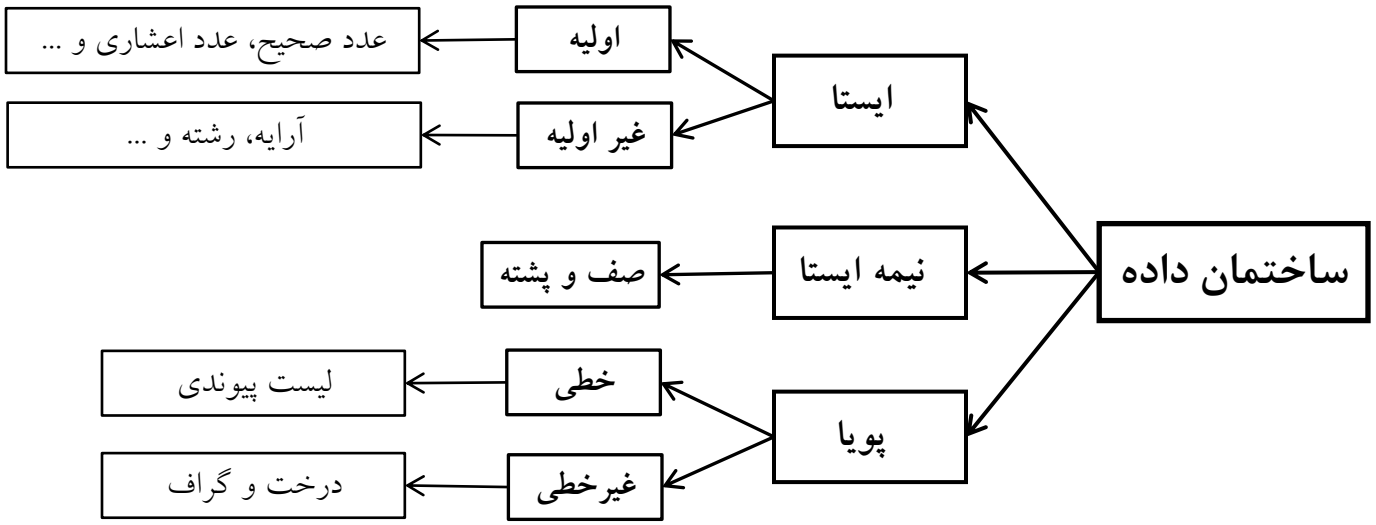
محمدمهدی گیلانیان صادقی

دانشگاه آزاد اسلامی واحد قزوین

نیمسال دوم ۱۴۰۱-۱۴۰۲

ساختمان داده ها: شیوه قرار گرفتن داده ها در حافظه کامپیوتر یا دیسک، ساختمان داده نام دارد.

برنامه = الگوریتم ها + ساختمان داده ها



الگوریتم:

مجموعه محدودی از دستورالعمل ها که اگر دنبال شوند، موجب انجام کار خاصی میشوند.

ویژگیهای الگوریتم:

ورودی: یک الگوریتم می تواند هیچ یا چندین کمیت ورودی داشته باشد.

خروجی: یک الگوریتم باید حداقل یک کمیت به عنوان خروجی داشته باشد.

قطعیت: هر دستورالعمل باید بدون ابهام و کاملا واضح باشد.

محدودیت: یک الگوریتم باید پس از طی مراحل محدودی خاتمه یابد.

کارایی: هر دستورالعمل باید به گونه ای باشد که با استفاده از قلم و کاغذ بتوان آن را اجرا کرد. به عبارت دیگر هر دستورالعمل باید انجام پذیر باشد.

تحلیل الگوریتم:

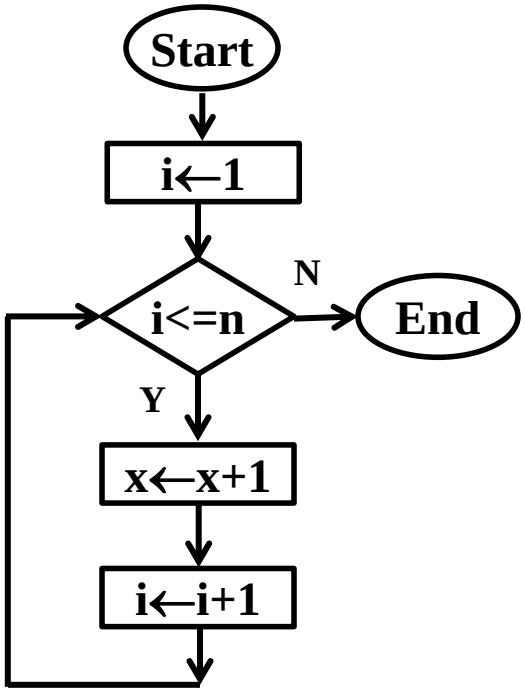
- زمان مصرفی
- حافظه مصرفی

معیارهای مقایسه الگوریتم ها از نظر زمان مصرفی:

- زمان اجرای الگوریتم
- تعداد تکرار دستورات الگوریتم
- تعداد تکرار دستورات اصلی الگوریتم

زمان اجرای الگوریتم:

```
for (i=1; i<=n; i++)
    x ++;
```



زمان اجرا	تعداد تکرار	دستورات
t1	1	S1
t2	n+1	S2
t3	n	S3
t4	n	S4

$$T = \sum_{i=1}^{k} fi * ti$$

K=4

تعداد تکرار دستورات الگوریتم:

```
for (i=1; i<=n; i++)
        x ++;
```

دستورات	تعداد تکرار
S1	1
S2	n+1
S3	n
S4	n

$$FC = \sum_{i=1}^{k} fi$$

K=4

تعداد تکرار دستورات اصلی الگوریتم (O):

```
for(i=1; i<=n; i++)
        x++;
```

تعداد تکرار دستورات اصلی الگوریتم (O):

O(n)= زمان اجرا

```
for(i=1; i<=n; i++)
        x++;
```

```
for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
        x++;
```

تعداد تکرار دستورات اصلی الگوریتم (O):

زمان اجرا =O(n)

```
for(i=1; i<=n; i++)
        x++;
```

زمان اجرا =O(n^2)

```
for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
        x++;
```

تمرین۱ : زمان اجرای الگوریتم (O) را بدست آورید؟

ب)

```
for(i=1; i<=n; i++)
    for(j=1; j<=i; j++)
        x++;
```

الف)

```
for(i=1; i<=n; i++)
    for(j=i; j<=n; j++)
        x++;
```

آرایه ها:

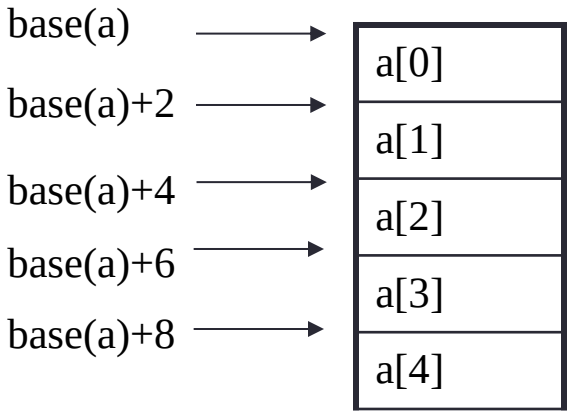
آرایه نوعی ساختمان داده است که عناصر آن هم نوع بوده و هریک از عناصر بـه صـورت مستقیم قابل دستیابی است. آرایه می تواند یک بعدی، دو بعدی و یا چند بعدی باشد.

آرایه یک بعدی:

int a[5];

base(a) + i *size

آدرس عنصر i ام:



مثال: در آرایه float a[100]، اگر آدرس شروع آرایه در حافظه ۱۰۰ باشد، آنگاه عنصر a[20] در کدام آدرس قرار دارد؟

base(a) + i *size

آدرس عنصر (i=20) ام در حافظه: ۱۰۰+۲۰*۴ =۱۸۰

آرایه دوبعدی/ماتریس:

int a[2][3]; m=2, n=3

2	1	4
7	5	3

2x3

آرایه های دوبعدی یا ماتریس ها به دو روش در حافظه ذخیره می شوند:

۱) روش سطری

۲) روش ستونی

2	**1**	**4**
7	**5**	**3**

۱) روش سطری

آدرس عنصر a[i][j] :

$$base(a) + (i \times n + j) \times size$$

a[0][0]	**2**
a[0][1]	**1**
a[0][2]	**4**
a[1][0]	**7**
a[1][1]	**5**
a[1][2]	**3**

۲) روش ستونی

آدرس عنصر a[i][j] :

$$base(a) + (j \times m + i) \times size$$

a[0][0]	**2**
a[1][0]	**7**
a[0][1]	**1**
a[1][1]	**5**
a[0][2]	**4**
a[1][2]	**3**

مثال: در آرایه دوبعدی int a[2][3]، اگر آدرس شروع آرایه در حافظه ۱۰۰۰ باشد، آنگاه عنصر a[1][2] به روش سطری و ستونی در کدام آدرس قرار دارد؟

2	1	4
7	5	3

روش سطری:

آدرس عنصر a[1][2] = 1000+ (1 × 3 + 2) × 2=1010

روش ستونی:

آدرس عنصر a[1][2] = 1000+ (2 × 2 + 1) × 2 = 1010

تمرین۲ : مطلوبست آدرس عنصر a[1][1] برای مثال بالا؟

ماتریس اسپارس/ خلوت:

ماتریسی که عناصر صفر آن زیاد باشد را ماتریس اسپارس می نامند.

int A[5][4];

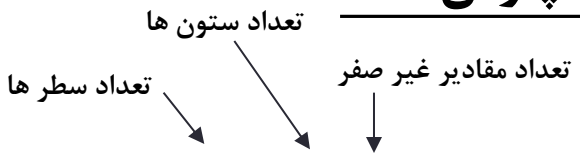
2	0	0	0
0	0	0	0
0	0	0	7
0	0	15	0
0	0	0	10

5x4

نمایش ماتریس اسپارس:

- ✓ اگر یک ماتریس اسپارس شامل n عنصر غیر صفر باشد برای نمایش آن از یک آرایه دو بعدی (A) که دارای n+1 سطر و ۳ ستون است استفاده می شود.
- ✓ عناصر سطر اول به ترتیب، تعداد سطر، تعداد ستون و تعداد عناصر غیر صفر آرایه دو بعدی/ ماتریس A را نشان می دهد.
- ✓ در سطر های بعدی به ترتیب شماره سطر، شماره ستون و مقدار عناصر غیر صفر قرار می گیرد.
- ✓ در ستون اول شماره سطرها به صورت صعودی قرار می گیرد.

نمایش ماتریس اسپارس (۲):



5	4	4
0	0	2
2	3	7
3	2	15
4	3	10

5x3

نمایش ماتریس اسپارسAS

2	0	0	0
0	0	0	0
0	0	0	7
0	0	15	0
0	0	0	10

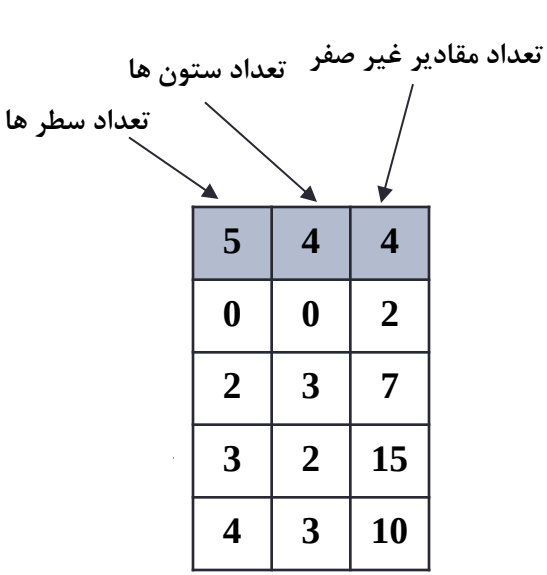
5x4

ماتریس اسپارسA

فضای حافظه A = 5 * 4 * 2 = 40 B

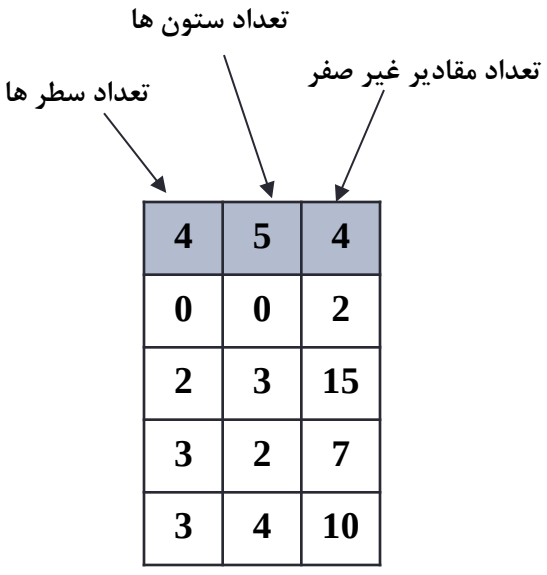
فضای حافظه AS = 5 * 3 * 2 = 30 B

ترانهاده ماتریس اسپارس:



5	4	4
0	0	2
2	3	7
3	2	15
4	3	10

AS



4	5	4
0	0	2
2	3	15
3	2	7
3	4	10

AS^t

ترانهاده ماتریس اسپارس(۲):

```
void transpose( int bs[][3], int bt[][3])
{
   int i, j, k;
   bt[0][0]=bs[0][1]; bt[0][1]=bs[0][0]; bt[0][2]=bs[0][2];
   k=1;
   for(i=0;i< bs[0][1];i++)
     for(j=1; j< bs[0][2]; j++ )
        if(i==bs[j][1])
           {
            bt[k][0]=bs[j][1];
            bt[k][1]=bs[j][0];
            bt[k][2]=bs[j][2];
            k++;
         }
}
```

جمع دو ماتریس اسپارس:

4	5	4
0	0	2
1	0	4
1	3	1
3	2	2

5x3

+

4	5	5
0	3	5
0	4	1
1	3	-1
2	3	5
3	2	3

6x3

=

4	5	6
0	0	2
0	3	5
0	4	1
1	0	4
2	3	5
3	2	5

7x3

ماتریس های بالا مثلثی و پایین مثلثی:

ماتریس پایین مثلثی:

روش سطری

2	3	7	6	1	1	7	3	0	5

روش ستونی

2	3	6	7	7	1	3	1	0	5

2	0	0	0
3	7	0	0
6	1	1	0
7	3	0	5

ماتریس بالا مثلثی:

روش سطری

2	7	8	9	7	1	0	1	3	5

روش ستونی

2	7	7	8	1	1	9	0	3	5

2	7	8	9
0	7	1	0
0	0	1	3
0	0	0	5

تمرین۳ : رابطه های نگاشت ماتریس های بالا مثلثی و پایین مثلثی را در یک آرایه یک بعدی به روش های سطری و ستونی محاسبه کنید؟

تمرین۴ : برنامه ای بنویسید که دو ماتریس اسپارس را با هم جمع نماید؟

تمرین۵ : آدرس عنصر را در آرایه های سه بعدی و n بعدی به روش سطری و ستونی به دست آورید؟

پایان