Data Structure

Dr. Rastgoo

فصل ۳:

آرایه

مقدمه

داده ها می توانند به روشهای متفاوتی سازماندهی شوند. مدل منطقی یا ریاضی سازماندهی داده ها به یک صورت خاص، یک ساختمان داده نامیده می شود. انواع ساختمان داده ها عبارتند از:

۱- ایستا (مانند آرایه و رکورد)

۲- نیمه ایستا (مانند صف و پشته)

۳- پویا (مانند لیست پیوندی، درخت و گراف)

ساختار ساختمان داده‌های ایستا در طول حیات آنها ثابت است ولی در نوع پویا تغییرات مجاز و نامحدود است.

عملیات رایج در ساختمان داده ها عبارتند از: اضافه کردن (insert) ، حذف داده (delete) ، جستجوی داده (search) و پیمایش.

آرایه

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

مثال

در زیر نحوه تعریف آرایه های یک بعدی و دو بعدی در زبان پاسکال آورده شده است:

الف- تعریف آرایه ای یک بعدی به نام a شامل 3 عنصر(اندیسهای 2 تا 4) از نوع صحیح:

```
a : array [2..4] of integer ;
```

ب- تعریف آرایه ای دو بعدی به نام b شامل 6 عنصر (2 سطر و3 ستون) از نوع کاراکتری:

```
b : array [1..2,4..6] of char ;
```

مثال

در زیر نحوه تعریف آرایه های یک بعدی و دو بعدی در زبان c آورده شده است:

الف- تعریف آرایه ای یک بعدی به نام c شامل 3 عنصر(اندیس های 0 تا 2) از نوع اعشاری: float c[3];

ب- تعریف آرایه ای دو بعدی به نام d شامل 12 عنصر (3 سطر و4 ستون) از نوع صحیح: int d[3][4];

مثال

تعداد عناصر آرایه چهار بعدی A[2..5,3..8,1..3,3..4] را بدست آورید؟

حل:

$$(5-2+1)(8-3+1)(3-1+1)(4-3+1) = 4\times 6\times 3\times 2 = 144$$

تعداد عناصر آرایه n بعدی به شکل $A[l_1..u_1, l_2..u_2, ..., l_n..u_n]$ برابر است با: $\prod_{i=1}^{n}(u_i - l_i + 1)$

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 α 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:

۱- آدرس عنصر A[i] **در آرایه یک بعدی** $A[l..u]$:

$$\alpha + (i - l) \times w$$

۲- آدرس عنصر A[i,j] **در آرایه دو بعدی** $A[l_1..u_1, l_2..u_2]$:

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

ستونی :
$$\alpha + [(i - l_1) + (j - l_2) \times (u_1 - l_1 + 1)] \times w$$

۳- آدرس عنصر A[i,j,k] **در آرایه سه بعدی** $A[l_1..u_1, l_2..u_2, l_3..u_3]$:

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

ستونی $\alpha + [(i-l_1)+(j-l_2)\times(u_1-l_1+1)+(k-l_3)\times(u_1-l_1+1)\times(u_2-l_2+1)]\times w$

۴- آدرس عنصر A[i , j, k, t] **در آرایه چهار بعدی** $A[l_1..u_1, l_2..u_2, l_3..u_3, l_4..u_4]$:

(فرض : $d_i = u_i - l_i + 1$)

سطری :

$$\alpha + [(i - l_1) \times d_2 \times d_3 \times d_4 + (j - l_2) \times d_3 \times d_4 + (k - l_3) \times d_4 + (t - l_4)] \times w$$

ستونی :

$$\alpha + [(i - l_1) + (j - l_2) \times d_1 + (k - l_3) \times d_1 \times d_2 + (t - l_4) \times d_1 \times d_2 \times d_3] \times w$$

مثال

مطلوب است آدرس عنصر x[7] در حافظه؟ (عنصر اول آرایه در آدرس 1000 حافظه قرار دارد)

```
x : array [-2..30] of  Real;
```

حل: متغیر از نوع real در پاسکال ، 6 بایتی است.

$1000+[\ 7-(-2)] \times 6=1054$

مثال

آرایه دو بعدی x با 3 سطر و 4 ستون مفروض است. اگر خانه اول آرایه در محل 10 حافظه ذخیره شود. محـل عنصـر x[2,3] در حافظه کدام است؟(w=1)

x : array [1..3,1..4] of char ;

حل:

سطری : $10+[(2-1)(4-1+1)+(3-1)]\times 1=16$

ستونی : $10+[(2-1)+(3-1)(3-1+1)]\times 1=17$

مثال

آرایه سه بعدی A(1..11, 2..13 ,3..14) از آدرس 100 حافظه قرار گرفته می‌باشد. آدرس عنصر A(9,7,6) چه خواهد بـود؟ (w=1)

حل:

سطری : $100+[(9-1)\times(13-2+1)\times(14-3+1)+(7-2)\times(14-3+1)+(6-3)]\times 1=1315$

ستونی : $100+[(9-1)+(7-2)\times(11-1+1)+(6-3)\times(13-2+1)\times(11-1+1)]\times 1=559$

مثال

آرایه چهار بعدی A[1..10,1..20,1..10,1..20] را در نظر بگیرید که به صورت سطری ذخیره شده است. اگر آدرس شروع آرایه صفر باشد،آدرس عنصر A[1,2,3,4] کدام است؟ (w=1)

حل:

$$0+[\ (1-1)\times 20\times 10\times 20 + (2-1)\times 10\times 20 + (3-1)\times 20 + (4-1)]\times 1 = 243$$

مثال

اگر آرایه ی x(-2:3 ,-4:-2 ,'A':'F', 0:3) با طول داده ای **2** و از آدرس **100** در حافظه ذخیـره شـده باشـد، LOC(x[-2,-2,'D',0]) کدام است؟

$$100+[(-2+2)\times 3\times 6\times 4+(-2+4)\times 6\times 4+('D'-'A')\times 4+(0-0)]\times 2=220$$

جستجو در آرایه

به دو روش خطی و دودویی می توان در یک آرایه جستجو کرد.

جستجوی خطی

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.

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

میانگین تعداد عمل مقایسه در یک جستجوی موفقیت آمیز برابر $\frac{n+1}{2}$ می باشد.

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

جستجوی دودویی

اگر عناصر یک آرایه مرتب شده باشند، بهتر است از روش جستجوی دودویی استفاده کرد. در این روش با فرض اینکه آرایه به طور صعودی مرتب شده باشد، عنصر مورد جستجو با عنصروسط آرایه مقایسه می شود، در صورت برابر بودن، پیدا شده است و 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.

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

```
int  bsearch( int a[ ] , int  x , int  low , int  high){
   int  mid ;
   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;
}
```

پیاده سازی بازگشتی جستجوی دودویی

```
int  bsearch (int  a[ ] , int  x , int  low , int  high){
   int  mid;
   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 تابع بالا $T(n) = T(\frac{n}{2}) + 1$ بوده که از مرتبه $O(\lg n)$ است.

مثال

برای پیدا کردن عدد 9 در آرایه مرتب زیر با روش جستجوی دودویی، به چند مقایسه نیاز است؟

1	2	3	4	5	6	7	8	9
5	**9**	12	20	**35**	50	82	88	97

حل:

ابتدا عدد 9 با عنصر وسط آرایه یعنی 35 مقایسه می شود و چون از آن کوچکتر است مقایسه به طور بازگشتی در زیر آرایه x[1..4] انجام می گیرد. بنابراین مقایسه بعدی با عنصر وسط این آرایه یعنی 9 است که برابر است. بنابراین با دو مقایسه به نتیجه می رسیم.

حداکثر تعداد مقایسه‌ها برای پیدا کردن یک عنصر در آرایه مرتب n عنصری به روش جستجوی دودویی برابر $\lfloor \lg n \rfloor + 1$ می باشد.

مثال

حداکثر چند عمل مقایسه برای پیدا کردن یک عدد به روش جستجوی دودویی در آرایه مرتب هزار عنصری انجام می‌گیرد؟

حل: کافی است در رابطه $\lfloor \lg n \rfloor + 1$ به جای n ، عدد 1000 قرار دهیم که حاصل برابر 10 خواهد شد.

البته می‌توان به جای عدد هزار، از عدد توان دو و بزرگتر از آن یعنی 1024 استفاده کرده و از یک صرف نظر کرد.

حداکثر تعداد مقایسه ها برای پیدا کردن عنصری به روش جستجوی دودویی در یک آرایه با هـزار عنصـر برابـر 10 ، در آرایه با ده هزار عنصر برابر 14، در آرایه با صد هزار عنصر برابر 17 و در آرایه با یک میلیون عنصـر برابـر 20 مـی باشـد. بنابراین برتری جستجوی دودویی به جستجوی خطی در آرایه با تعداد عناصر زیاد بیشتر خود را نشان می دهد.

مثال

میانگین تعداد مقایسه ها برای یافتن یک عنصر دلخواه در آرایه مرتب شده با 10 عنصر با استفاده از جستجوی دودویی کـدام است؟

حل: در شکل زیر تعداد مقایسه های لازم برای هر عنصر در زیر آن نوشته شده است:

1	2	3	4	5	6	7	8	9	10
12	20	26	30	45	65	68	70	75	92
3	2	3	4	1	3	4	2	3	4

به طور نمونه برای پیدا کردن عدد 45 نیاز به یک مقایسه است، چون در وسط آرایه قرار دارد. همچنین برای پیدا کردن عـدد 20 نیاز به 2 مقایسه می باشد، چون در ابتدا با مقدار 45 مقایسه شده و چون از آن کوچکتر است با وسط آرایـه پـایین یعنـی 20 مقایسه می شود. برای محاسبه میانگین تعداد مقایسه ها کافی است مجموع این اعداد را بر تعداد آنها تقسیم کـرد، یعنـی:

$$\frac{29}{10}$$

روش دوم: می توان با اعداد داده شده یک درخت دودویی کامل ساخت و سپس مجموع حاصلضرب تعداد گره ها در یک سطح و سطح آن گره ها را محاسبه کرد و در نهایت حاصل را به تعداد اعداد تقسیم کرد. به طور نمونه برای آرایه 10 عنصری بالا داریم:

$$\frac{1\times1+2\times2+4\times3+3\times4}{10}=\frac{29}{10}$$

مثال

میانگین تعداد مقایسه ها برای پیدا کردن یک عنصر معلوم به روش جستجوی دودویی در یک آرایه 200 عنصری مرتب کدام است؟

حل: با فرض اینکه N_K ، تعداد عناصری در آرایه باشد که برای تعیین مکان آن به K مقایسه نیاز است ، داریم:

K	1	2	3	4	5	6	7	8
N_K	1	2	4	8	16	32	64	73

به طور نمونه ، 8 عنصر در آرایه 200 عنصری وجود دارد که برای پیدا کردن آنها به 4 مقایسه نیاز است.

تذکر: عدد 73 از رابطه مقابل بدست می آید:

200-(1+2+4+8+16+32+64)=73

بنابراین میانگین تعداد مقایسه ها برابر است با:

$$g(n) = \frac{1}{n}\sum_{k=1}^{8} k.n_k = \frac{1.1 + 2.2 + 3.4 + 4.8 + 5.16 + 6.32 + 7.64 + 8.73}{200} = \frac{1353}{200} = 6.765$$

مثال

میانگین تعداد مقایسه ها برای جستجوی موفق (S) و همچنین میانگین تعداد مقایسه ها در جستجوی ناموفق (U) را برای آرایه 4 عنصری مرتب زیر بدست آورید.

حل:

تعداد مقایسه ها در جستجوی موفق برای هر یک از عناصر در زیر آن آورده شده است:

5	10	15	20
2	1	2	3

$$S = \frac{2+1+2+3}{4} = \frac{8}{4}$$

تعداد مقایسه ها در جستجوی ناموفق برای هر 5 حالت ممکن در شکل زیر آورده شده است:

5	10	15	20
2 2	2	3	3

به طور نمونه برای جستجوی عدد 4 که در آرایه نیست(جستجوی ناموفق)، ابتدا مقایسه با وسط آرایه یعنی عدد 10 انجام شده و سپس با عدد 5 مقایسه انجام می شود. یعنی با 2 مقایسه متوجه می شویم که عدد 4 در آرایه نمی باشد. همچنین برای جستجوی عددی مابین 10,5 مانند عدد 7 ، با 2 مقایسه مشخص می شود که در آرایه نیست و یا برای جستجوی عددی بزرگتر از 30 ، با 3 مقایسه مشخص می شود که در آرایه نمی باشد.

بنابراین میانگین تعداد مقایسه ها در جستجوی ناموفق برابر است با:

$$U = \frac{2+2+2+3+3}{5} = \frac{12}{5}$$

بین S و U رابطه $S = (1+\frac{1}{n})U - 1$ برقرار می باشد. که مـی تـوان آن را بـه صـورت $U = (\frac{n}{n+1})(S+1)$ نیـز نوشت.

الگوریتم جستجوی سه تایی(ترنری)

در جستجوی سه تایی، آرایه باید به سه قسمت تقسیم شود. در هر تکرار نقطه $\frac{1}{3}$ که با m_1 و نقطه $\frac{2}{3}$ که با m_2 نشان داده می شود، به صورت زیر محاسبه می گردد:

(L حد پایین و h حد بالا)

$$m_1 = L + \frac{1}{3}(h - L) = \frac{3L + h - L}{3} = \frac{h + 2L}{3}$$

$$m_2 = L + \frac{2}{3}(h - L) = \frac{3L + 2h - 2L}{3} = \frac{2h + L}{3}$$

اضافه و حذف در آرایه

تابع زیر مقدار صحیح x را در مکان k ام آرایه a با n عنصر که m عنصر آن پر است،(m<n) اضافه می‌کند:

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

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

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

تعداد شیفت مورد نیاز برای اضافه کردن (insert) در محل k ام آرایه، برابر است با: $m-k$

تعداد شیفت مورد نیاز برای حذف عنصر واقع در محل k ام آرایه، برابر است با: $m-k-1$

پیدا کردن عنصر کمینه در یک آرایه

الگوریتم زیر کوچکترین عنصر در آرایه A[1..n] را پیدا می کند و در min ذخیره می کند:

```
MINIMUM(A,n)
1   min ← −∞
2   for i ←1   to   n do
3       if  min > A[i]  then
4           min ← A[i]
```

اگر A[i] ، کوچکترین عنصر زیر آرایه A[1..i] باشد، روشن است که A[i] مقدار min را در سطر ۴ تغییر می دهد. احتمال این امر $\frac{1}{i}$ است. بنابراین میانگین تعداد دفعاتی است که سطر ۴ الگوریتم بالا اجرا می شود برابر است با: $\sum_{i=1}^{n} \frac{1}{i} = \theta(\lg n)$

ماتریس

آرایه دو بعدی را ماتریس می گویند که انواع معروف آن عبارتند از: اسپارس،مثلثی، سه قطری، متقارن و پله ای. قبل از پرداختن به این ماتریس ها، عملیات رایج بر روی ماتریس را بررسی می نماییم.

۱- جمع دو ماتریس

۲- ضرب دو ماتریس

۳- ترانهاده کردن ماتریس

۱- جمع دو ماتریس $A_{m\times n}$ و $B_{m\times n}$

```
void   add(int  a[ ] [max] ,int b[ ] [max] , int c [ ][max] , int  m , int n){
   for( i=0;  i<m ; i++)
      for( j=0;  j<n ; j++)
         c[i][j] = a[i][j] + b[i][j];
}
```

۲- ضرب دو ماتریس $A_{m\times p}$ و $B_{p\times n}$

```
void   mult(int  a[ ] [max] ,int b[ ] [max] , int c [ ][max] , int m , int  p , int  n){
   for(i=0;  i<m ; i++)
     for(j=0;  j<n ; j++)
       {
         c[i][j] =0;
          for(k=0;  k<p ; k++)
             c[i,j] = c[i][j] + a[i][k] *b[k][j] ;
  }
}
```

ماتریس c دارای ابعاد $m \times n$ است و داریم:

$$c_{ij} = \sum_{k=0}^{p-1} a_{ik} b_{kj} \quad 0 \le i < m \text{ و } 0 \le j < n$$

مرتبه اجرایی الگوریتم جمع دو ماتریس n*n برابر $o(n^2)$ و مرتبه ضرب آنها برابر $o(n^3)$ می باشد.

۳- ترانهاده کردن ماتریس $A_{n\times m}$

برای پیدا کردن ترانهاده یک ماتریس باید جای سطرها و ستونها را عوض کنیم.

```
for ( j=0 ; j<m ; j++)
  for( i=0 ; i<n ; i++)
      b[j][i] = a[i][j];
```

در ماتریس $A_{n\times n}$ ، عنصر A[i,j] روی قطر فرعی قرار دارد، اگر $i+j = n+1$.

ضرب بهینه ماتریس ها

در ضرب چند ماتریس در یکدیگر باید ضربها به نحوی انجام شود که تعداد آنها حداقل باشد. بنابراین ابتدا دو ماتریسی را ضرب می کنیم که عدد حذف شده بزرگتر باشد.

در ضرب دو ماتریس $A_{m\times n}, B_{n\times k}$ به $m\times n\times k$ عمل ضرب نیاز می‌باشد.

مثال

حداقل تعداد ضرب های لازم برای ضرب ۴ ماتریس زیر کدام است؟

$$A_{30\times1}, B_{1\times40}, C_{40\times10}, D_{10\times25}$$

حل: ابتدا دو ماتریس B و C را ضرب می کنیم، چون بعد مشترک آنها یعنی 40 از همه بیشتر است:

$$A_{30\times1} \quad (BC)_{1\times10} \; D_{10\times25}$$

سپس ماتریس بدست آمده از مرحله قبل را در ماتریس D ضرب می کنیم:

$$A_{30\times1}(BCD)_{1\times25}$$

و در نهایت ماتریس A را در ماتریس بدست آمده ضرب می کنیم:

$$(ABCD)_{30\times25}$$

تعداد ضرب ها برابر است با:

$$1\times40\times10+1\times10\times25+30\times1\times25=1400$$

تعداد حالات شرکت پذیری ضرب n+1 ماتریس برابر است با : $\frac{\binom{2n}{n}}{(n+1)}$

مثال

تعداد حالات شرکت پذیری ضرب 4 ماتریس چند می باشد؟

حل: با قرار دادن عدد 3 در رابطه $\frac{\binom{2n}{n}}{(n+1)}$ جواب برابر 5 خواهد شد. این 5 حالت به صورت زیر می باشد:

(AB)(CD) , ((A(BC))D) , (A((BC)D)) , (((AB)C)D) , (A(B(CD)))

انواع ماتریس

ماتریسهای معروف عبارتند از:

۱- 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.

ذخیره ماتریس اسپارس به کمک آرایه دو بعدی

ماتریس اسپاس را می‌توان به صورت آرایه دو بعدی ذخیره کرد:

```
var  a : array[0..count , 1..3] of  integer;
```

count تعداد عناصر غیر صفر ماتریس اسپارس است. در خانه a[0,1] ، تعداد سطرها در a[0,2] تعداد ستونها و در a[0,3] تعداد عناصر غیر صفر ماتریس قرار می‌گیرد. در خانه‌های a[1,1] و a[1,2] و a[1,3] به ترتیب سطر و ستون و مقدار اولین عنصر غیرصفر قرار می‌گیرد و به همین ترتیب عناصر غیر صفر دیگر ذخیره می‌شود.

مثال

ذخیره ماتریس اسپارس S در ماتریس دو بعدی a :

$$S=\begin{pmatrix}0 & 0 & 2 & 0\\ 0 & -6 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}\Rightarrow a=\begin{pmatrix}3 & 4 & 2\\ 1 & 3 & 2\\ 2 & 2 & -6\end{pmatrix}$$

سطر اول ماتریس a ، نشان می دهد که ماتریس اسپارس دارای 3 سطر و 4 ستون و 2 عنصر غیر صفر است و سطر دوم ماتریس a نشان می دهد که در سطر 1 و ستون 3 ماتریس اسپارس مقدار 2 قرار دارد. و سطر سوم نشان می دهد که در سطر دوم و ستون دوم ماتریس اسپارس، مقدار 6- قرار دارد.

ترانهاده کردن ماتریس اسپارس

برای ترانهاده کردن یک ماتریس کافی است که سطر اول با ستون اول ، سطر دوم با ستون دوم و .... تعویض شود. برای ترانهاده کردن یک ماتریس اسپارس که به صورت آرایه دو بعدی ذخیره شده ، کافی است که محل ستون اول و ستون دوم در آرایه A را با یکدیگر جابجا کرده و سپس از سطر دوم تا سطر آخر را بطور مرتب نوشت.

مثال

ترانهاده ماتریس اسپارس s که به کمک ماتریس a نمایش داده شده را بدست آورید.

$$S=\begin{pmatrix}0 & 0 & 2 & 0\\ 0 & -6 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}\Rightarrow a=\begin{pmatrix}3 & 4 & 2\\ 1 & 3 & 2\\ 2 & 2 & -6\end{pmatrix}$$

حل:

در ماتریس a، محل ستون اول و دوم را با یکدیگر جابجا کرده:

4	3	2
3	1	2
2	2	-6

و سپس سطر 2 و سطر 3 را بطور مرتب می نویسیم:

4	3	2
2	2	-6
3	1	2

الگوریتم سریع ترانهاده کردن ماتریس اسپارس $m \times n$ از مرتبه $o(n+t)$ می باشد.

(t : تعداد عناصر غیر صفر)

ماتریس مثلثی

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

مثال

ماتریس پایین مثلثی A را به ترتیب سطری در آرایه یک بعدی B ذخیره نمایید.

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

حل: کافی است فقط عناصر غیر صفر ماتریس A را در آرایه یک بعدی B ذخیره نماییم:

	1	2	3	4	5	6
B	2	3	6	5	4	1