به نام خدا

درس: ساختمان داده

مدرس: ملیحه دشتی

فصل اول

مرتبه اجرایی (پیچیدگی اجرایی)

پیچیدگی اجرایی

پیچیدگی یک الگوریتم، تابعی است که مدت زمان اجرای استفاده شده توسط الگوریتم را بر حسب تعداد داده‌های ورودی n اندازه می‌گیرد.

notation	name
O(1)	constant
O(n)	linear
O(logn)	logarithmic
O(n^2)	quadratic
O(n^c)	polynomial
O(c^n)	exponential
O(n!)	factorial

مرتبه اجرایی توابع چند جمله ای

در صورتی که داشته باشیم:

$$f(n) = n^m + n^{m-1} + \ldots + n^2 + n + c \Rightarrow f(n) = O(n^m)$$

مثال:

$$f(n) = 5n^2 - 3n + 4 \Rightarrow O(n^2)$$

$$O(1) < O(\lg n) < O(n) < O(n \lg n) < O(n^2) < O(2^n) < O(n!) < O(n^n)$$

$$\log_2^n = \lg n$$

نمادهای پیچیدگی اجرایی

اوی بزرگ

عبارت $f(n) \in O(g(n))$ یعنی: برای تابع پیچیدگی مفروض g(n)، O(g(n)) به مجموعه ای از توابع اشاره دارد که برای آنها ثابتهای c و n_0 وجود دارند، بطوریکه برای همه $n \geq n_0$ داریم: $f(n) \leq cg(n)$

امگا بزرگ

عبارت $f(n) \in \Omega(g(n))$ یعنی: برای تابع پیچیدگی مفروض g(n)، $\Omega(g(n))$ به مجموعه ای از توابع اشاره دارد که برای آنها ثابتهای c و n_0 وجود دارند، بطوریکه برای همه $n \geq n_0$ داریم: $f(n) \geq cg(n)$

تتا

عبارت $f(n) \in \theta(g(n))$ یعنی : $f(n) \in O(g(n))$ و $f(n) \in \Omega(g(n))$

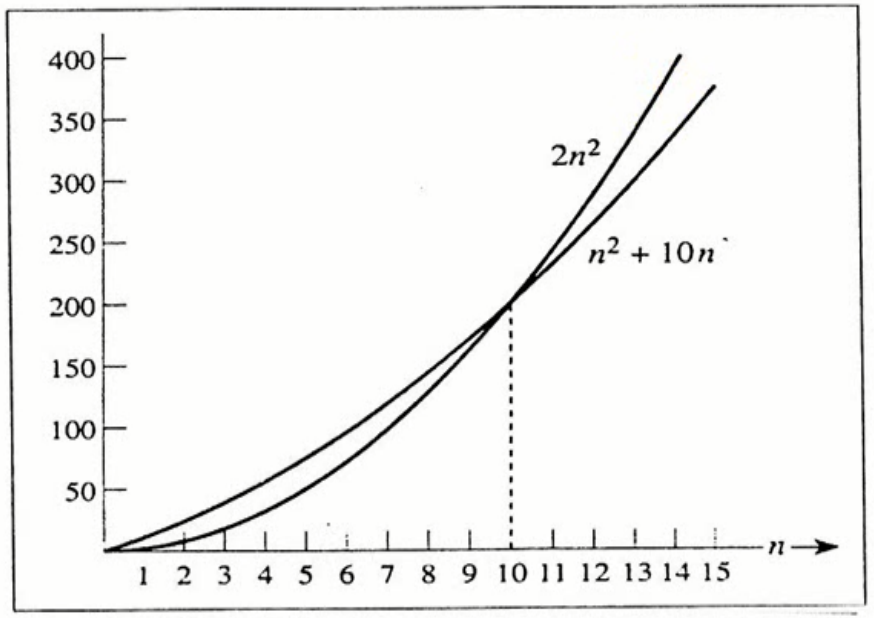
مثال

نشان دهید : $n^2 + 10n \in O(n^2)$

$n^2 + 10n \le cn^2$

$n^2 + 10n \le 2n^2$

$n^2 + 10n = 2n^2 \Rightarrow 10n = n^2 \Rightarrow n = 10$



مثال

نشان دهید : $\frac{n(n-1)}{2} \in O(n^2)$

حل:

$$\frac{n(n-1)}{2} \le cn^2$$

$$\frac{n(n-1)}{2} \le \frac{n^2}{2}$$

$$\frac{n(n-1)}{2} = \frac{n^2}{2} \Rightarrow n = 0$$

یعنی : $n_0 = 0, c = \frac{1}{2}$

مثال

نشان دهید : $\frac{n(n-1)}{2} \in \Omega(n^2)$

حل:

$$\frac{n(n-1)}{2} \geq cn^2$$

$$\frac{n^2}{2} - \frac{n}{2} \geq \frac{1}{4}n^2$$

$$\frac{n^2}{2} - \frac{n}{2} = \frac{n^2}{4} \Rightarrow n = 2$$

یعنی: $n_0 = 2, c = \frac{1}{4}$

مثال

نشان دهید که : $\frac{n(n-1)}{2} \in \theta(n^2)$

جواب:

در مثال های قبل دیدیم که $\frac{n(n-1)}{2} \in O(n^2)$ و $\frac{n(n-1)}{2} \in \Omega(n^2)$ بنابراین :

$$\frac{n(n-1)}{2} \in \theta(n^2)$$

پیچیدگی دستور if

به دستور if زیر توجه کنید:

```
if(cond)
    block1
else
    block2
```

زمان اجرا در بدترین حالت برابر است با:

$$Max(time(block1) , time(block2))$$

مرتبه اجرایی حلقه های ساده

حلقه for زیر را در نظر بگیرید: (a<=b)

```
for (i=a ; i<=b ; i=i+k )
    s;
```

```
for (i=b ; i>=a ; i=i-k )
    s;
```

تعداد تکرار : $\frac{b-a+1}{k}$

اگر این مقدار اعشاری بود، حد بالای آن را در نظر بگیرید.

منظور از s ، توالی از دستورها (sequence of statement) است.

مثال

تعداد تکرار دستور داخل حلقه را مشخص کنید.

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

جواب:

تعداد تکرار : $\frac{n-1+1}{1} = n$

مرتبه اجرایی : o(n)

مثال

تعداد تکرار دستور داخل حلقه را مشخص کنید.

```
for ( i=3 ; i<=n ; i=i+2 )
        s;
```

جواب:

تعداد تکرار : $\frac{n-3+1}{2} = \frac{n}{2} - 1$

مرتبه اجرایی : o(n)

مثال

تعداد تکرار دستور داخل حلقه را مشخص کنید.

```
for ( i=9 ; i<3n+4 ; i=i+5  )
        s;
```

جواب:

تعداد تکرار: $\frac{3n+4-9}{5} = \frac{3n}{5} - 1$

مرتبه اجرایی : $o(n)$

مرتبه ثابت

مرتبه اجرایی دستور x=x+1 را مشخص کنید.

```
i=n;
while(i>1)
{
   i=i % 2;
   x=x+1;
}
```

جواب: $O(1)$

مرتبه لگاریتمی

حلقه زیر را در نظر بگیرید:

```
for ( i=b ; i>=a ; i=i/k )
    s;
```

```
for ( i=a ; i<=b ; i=i*k )
    s;
```

تعداد تکرار : $\log_k^b - \log_k^a + 1$

اگر این مقدار صحیح نبود، حد بالای آن را در نظر بگیرید.

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=8 ; i=i*2 )
        s;
```

جواب: $\log_2^8 - \log_2^1 + 1 = 4$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=27 ; i<=n ; i=i*3  )
        s;
```

جواب:

$$\log_3^n - \log_3^{27} + 1 = \log_3^n - 2 \Rightarrow O(\log_3^n)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=n ; i>=16 ; i=i/4  )
        s;
```

جواب:

$$\log_4^n - \log_4^{16} + 1 = \log_4^n - 1 \Rightarrow O(\log_4^n)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=n ;  i>=1;  i=i - i/3 )
        s;
```

```
for ( i=n ;   i>=1;  i= i / (3/2)  )
        s;
```

جواب:

می توان نوشت : $i = \frac{2}{3} i = \frac{i}{\frac{3}{2}}$

بنابراین تعداد تکرار برابر است با :

$$\log_{3/2}^{n} - \log_{3/2}^{1} + 1 = \log_{3/2}^{n} + 1$$

حلقه های تودرتو

تعداد تکرار را مشخص کنید.

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

حل:

تعداد تکرار هر حلقه: n

تعداد تکرار دستور s : n^2

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=2 ; i<=n ; i=i+4 )
    for ( j=n ; j>3 ; j=j-2 )
            s;
```

جواب:

$$\frac{n-2+1}{4} \times \frac{n-3}{2} \Rightarrow O(n^2)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ;  i=i*2 )
      for ( j=1 ; j<=n ; j++ )
                  s;
```

جواب:

$$(\lg n + 1) \times n \Rightarrow O(n \lg n)$$

حلقه های پشت سرهم

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++) {
            s;
 }
for ( j=1 ; j<=m ; j++) {
            s;
}
```

جواب: $O(\max(n,m))$

یا $O(n+m)$

ترکیب حلقه های تودر تو و پشت سرهم

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++) {
      for ( j=1 ; j<=n ; j++) {
                  s;
      }
}
for ( k=1 ; j<=n ; k++) {
                  s;
}
```

جواب: $O(\max(n^2, n)) = O(n^2)$

حلقه های تودرتو وابسته

تعداد تکرار را مشخص کنید.

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

جواب:

i	1	2	3	...	n
تعداد تکرار	1	2	3	...	n

مجموع تعداد تکرار :

$$1+2+3+\ldots+n=\frac{n(n+1)}{2}\Rightarrow O(n^2)$$

روش دوم:

$$\sum_{i=1}^{n}\sum_{j=1}^{i}1=\sum_{i=1}^{n}i=\frac{n(n+1)}{2}$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i=i*2  )
    for ( j=1 ; j<=i ; j++ )
            S;
```

جواب:

i	1	2	4	...	n
تعداد تکرار	1	2	4	...	n

مجموع تعداد تکرار :

$$1+2+4+\ldots+n=2^0+2^1+2^2+\ldots+2^{\log_2^n}=\frac{2^{\lg n+1}-1}{2-1}=2^{\lg n}\times 2-1=2n-1\Rightarrow O(n)$$

$$a^0+a^1+a^2+a^3+\ldots+a^k=\frac{a^{k+1}-1}{a-1}$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( k=1 ; k<=n ; k++ )
    for ( i=1 ; i<=n ; i=i*2 )
        for ( j=1 ; j<=i ; j++ )
            s;
```

جواب:

در مثال قبل دیدیم که دو حلقه داخلی (2n-1) مرتبه تکرار می شود. چون حلقه اول n مرتبه تکرار می شود، تعداد کل تکرار برابر است با:

$$n(2n-1) \Rightarrow O(n^2)$$

مثال

تعداد تکرار را مشخص کنید.

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

جواب:

i	1	2	3	...	n
تعداد تکرار	n	n/2	n/3	...	1

مجموع تعداد تکرار :

$$n+\frac{n}{2}+\frac{n}{3}+\ldots+1=n(1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n})\approx n\ln n\Rightarrow O(n\lg n)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i=i*3 )
    for ( j=i ; j<=n ; j++ )
        s;
```

جواب:

i	1	3	9	...	n
تعداد تکرار	n-1+1	n-3+1	n-9+1	...	n-n+1

مجموع تعداد تکرار :

$$(\log_3^n+1)(n+1)-(1+3+9+\ldots+n)=(\log_3^n+1)(n+1)-\frac{3n-1}{2}\Rightarrow O(n\lg n)$$

$$1+3+9+\ldots+n=3^0+3^1+3^2+\ldots+3^{\log_3^n}=\frac{3^{\log_3^n+1}-1}{3-1}=\frac{3n-1}{2}$$

مثال

تعداد تکرار را مشخص کنید.

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

جواب:

i	1	2	...	n
تعداد تکرار	log(1)	log(2)	...	log(n)

مجموع تعداد تکرار :

$$\log(1)+\log(2)+\ldots+\log(n)=\log(1\times 2\times\ldots\times n)=\log(n!)$$

$$\log(n!)=\theta(n\lg n)$$

پایان فصل اول

فصل دوم

توابع بازگشتی

تعریف

تابع بازگشتی (recursive) :

تابعی است که حاوی حداقل یک دستور باشد که خود تابع را صدا بزند. این تابع به تعداد مراحل محدودی اجرا می‌شود و پس از آن متوقف می‌شود.

مثال هایی از توابع بازگشتی :

۱- فاکتوریل

۲- مجموع اعداد ۱ تا n

۳- توان

۴-ترکیب

۵-خارج قسمت تقسیم صحیح

۶- آکرمان

۷- هانوی

۸- فیبوناچی

تابع فاکتوریل

تابع زیر فاکتوریل **n** را محاسبه می کند:

```
f( n ){
  if (n==1)
        return  1;
  else
        return  n * f(n-1);
}
```

$$f(n)=\begin{cases}1 & n=1\\ n\times f(n-1) & n>1\end{cases}$$

مثال: فراخوانی تابع با مقدار ۴ :

f(4)=4*f(3)=4*6=24
f(3)=3*f(2)=3*2=6
f(2)=2*f(1)=2*1=2

تابع فاکتوریل

تابع زیر فاکتوریل **n** را محاسبه می کند:

```
f( n ){
  if (n==1)
        return  1;
  else
        return  n * f(n-1);
}
```

$$f(n)=\begin{cases}1 & n=1\\ n\times f(n-1) & n>1\end{cases}$$

مثال: فراخوانی تابع با مقدار ۴ :

f(4)=4*f(3)=4*6=24
f(3)=3*f(2)=3*2=6
f(2)=2*f(1)=2*1=2

مجموع اعداد 1 تا n

تابع زیر مجموع اعداد **1** تا **n** را محاسبه می کند:

```
sum( n )
{
  if (n==1)
        return  1;
  else
        return   n+ sum(n-1) ;
}
```

$$sum(n)=\begin{cases}1 & n=1\\ n+sum(n-1) & n>1\end{cases}$$

خروجی فراخوانی تابع به ازای n=3 برابر 6 است:

```
sum(3)=3+sum(2)  =3+3=6
sum(2)=2+sum(1)  =2+1=3
```

تابع توان

تابع زیر مقدار n^m را محاسبه می کند : (ورودی ها صحیح و مثبت هستند)

```
f (n , m){
  if (m==1)
        return  n;
  else
        return  n * f(n,m-1);
}
```

$$f(n,m)=\begin{cases} n & m=1 \\ n\times f(n,m-1) & m>1 \end{cases}$$

به طور نمونه تابع را به صورت f(3,4) فراخوانی می کنیم:

f(3,4) = 3 * f(3,3) =3*3*3*3

f(3,3) = 3 * f(3,2) =3*3*3

f(3,2) = 3 * f(3,1) =3*3

تابع ترکیب

تابع **f** ترکیب **m** از **n** را محاسبه می کند:

```
f ( n , m )
{
  if ( (n==m)   ||  (m==0) )
        return  1;
  else
        return   f(n-1 , m) + f(n-1 , m-1);
}
```

$$\binom{n}{m} = 1 \quad \text{if } m = 0 \text{ or } m = n$$

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1} \quad \text{if } 0 < m < n$$

فراخوانی f(4,2) :

f (4,2) = f (3,2) + f (3,1) =3+3=6
f (3,2) = f (2,2) + f (2,1) =1+2=3
f (3,1) = f (2,1) + f (2,0) =2+1=3
f (2,1) = f (1,1) + f (1,0) =1+1=2

تابع div

(محاسبه خارج قسمت تقسیم صحیح)

محاسبه خارج قسمت تقسیم صحیح **a** بر **b** :

$$f(a,b)=\begin{cases}0 & a<b\\ f(a-b,b)+1 & a>=b\end{cases}$$

فراخوانی f(11,3) :

f (11,3) = f (8,3) + 1 = 2+1=3
f (8,3) = f (5,3) + 1 = 1+1=2
f (5,3) = f (2,3) + 1 = 0+1=1

تمرین: با تغییری در این تابع، mod را پیاده سازی کنید.

تابع آکرمان

تابع زیر به نام آکرمان معروف است:

$$f(a,b)=\begin{cases} b+1 & a=0 \\ f(a-1,1) & b=0 \\ f(a-1,f(a,b-1)) & a>0,b>0 \end{cases}$$

فراخوانی f(1,1) :

```
f(1,1)
=f(0,f(1,0))
=f(0,f(0,1))
=f(0,2)
=3
```

روابطی در تابع آکرمان

در تابع آکرمان روابط زیر برقرار است:

$$f(1,n) = 2 + (n+3) - 3$$

$$f(2,n) = 2(n+3) - 3$$

$$f(3,n) = 2^{n+3} - 3$$

$$f(4,n) = \underbrace{2^{2^{\cdots^{2}}}}_{n+3} - 3$$

به طور نمونه :

$$f(1,7) = 2 + (7+3) - 3 = 9$$

$$f(2,7) = 2(7+3) - 3 = 17$$

$$f(3,7) = 2^{(7+3)} - 3 = 1021$$

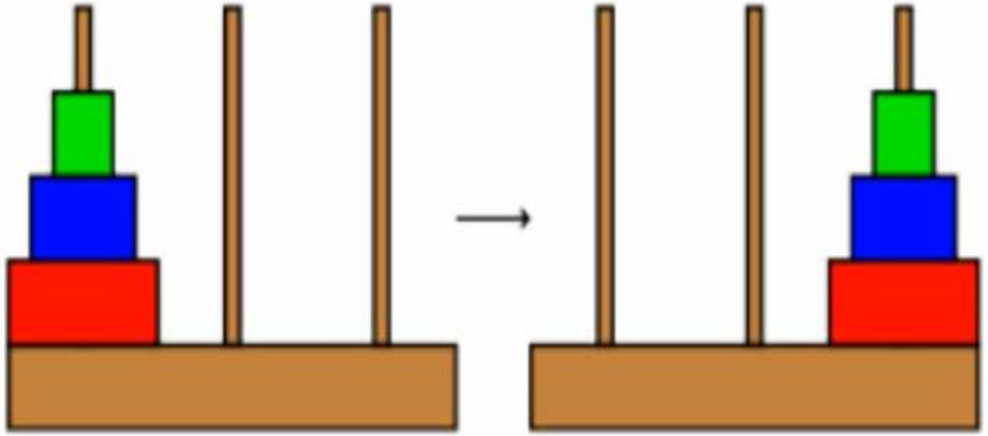
$$f(4,1) = 2^{2^{2^{2}}} - 3$$

برج هانوی

سه میله A,B,C وجود دارد، که n مهره بر روی میله A قرار دارد. هدف انتقال n مهره به میله C است که برای این کار از میله Bکمک گرفته می شود.

اندازه مهره ها در میله A از پایین به بالا کاهش می یابد.

در انتقال مهره ها هرگز مهره بزرگتر بر روی مهره کوچکتر نباید قرار بگیرد. در هر بار فقط امکان انتقال یک مهره وجود دارد که از بالای میله انتخاب می شود.



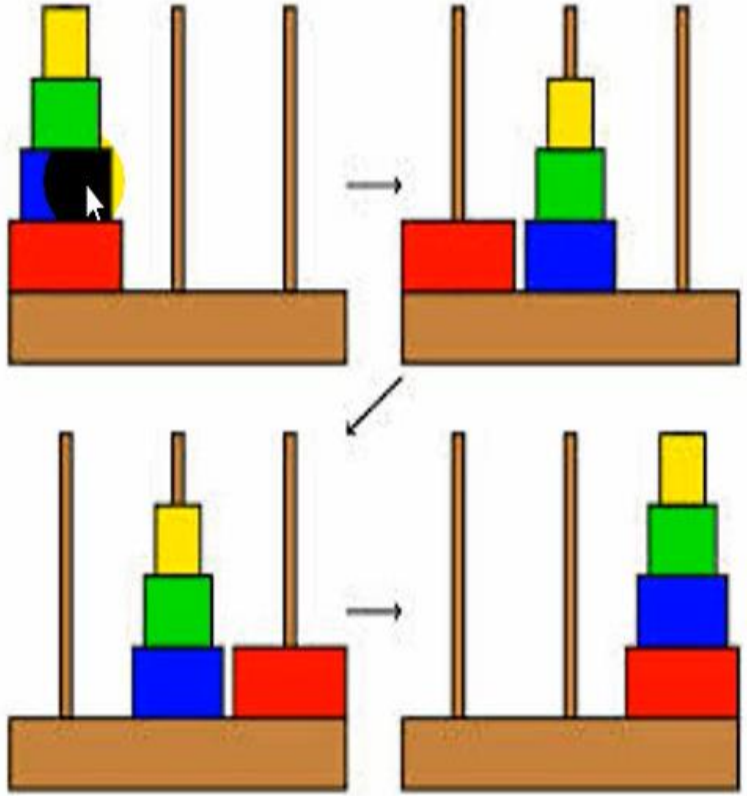
برج هانوی(ادامه)

در صورت وجود بیش از یک مهره ، مسئله را به صورت زیر حل می کنیم:

۱) انتقال n-1 مهره از میله Aبه میله B.

۲) انتقال مهره nام از میله Aبه میله C.

۳) انتقال n-1 مهره از میله Bبه میله C.



برج هانوی(ادامه)

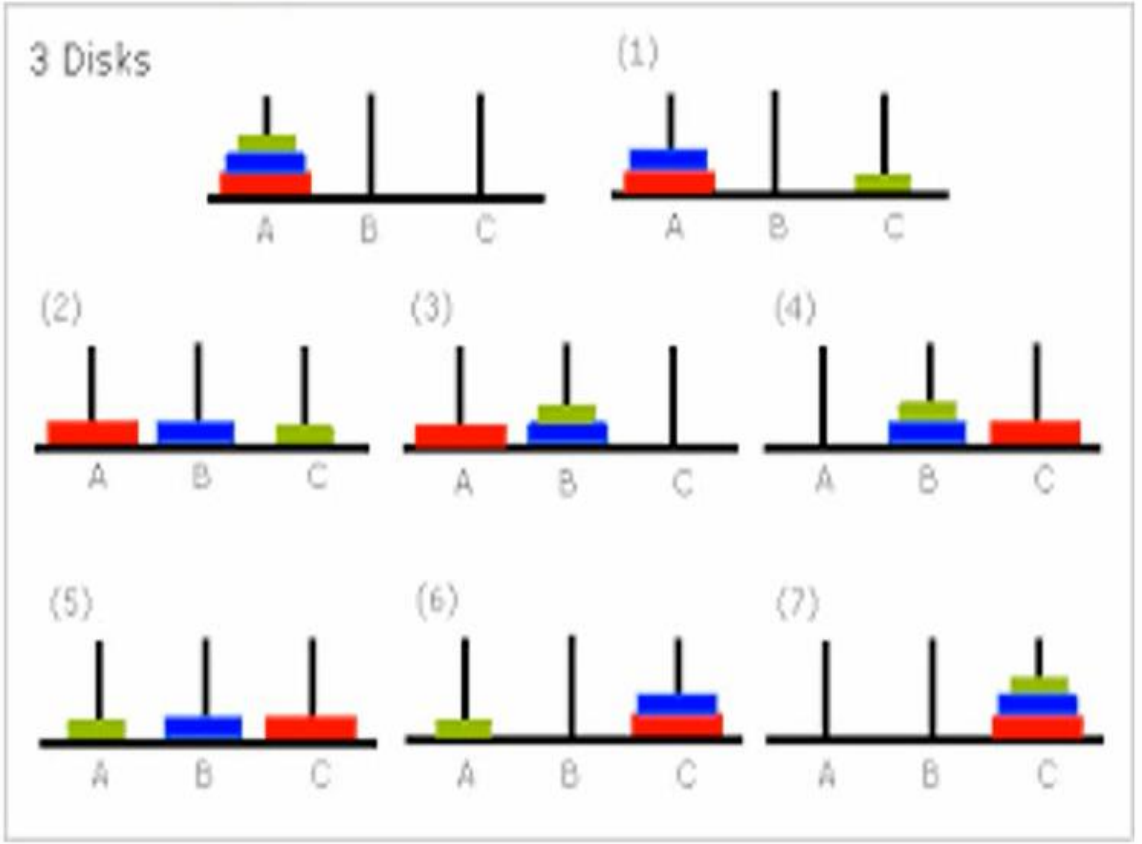
الگوریتم انتقال **n** مهره از **A** به **C** به کمک **B**

```
tower (n , A, B, C)
{
    if  (n ==1 )
        A  to C;
    else {
            tower  ( n–1 , A, C, B);
            A to C;
            tower ( n–1 , B, A, C);
    }
}
```

برج هانوی(مثال)

انتقال 3 مهره از میله A به میله C به کمک میله B : T(3,A,B,C)

$$
\begin{cases}
T(2,A,C,B)=\begin{cases} T(1,A,B,C)=A\to C & (1)\\ A\to B & (2)\\ T(1,C,A,B)=C\to B & (3)\end{cases}\\
A\to C \quad (4)\\
T(2,B,A,C)=\begin{cases} T(1,B,C,A)=B\to A & (5)\\ B\to C & (6)\\ T(1,A,B,C)=A\to C & (7)\end{cases}
\end{cases}
$$



تابع فیبوناچی

محاسبه جمله n ام سری فیبوناچی : (0,1,1,2,3,5,8,13,21,34,...)

```
f(n){
  if  ((n==0) || (n==1) )
        return  n;
  else
        return  f(n-1) + f(n-2);
}
```

$$f(n)=\begin{cases}0 & n=0\\ 1 & n=1\\ f(n-1)+f(n-2) & n\geq 2\end{cases}$$

محاسبه جمله چهارم سری فیبوناچی :

f(4)=f(3) + f(2) =2+1=3

f(3)=f(2) + f(1) =1+1=2

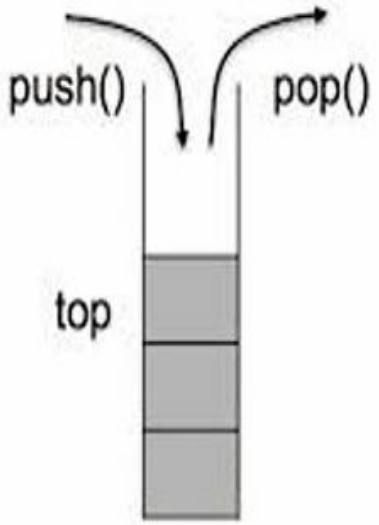
f(2)=f(1) + f(0) =1+0=1

کاربرد پشته در زیر برنامه های بازگشتی

تعریف پشته: (stack)

پشته ساختمان داده ای است که حذف و اضافه از بالای آن انجام می‌شود.

پشته را FILO می‌نامند، چون آخرین عنصر وارد شده در آن، اولین عنصری است که از آن برداشته می‌شود.



مثال

خروجی زیر برنامه زیر، به ازای فراخوانی **f (1)** چیست؟

```
f (n){
    if (n==3)
        exit( );
    else{
        n=n+1;
        f(n);
        cout<<n;
        }
}
```

$$f(1) \Rightarrow f(2) \Rightarrow f(3)$$

بنابراین خروجی برابر 32 خواهد بود.

مثال

خروجی f(4) چه می باشد؟

```
f (n){
   if (n >2)
    {
       f(n-1);
       cout << 'a' ;
       cout << 'b' ;
    }
}
```

$$f(4) \Rightarrow f(3) \Rightarrow f(2)$$

مثال

خروجی زیر برنامه زیر، به ازای فراخوانی **f (1,4)** چیست؟

```
f (n,m){
    if (n==3)
        exit( );
    else {
        n=n+1;
        m=m+1;
        f(n,m);
        cout<<n;
        cout<<m;
        }
}
```

$$f(1,4) \Rightarrow f(2,5) \Rightarrow f(3,6)$$

بنابراین خروجی برابر 3625 خواهد بود.

رابطه بازگشتی برای توابع بازگشتی

یک رابطه بازگشتی برای تعداد ضرب ها در تابع فاکتوریل بنویسید.

```
fact (n){
   if (n==0)   return  1;
    else   return   n*fact(n-1);
}
```

حل:

برای یک n معین تعداد ضرب هایی که انجام می شود برابر است با تعداد ضرب های انجام شده در فراخوانی (n-1) به اضافه عمل ضرب n در fact(n-1) . بنابراین اگر تعداد ضرب های انجام شده برای یک مقدار معین n را با T(n) نمایش دهیم، داریم:

$$T(n) = 1 + T(n-1)$$

وقتی n=0 باشد، هیچ ضربی صورت نمی گیرد: $T(0) = 0$

مثال

اگر T(n) تعداد ستاره های چاپ شده توسط f(n) باشد، رابطه بازگشتی T(n) را مشخص کنید.

```
f (n){
  if (n>=2)
   {
     f (n-1);
     f (n-1);
     f (n-2);
     cout<<"*";
   }
}
```

$$T(n) = T(n-1) + T(n-1) + T(n-2) + 1$$

$$T(0) = 0$$

$$T(1) = 0$$

رابطه بازگشتی برای مسئله برج هانوی

اگر به تابع برج هانوی نگاه کنیم، مشاهده می شود که تابع دو بار خودش را فراخوانی می کند:

$$T(n) = 2T(n-1) + 1$$

جواب این رابطه برابر است با:

$$T(n) = 2^n - 1$$

روش های حل رابطه های بازگشتی

رابطه های بازگشتی را می توان به روش های زیر حل کرد:

۱- حدس

۲- تکرار با جایگذاری

۳- قضیه اصلی

۴- درخت بازگشت

۵- روش های حل رابطه های بازگشتی همگن و ناهمگن

این روش ها در درس تحلیل و طراحی الگوریتم بررسی می شود.

پایان فصل دوم

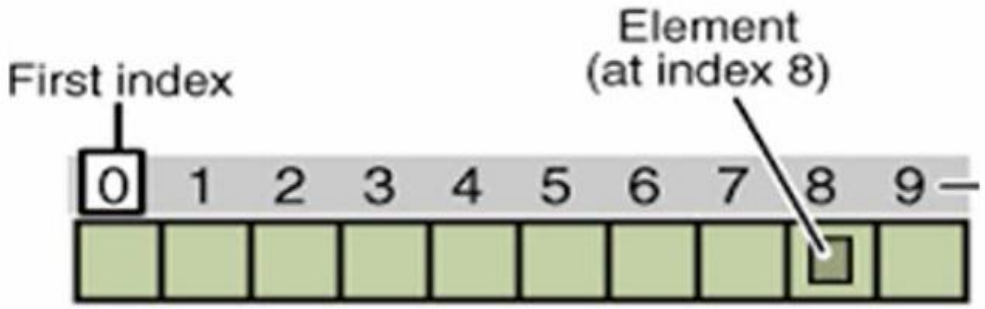
فصل سوم

آرایه

تعریف آرایه

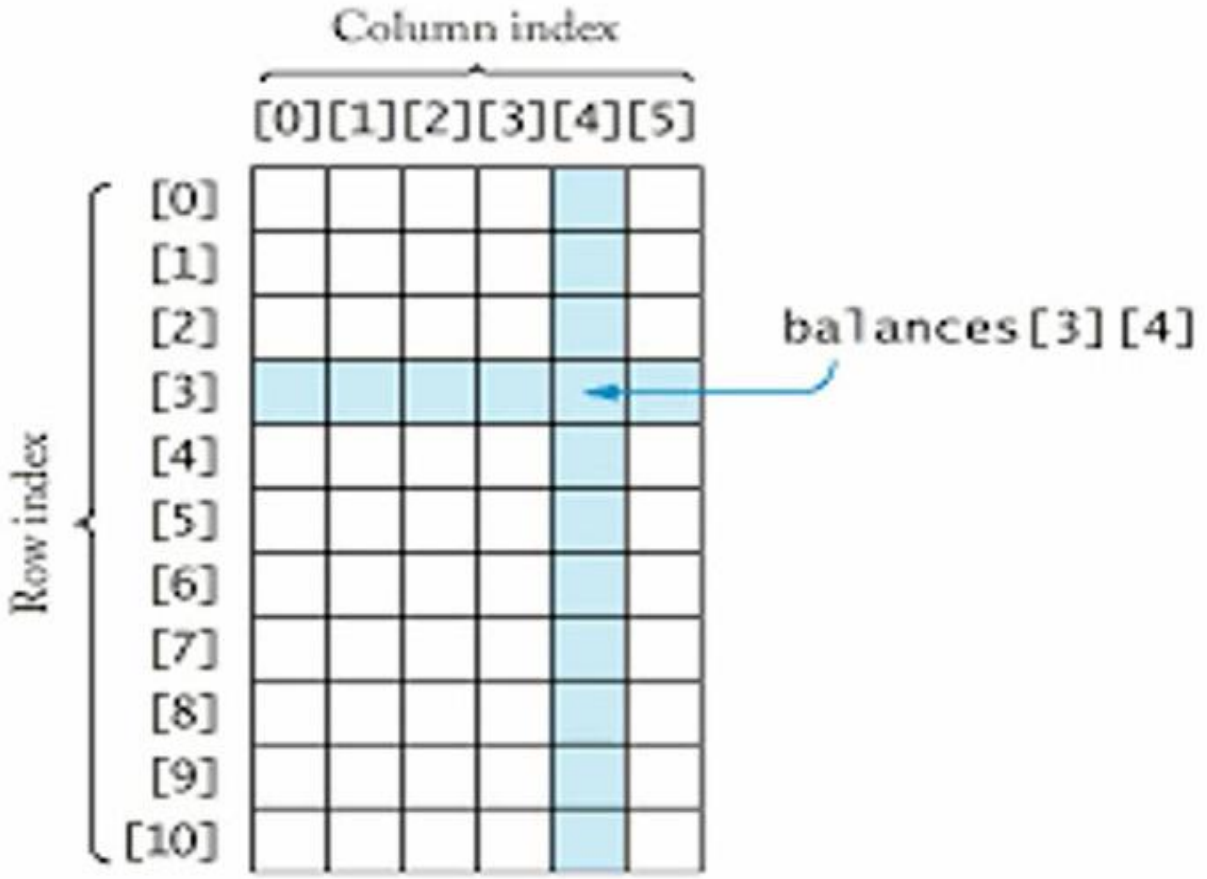
آرایه مجموعه‌ای از داده‌های هم نوع است که تحت یک نام معرفی شده و برای دسترسی به هر عنصر آن از اندیس مشخصی استفاده می‌شود.

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



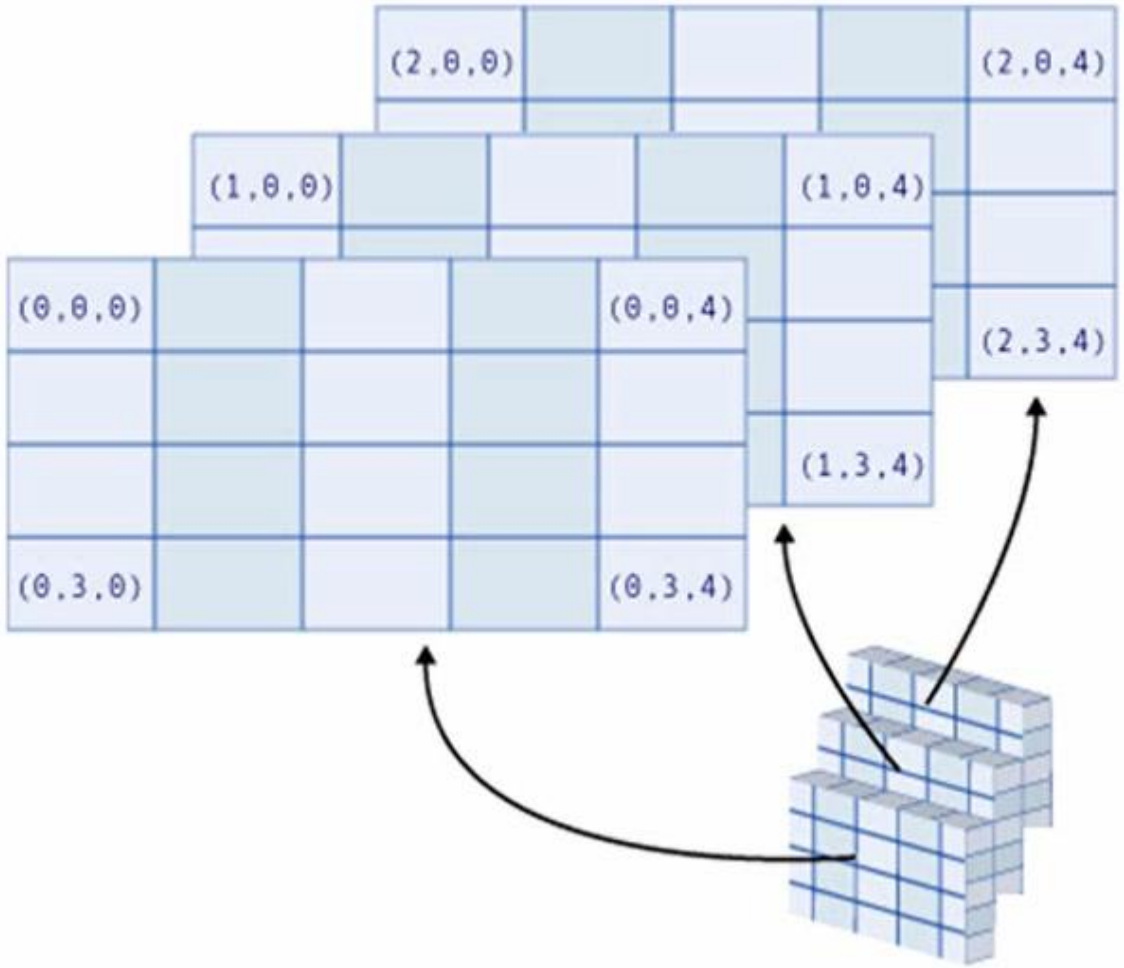
آرایه دو بعدی

```
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

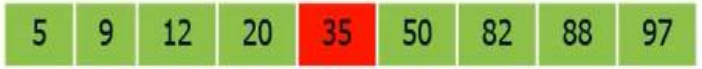
با فرض اینکه آرایه به طور صعودی مرتب شده باشد، عنصر مورد جستجو با عنصر وسط آرایه مقایسه می شود، در صورت برابر بودن، پیدا شده است .

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 ۱۲در آرایه مرتب:

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



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

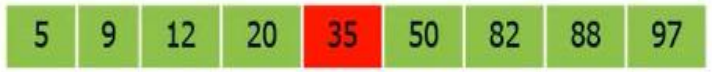
۲- مقایسه ۱۲ با عنصر وسط x[1..4] یعنی ۹ .

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

مثال

مثال: پیدا کردن عدد ۱۲در آرایه مرتب:

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



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.

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.

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:

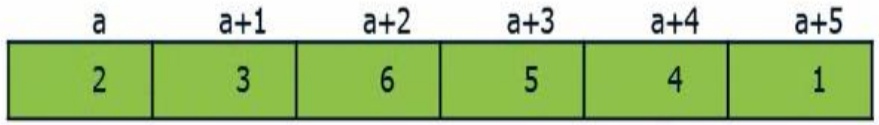
$$\begin{pmatrix} 0 & 0 & 2 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 3 & 4 & 2 \\ 1 & 3 & 2 \\ 2 & 2 & 5 \end{pmatrix}$$

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:

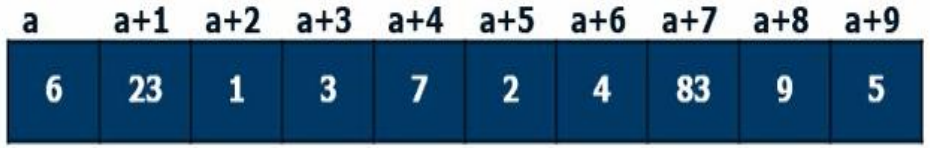
$$a + \frac{i(i-1)}{2} + j - 1$$

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:

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



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

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

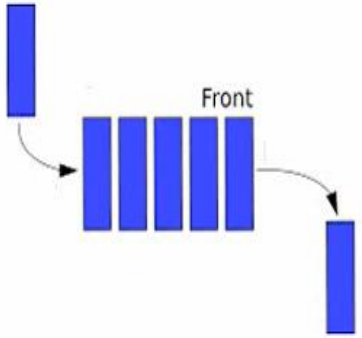
پایان فصل سوم

فصل چهارم

صف و پشته

صف (queue)

صف، ساختمان داده ای است که عمل حذف از ابتدای آن و درج به انتهای آن انجام می‌شود.



صف را لیست FIFO (First In First Out) می نامند، زیرا اولین عنصر وارد شده به صف، اولین عنصری است که خارج می‌شود.

برای نمایش صف، از آرایه یک بعدی queue[0..n-1] و دو متغیر front و rear استفاده می‌شود.

متغیر front : یکی کمتر از مقدار محل عنصر اول صف

متغیر rear : محل آخرین عنصر صف

مثال

یک صف با 4 خانه که در ابتدا خالی است مفروض می‌باشد. (q[0..3])

0	1	2	3
A			
A	B		
A	B	C	
	B	C	
		C	

front = -1 , rear = -1

front = -1 , rear = 0 درج A

front = -1 , rear = 1 درج B

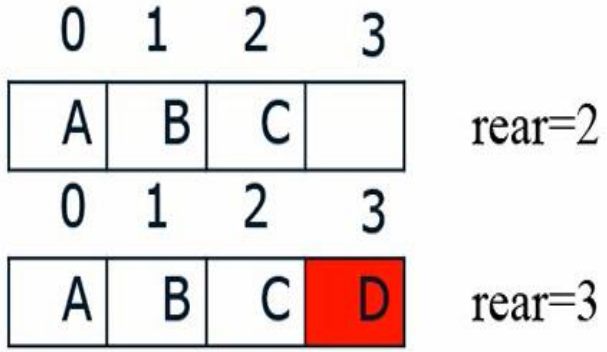
front = -1 , rear = 2 درج C

front = 0 , rear = 2 حذف A

front = 1 , rear = 2 حذف B

درج در صف

```
addq (rear , item)
{
  if (rear == n-1)
       {
        queue-full( );
        return;
       }
   rear= rear+1;
   queue[rear] = item;
}
```

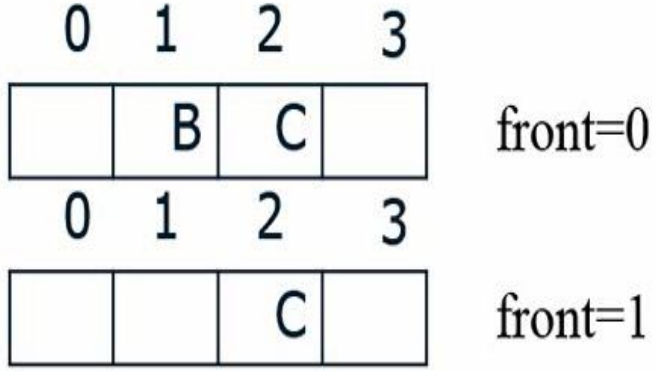


تذکر: در صف پر مقدار rear برابر n-1 می باشد.

تذکر: در صف خالی مقدار front با rear برابر است.

حذف عنصر از صف

```
delq (front , rear)
{
  if (front == rear)
       return queue-empty( );
  return   q[++front] ;
}
```



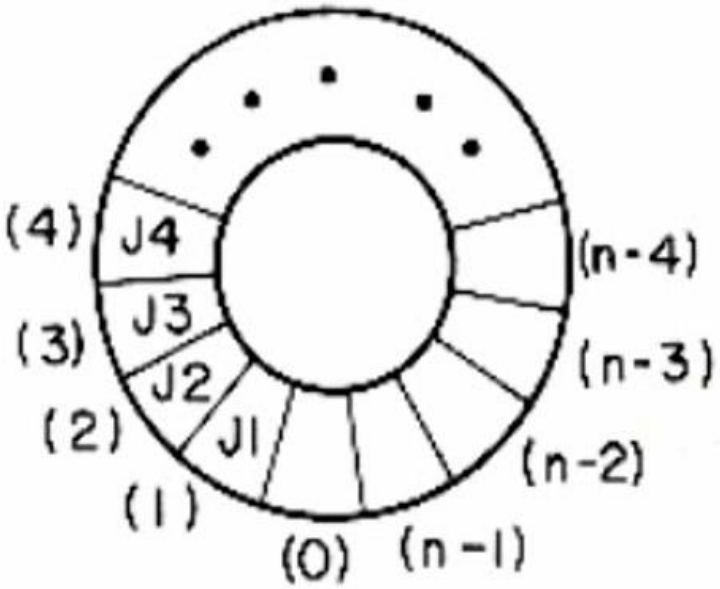
صف حلقوی

در صف حلقوی اندیس front به یک موقعیت عقب تر (در خلاف حرکت عقربه های ساعت) از اولین عنصر موجود در صف اشاره می کند. rear به انتهای فعلی صف اشاره می کند.

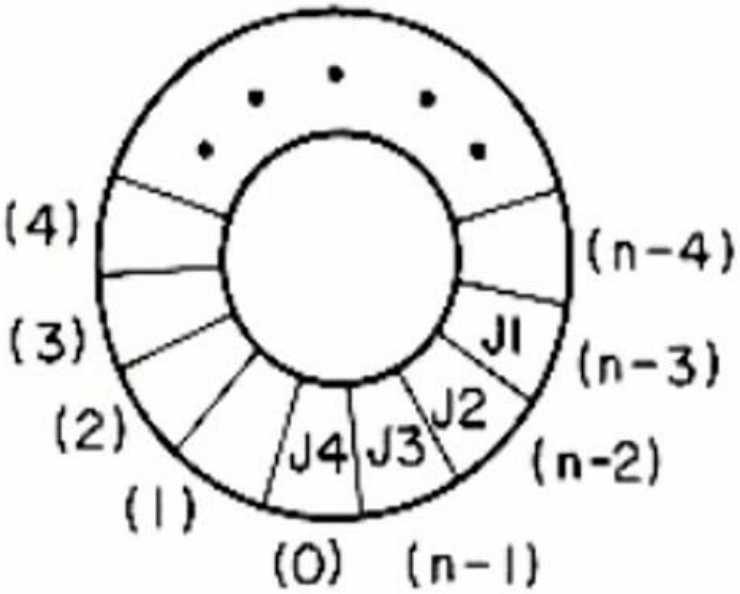
در یک صف حلقوی به اندازه n، حداکثر n-1عنصر می تواند، قرار گیرد.

اگر از آن یک خانه نیز استفاده شود، front=rear می شود و نمی توانیم یک صف پر و خالی را از هم تشخیص دهیم.

مثال



front =0; rear = 4



front = n-4, rear = 0

درج در صف حلقوی

```
addq (front , rear , item)
{
   rear = (rear+1) % n;
    if ( rear == front )
        {
          queue-full(rear);
          return;
        }
   queue[rear] = item ;
}
```

حذف از صف حلقوی

```
deleteq (front , rear)
{
  if (front == rear)
      return  queue-empty( );
  front = (front+1)  %  n ;
   return   queue[front] ;
}
```

حذف از صف حلقوی

```
deleteq (front , rear)
{
  if (front == rear)
      return  queue-empty( );
  front = (front+1)  %  n ;
   return   queue[front] ;
}
```

اگر front=rear باشد، صف خالی خواهد بود.

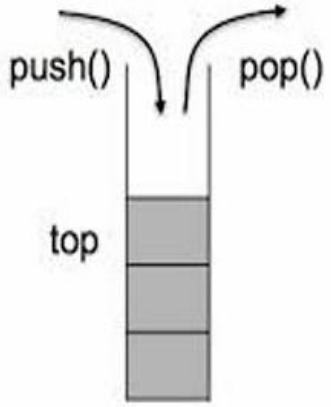
پیاده سازی توابع queue-full و queue-empty بسته به کاربردهای خاص دارد.

پشته (stack)

پشته ، ساختمان داده ای است که حذف و اضافه از بالای آن انجام می‌شود.

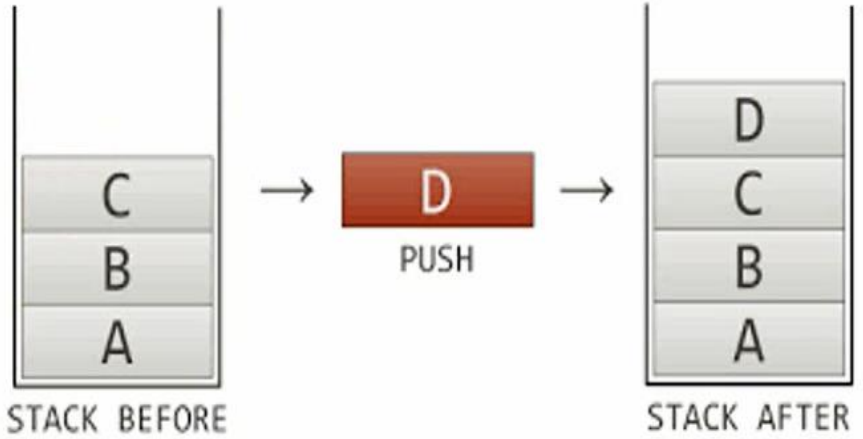
پشته را LIFO می‌نامند، چون آخرین عنصر وارد شده در آن، اولین عنصری است که از آن برداشته می‌شود.

top : مشخص کننده عنصر بالایی پشته



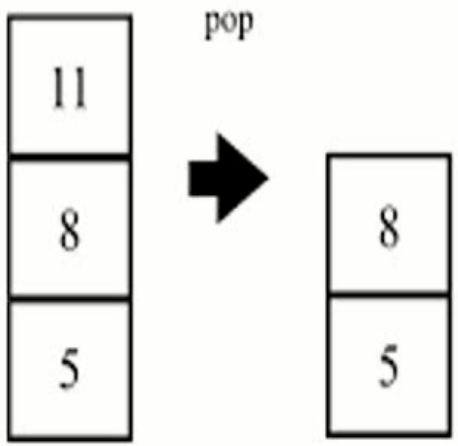
درج در پشته

```
push (top , item)
{
   if ( top >= max-1 )
    {
     stack-full( );
     return;
    }
   top=top+1;
   stack[top] = item;
}
```



حذف از پشته

```
pop ( top )
{
    if ( top == -1 )
          return  stack-empty( );
   return
          stack[top--];
}
```



جابه‌جایی قطارها

در ایستگاه‌های قطار برای جابه‌جا کردن قطارها از یک ریل اضافی(s) استفاده می کنند که مثل پشته عمل می کند. یعنی اولین قطار که وارد این ریل می شود آخرین قطاری است که از آن خارج می شود. فرض کنید در ریل ورودی دنباله‌ای از قطارها با شماره های < 1 , 2 , 3 > پشت سر هم قرار دارند. (۱ در ابتدای ردیف است). اگر مراحل زیر انجام شود، خروجی چه خواهد بود؟

۱- قطار ۱ وارد S می شود.

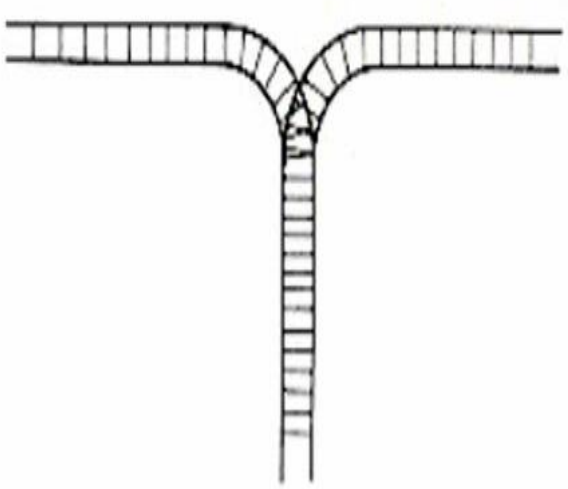
۲- قطار ۲ وارد S می شود.

۳- قطار ۲ از S خارج شده و وارد ریل خروجی می شود.

۴- قطار ۳ وارد S می شود.

۵- قطار ۳ از S خارج شده و به ریل خروجی می رود.

۶- قطار ۱ از S خارج شده و به ریل خروجی می رود.



جابه‌جایی قطارها

در ایستگاه‌های قطار برای جابه‌جا کردن قطارها از یک ریل اضافی(S) استفاده می کنند که مثل پشته عمل می کند. یعنی اولین قطار که وارد این ریل می شود آخرین قطاری است که از آن خارج می شود. فرض کنید در ریل ورودی دنباله‌ای از قطارها با شماره های < 1 , 2 , 3 > پشت سر هم قرار دارند. (۱ در ابتدای ردیف است). اگر مراحل زیر انجام شود، خروجی چه خواهد بود؟

۱- قطار ۱ وارد S می شود.

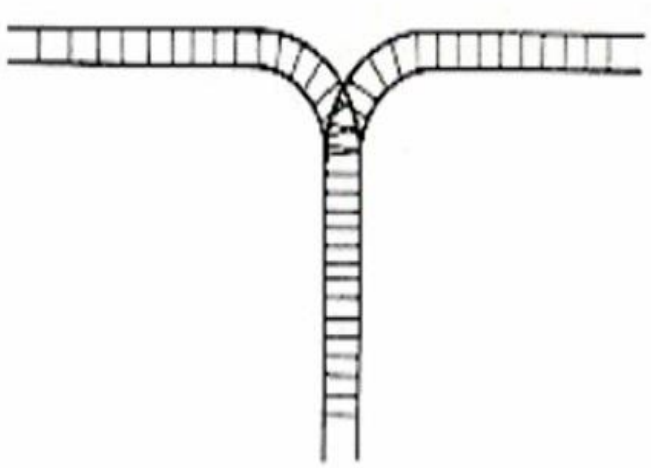
۲- قطار ۲ وارد S می شود.

۳- قطار ۲ از S خارج شده و وارد ریل خروجی می شود.

۴- قطار ۳ وارد S می شود.

۵- قطار ۳ از S خارج شده و به ریل خروجی می رود.

۶- قطار ۱ از S خارج شده و به ریل خروجی می رود.



مثال

اعداد 1 تا 8 به ترتیب در پشته قرار دارند (۸ در بالای پشته). عمل pop و push به صورت زیر تعریف شده اند.

Push : اولین عدد ورودی را برداشته و در بالای پشته قرار می دهد.

Pop : عدد بالای پشته را برداشته و در انتهای دنباله خروجی می نویسد.

با ترکیب مناسبی از ۸ عدد push و ۸ عدد pop می توان جایگشتی از اعداد ۱تا ۸را در خروجی تولید کرد که به آن جایگشت قابل قبول می گوییم. آیا جایگشت زیر قابل قبول است؟

$< 4,3,7,8,6,2,5,1 >$

حل:

push(1) , push(2) , push(3) , push(4) , pop(4) , pop(3) , push(5) , push(6) , push(7) , pop(7) , push(8) , pop(8) , pop(6) , ???

بعد از بیرون آوردن ۶ ، نمی توان ۲ را خارج کرد ، چون زیر ۵ مانده است.

راه سریع: برای آنکه یک دنباله خروجی قابل قبول باشد ،از انتها به سمت ابتدا حرکت کرده و برای هر عدد x ، اعداد کوچکتر از x که بعد از آن قرار دارند، باید یک دنباله نزولی باشند. در این دنباله ، اعداد بعد از 6 یعنی <2,5,1>، یک دنباله نزولی نمی باشند.

نگارش های مختلف عبارت ریاضی

یک عبارت از عملوندها و عملگرها ساخته شده که می‌تواند به سه شکل نمایش داده شود:

۱- prefix (پیشوندی) * A B

۲- infix (میانوندی) A * B

۳- postfix (پسوندی) A B *

اولویت بین عملگرهایی دودویی:

۱- توان

۲- ضرب و تقسیم

۳- جمع و تفریق

تبدیل infix به postfix

مثال :

((A + B) * D) ^ (E – F)

(**AB+** *D) ^ (E – F)

AB+D* ^ (E – F)

AB+D* ^ **EF-**

AB+D*EF-^

تبدیل infix به prefix

مثال:

((A+B)*D)^(E-F)

(+**AB** *D) ^ (E-F)

***+ABD** ^(E-F)

***+ABD** ^ **-EF**

^*+ABD-EF

تبدیل prefix به infix با استفاده از پشته

مثال :

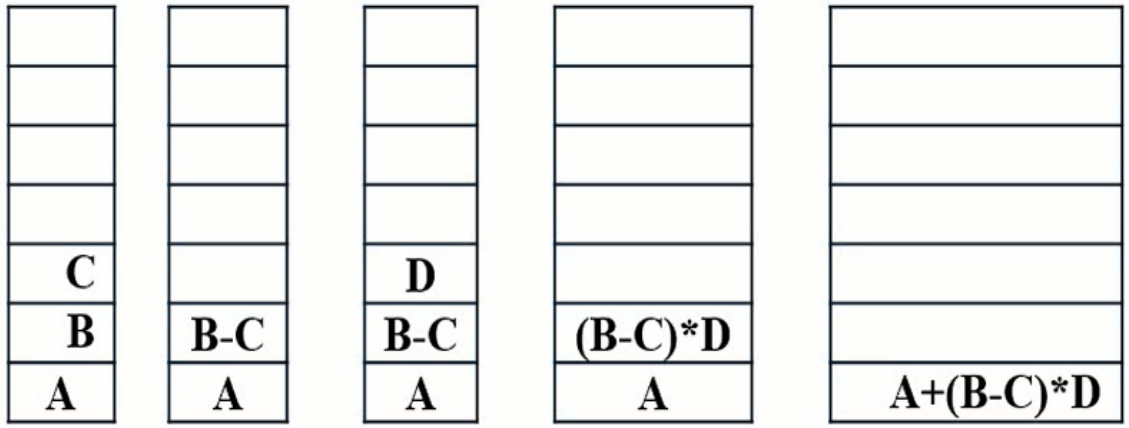
/+ a * b c d

(a + b * c) / d	/	a + b*c	*
	d	a	a
		+	+
		/	/

تبدیل postfix به infix با استفاده از پشته

مثال :

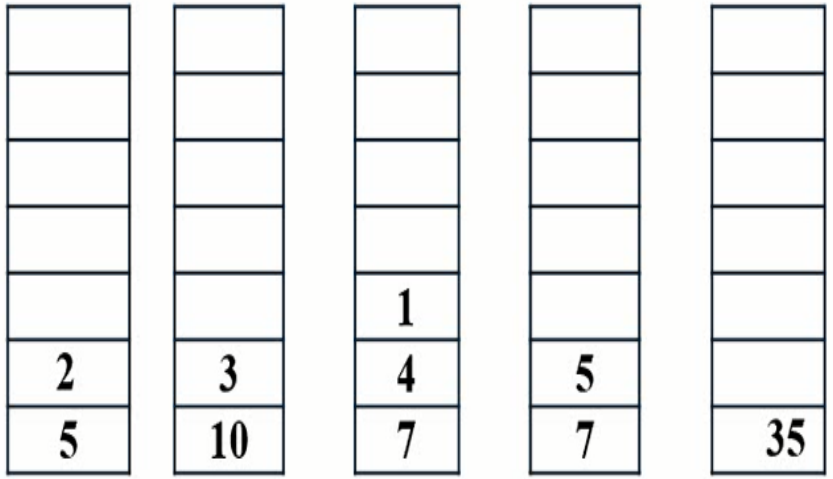
A B C – D * +



مثال

حاصل عبارت زیر را مشخص کنید.

5, 2, *, 3, -, 4, 1, +, *



پایان فصل چهارم

فصل پنجم

گراف

تعریف

گراف $G = (V,E)$، شامل دو مجموعه V و E است.

V : مجموعه محدود و غیرتهی از رئوس

E : مجموعه‌ای از زوج رئوس (یال)

انواع گراف

۱- **جهت‌دار** :

ترتیب گره‌ها در زوج مرتب اهمیت داشته باشد،

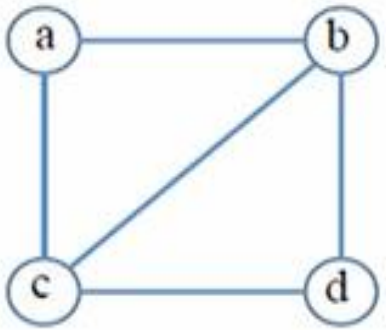
۲- **غیر جهت‌دار** :

گرافی که در آن جای دو رأس در مجموعه یالها اهمیت نداشته باشد.

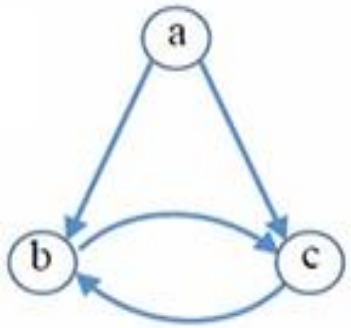
تعریف

مثال

- ***Undirected***
- V={a,b,c,d}
- E={{a,b}, {a,c}, {b,c}, {b,d}, {c,d}}

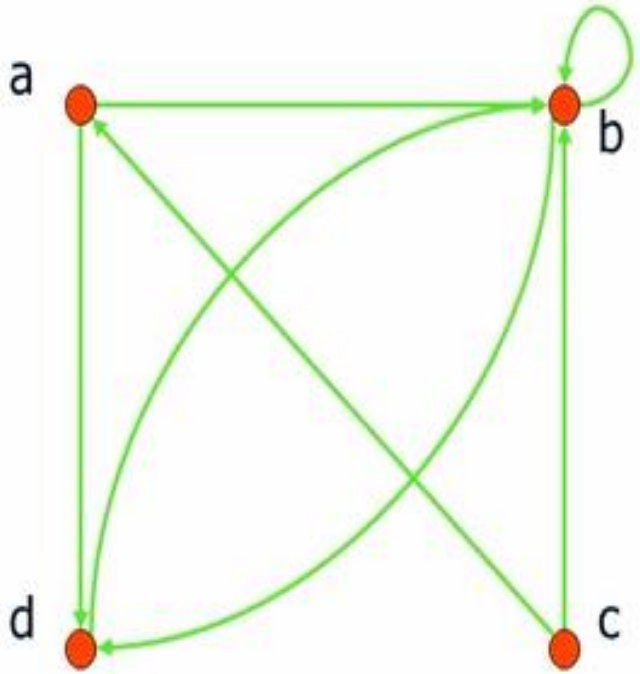


- ***Directed***
- V = {a,b,c}
- E = {(a,c), (a,b) (b,c), (c,b)}



درجه گراف

مثال



$\deg^-(a) = 1$
$\deg^+(a) = 2$

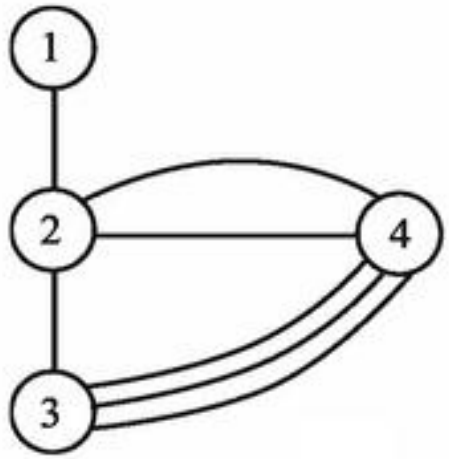
$\deg^-(b) = 4$
$\deg^+(b) = 2$

$\deg^-(d) = 2$
$\deg^+(d) = 1$

$\deg^-(c) = 0$
$\deg^+(c) = 2$

گراف چند گانه

گرافی که در آن یالهای چندگانه (Multi Edge) مجاز باشد.

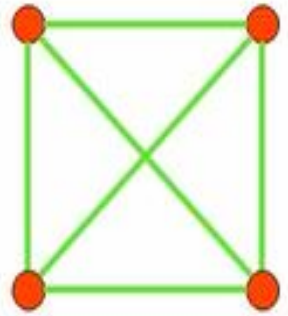


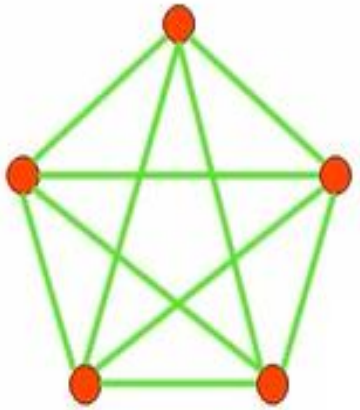
گراف کامل

گراف بدون جهتی که همه یالهای آن رسم شده باشد.

درجه هر یک از گره ها : n-1

تعداد یالهای: $\frac{n(n-1)}{2}$

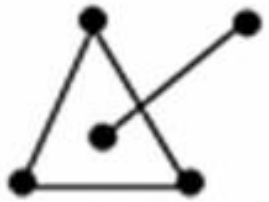




گراف همبند

گرافی که یک مسیر بین هر دو گره آن وجود داشته باشد.

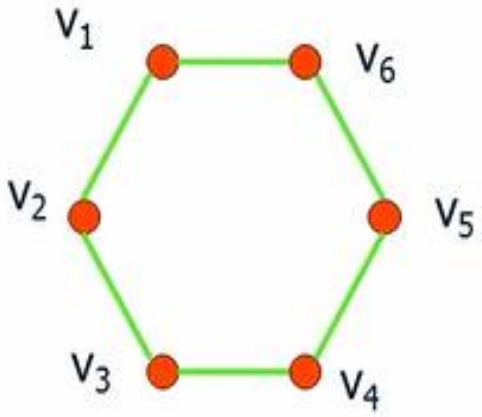
یک گراف ناهمبند :

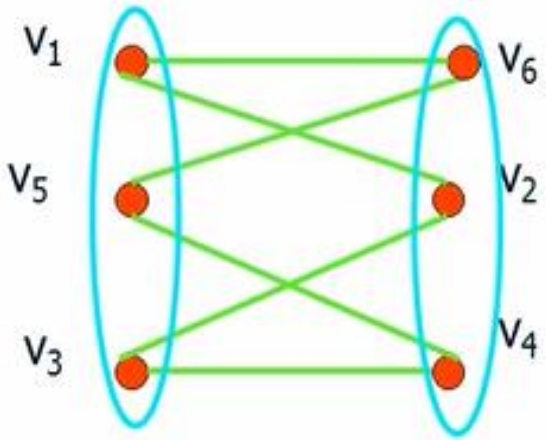


گراف همبند قوی : گراف جهت‌داری که برای هر زوج گره **V,U** هم یک مسیر از **U** به **V** و هم یک مسیر از **V** به **U** وجود داشته باشد.

گراف دو قسمتی(Bipartite)

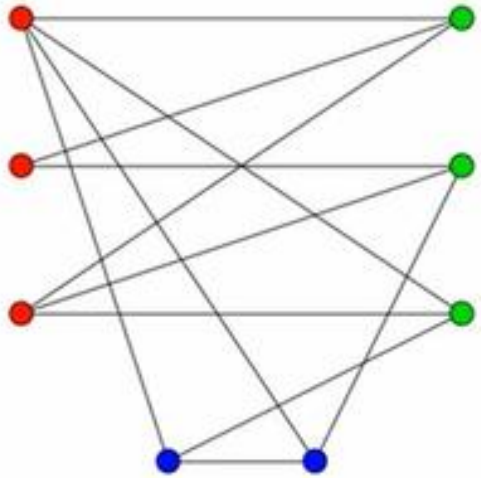
گرافی که نودهای آن قابل دسته بندی به دو مجموعه مستقل Uو Vهستند. هر یال در این گراف نودی از مجموعه Uرا به نودی در مجموعه V وصل می کند. این گراف را معمولا به صورت G=(V,U,E) نمایش می دهند.





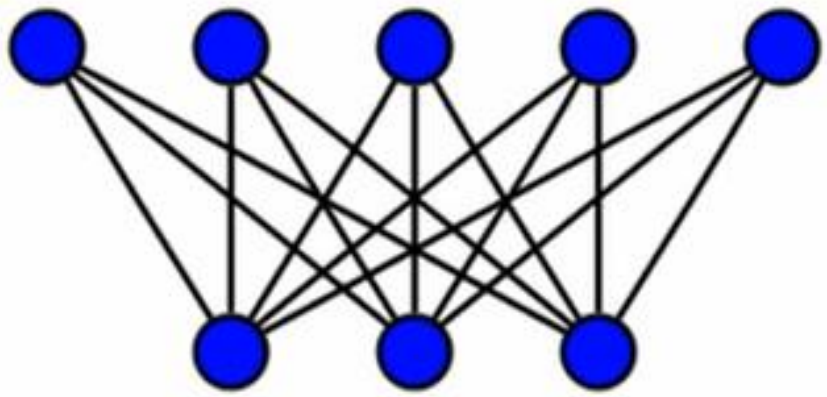
مثال

- گراف زیر دو قسمتی نیست. چون دارای سیکل به طول فرد دارد.



گراف دو قسمتی کامل

- مثال: $m=5,n=3$



زیرگراف (subgraph)

یک زیر گراف از گراف $G=(V,E)$ ، یک گراف $H=(W,F)$ است که :

$W \subseteq V$

and

$F \subseteq E$

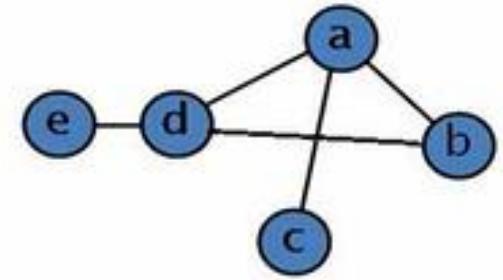
نمایش گراف

۱- ماتریس همجواری (Adjacency matrices)

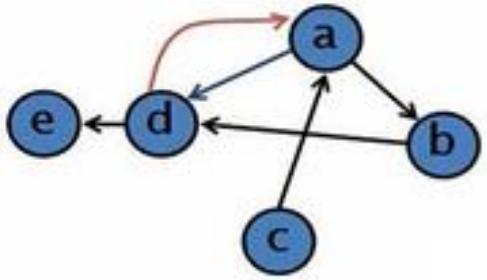
۲- لیست همجواری (Adjacency List)

ماتریس همجواری

مثال



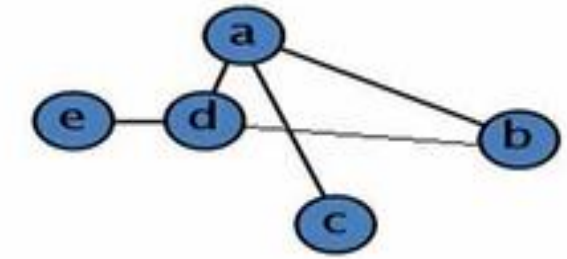
	a	b	c	d	e
a	0	1	1	1	0
b	1	0	0	1	0
c	1	0	0	0	0
d	1	1	0	0	1
e	0	0	0	1	0

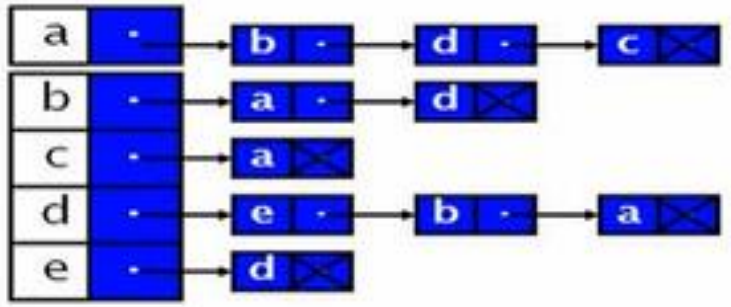


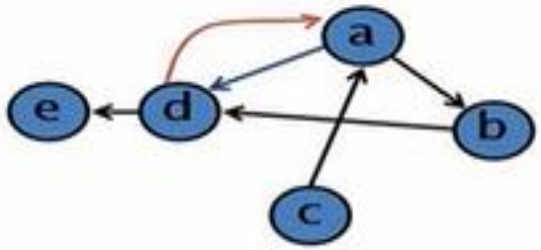
	a	b	c	d	e
a	0	1	0	1	0
b	0	0	0	1	0
c	1	0	0	0	0
d	1	0	0	0	1
e	0	0	0	0	0

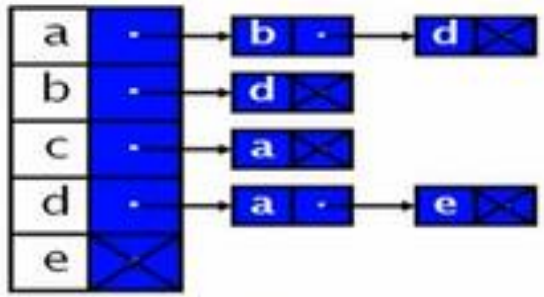
لیست همجواری

لیست همجواری



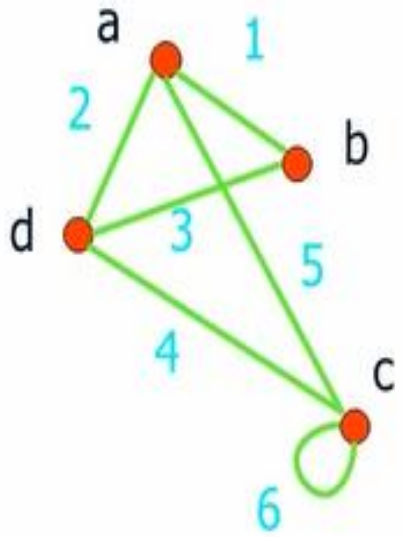






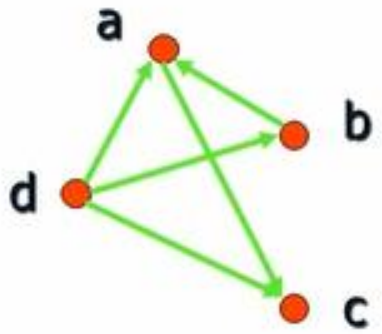
مثال (ماتریس برخورد)

این ماتریس برای یک گراف بدون جهت، یک ماتریس با $|V|$ سطر و $|E|$ ستون می باشد، چنانچه اگر لبه i راس j را تلاقی کند،آنگاه درایه واقع در سطر i و ستون j برابر 1 خواهد بود.



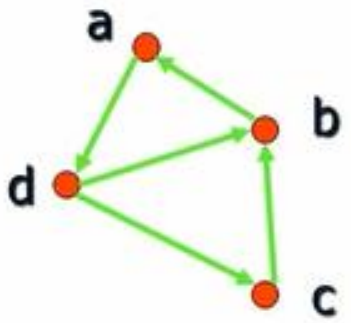
$$M = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Connectivity



Weakly connected

no path from b to d.



Strongly connected

پیمایش گراف

۱- عمقی (پیمایش اول ـ عمق) (DFS : Depth First Search)

۲- سطحی (پیمایش اول- عرض) (BFS : Breadth First Search)

در پیمایش DFS از پشته و در پیمایش BFS از صف استفاده می‌شود.

پیمایش سطحی(BFS)

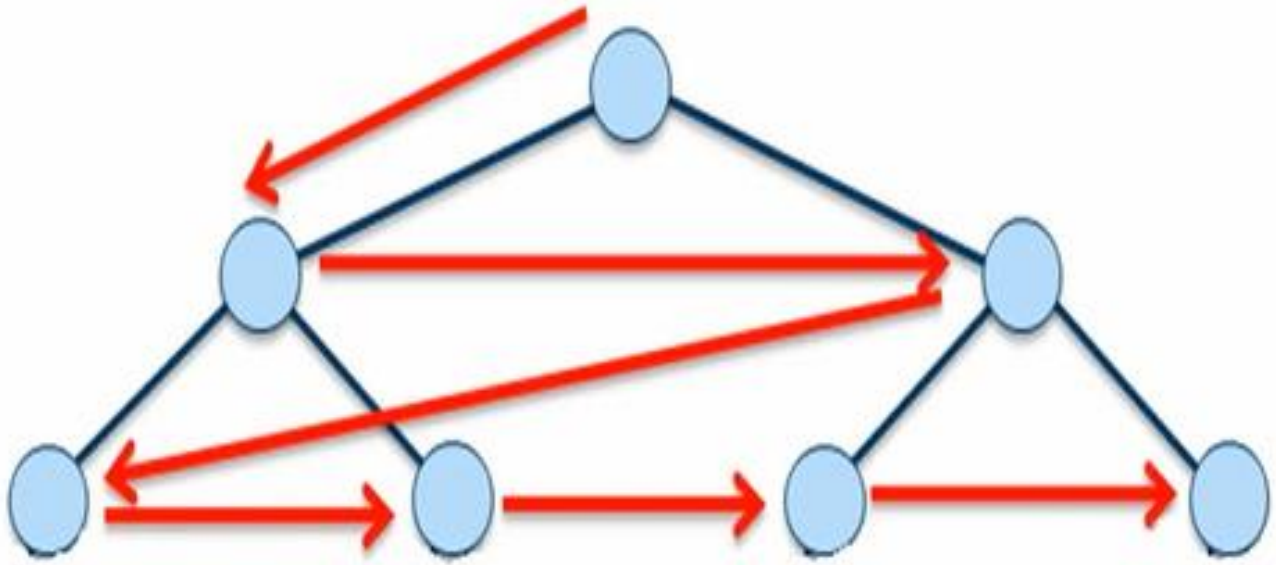
با شروع از یک گره، ابتدا گره را وارد صف کرده و سپس کلیه گره‌های مجاور با آن (فرزندان گره از چپ به راست) را در صف درج می کنیم.

حال عنصر بعدی از صف را حذف کرده و گره های مجاور آن را درج می کنیم.

این عمل را ادامه داده تا همه گره ها پیمایش شوند.

پیمایش سطحی منحصر به فرد نیست.

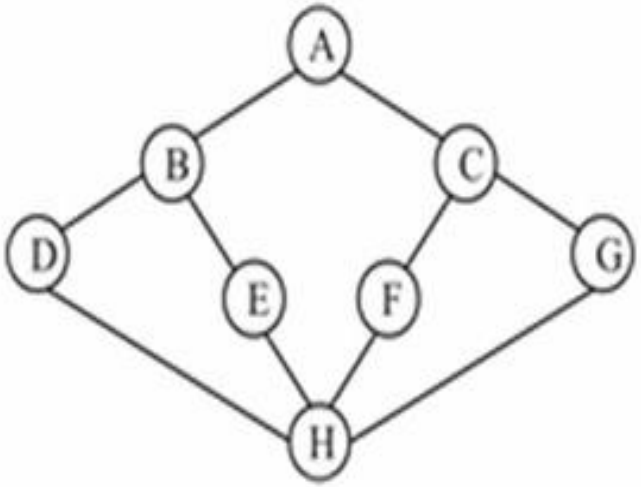
پیمایش سطحی



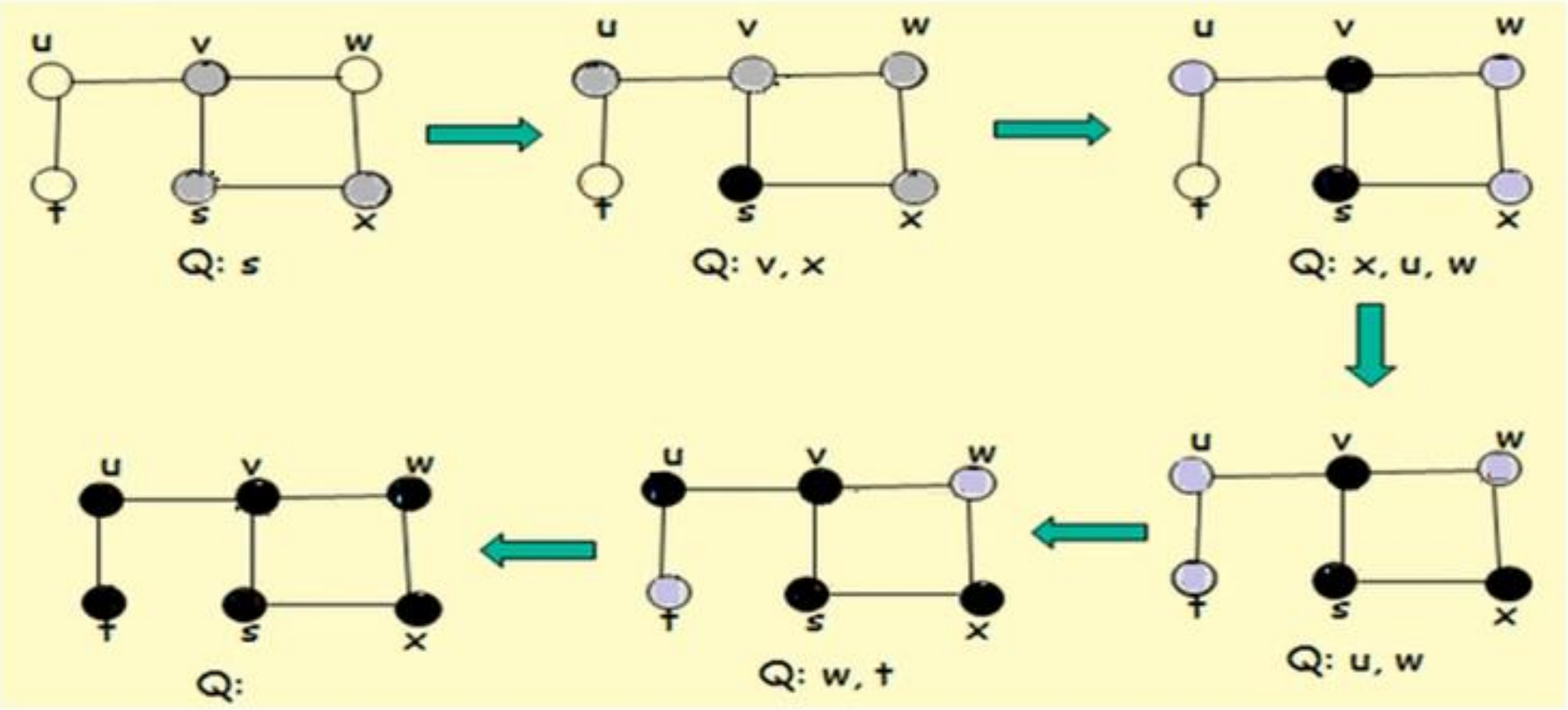
مثال

یکی از پیمایش های BFS گراف زیر را مشخص کنید.

A B C D E F G H



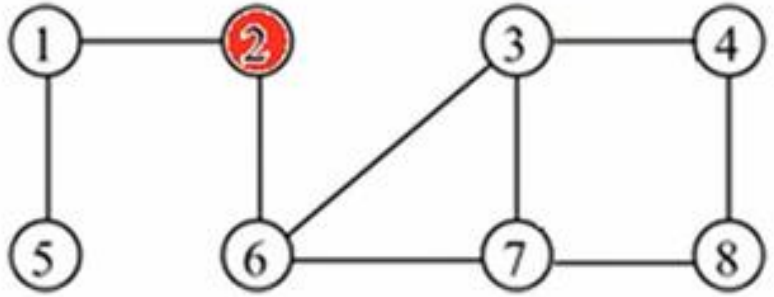
مثال



مثال

یکی از پیمایش های BFS گراف زیر با شروع از گره شماره ۲ را مشخص کنید.

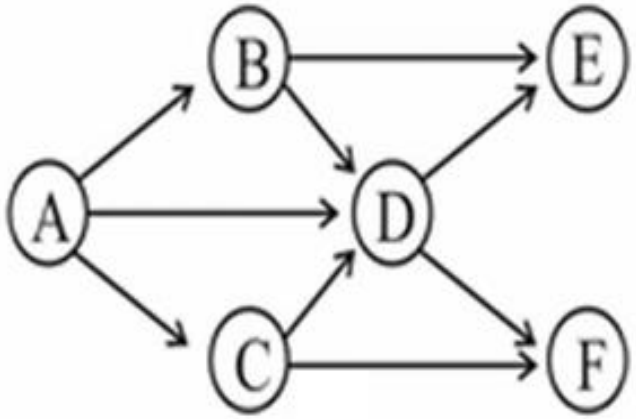
2		
6	1	
1	3	7
3	7	5
7	5	4
5	4	8
4	8	
8		



2,6,1,3,7,5,4,8

مثال

چند پیمایش BFS برای گراف زیر مشخص کنید.



A B C D E F

A D B C E F

A D B C F E

تابع پیمایش سطحی

```
bfs(v){
  front = rear = NULL;
  cout << v;
  visited[v] = TRUE;
  addq ( &front , &rear , v );
  while ( front) {
      v = deleteq (&front);
      for ( w = graph[v] ; w ; w = w->link )
       if ( !visited[w -> vertex] ) {
          cout << w -> vertex;
          addq ( &front , &rear , w -> vertex );
          visited[ w -> vertex ]=TRUE;
}}}
```

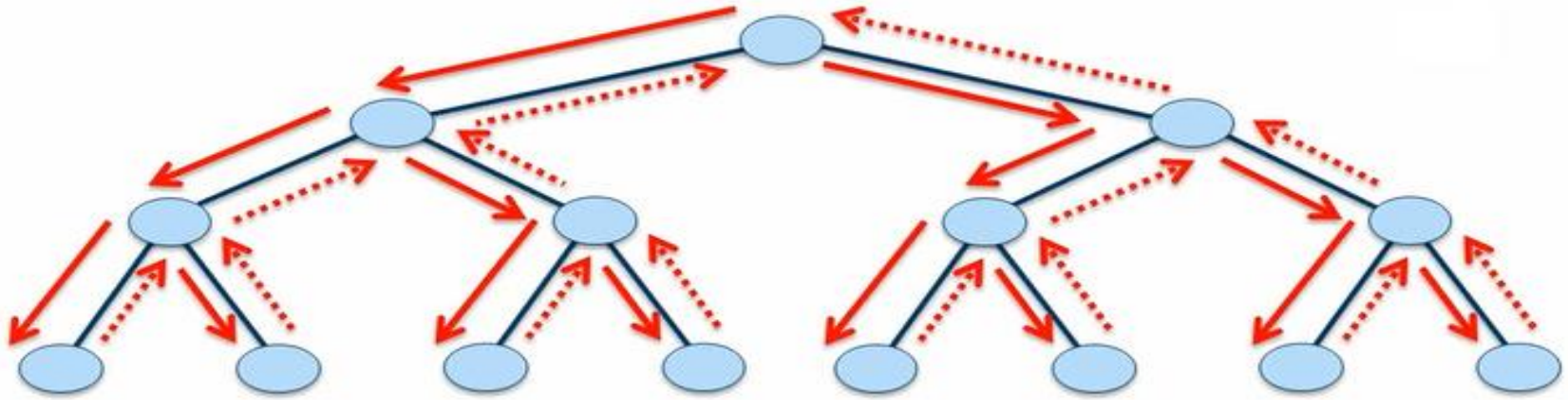
پیمایش عمقی (DFS)

با شروع از یک گره، آن را ملاقات کرده و سپس کلیه گره‌های سمت چپ را تا آخرین عمق پیمایش می کنیم.

اگر در پایین رفتن های متوالی، گره‌ای ملاقات شود که هیچ گره همجواری نداشته باشد یا تمام گره‌های همجوار آن ملاقات شده باشد، برگشت به یک سطح بالا انجام می‌گیرند و روند فوق تکرار می‌شود.

پیمایش عمقی منحصر به فرد نیست.

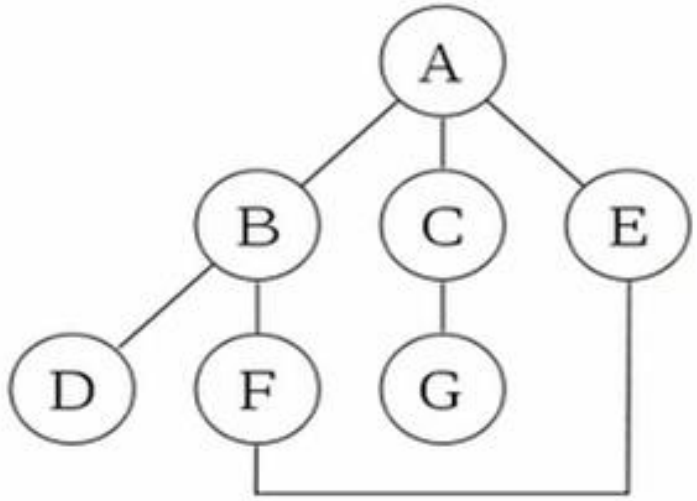
Depth-First Search (DFS)



مثال

یکی از پیمایش های DFS گراف زیر را مشخص کنید.

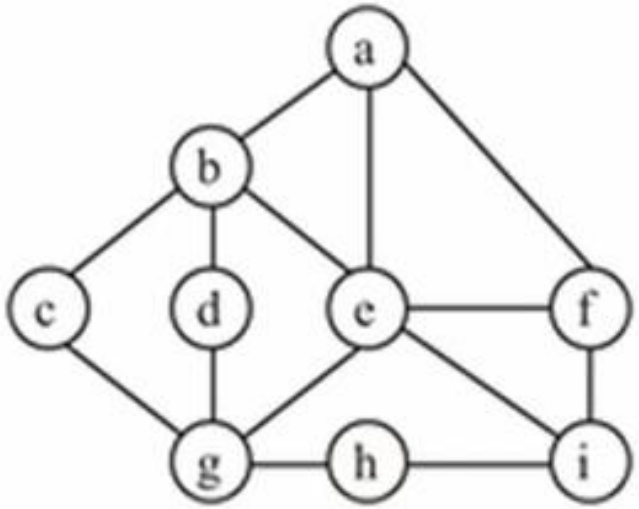
A B D F E C G



مثال

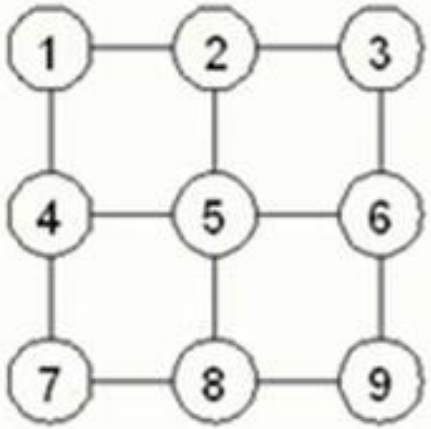
یکی از پیمایش های DFS گراف زیر را مشخص کنید.

a b c g d e f i h



مثال

سه پیمایش DFS برای گراف زیر مشخص کنید.



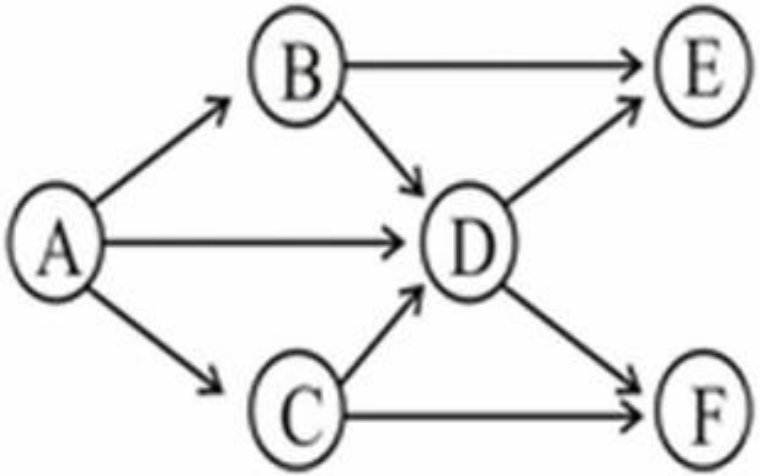
1,4,7,8,9,6,5,2,3

1,2,3,6,9,8,7,4,5

1,2,3,6,5,4,7,8,9

مثال

چند پیمایش DFS برای گراف زیر مشخص کنید.



A B E D F C

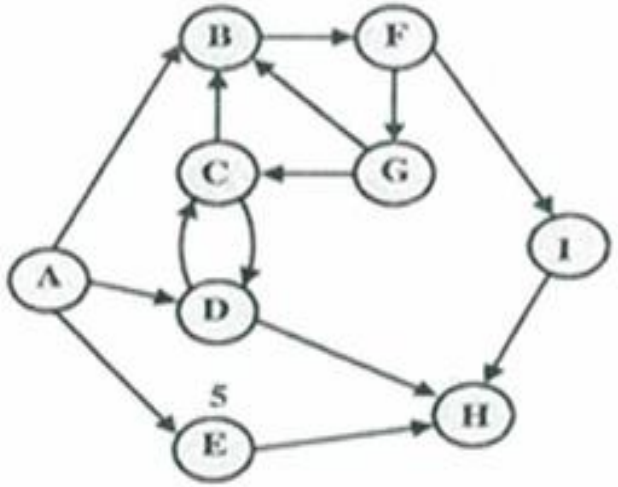
A B D F E C

A D E F B C

مثال

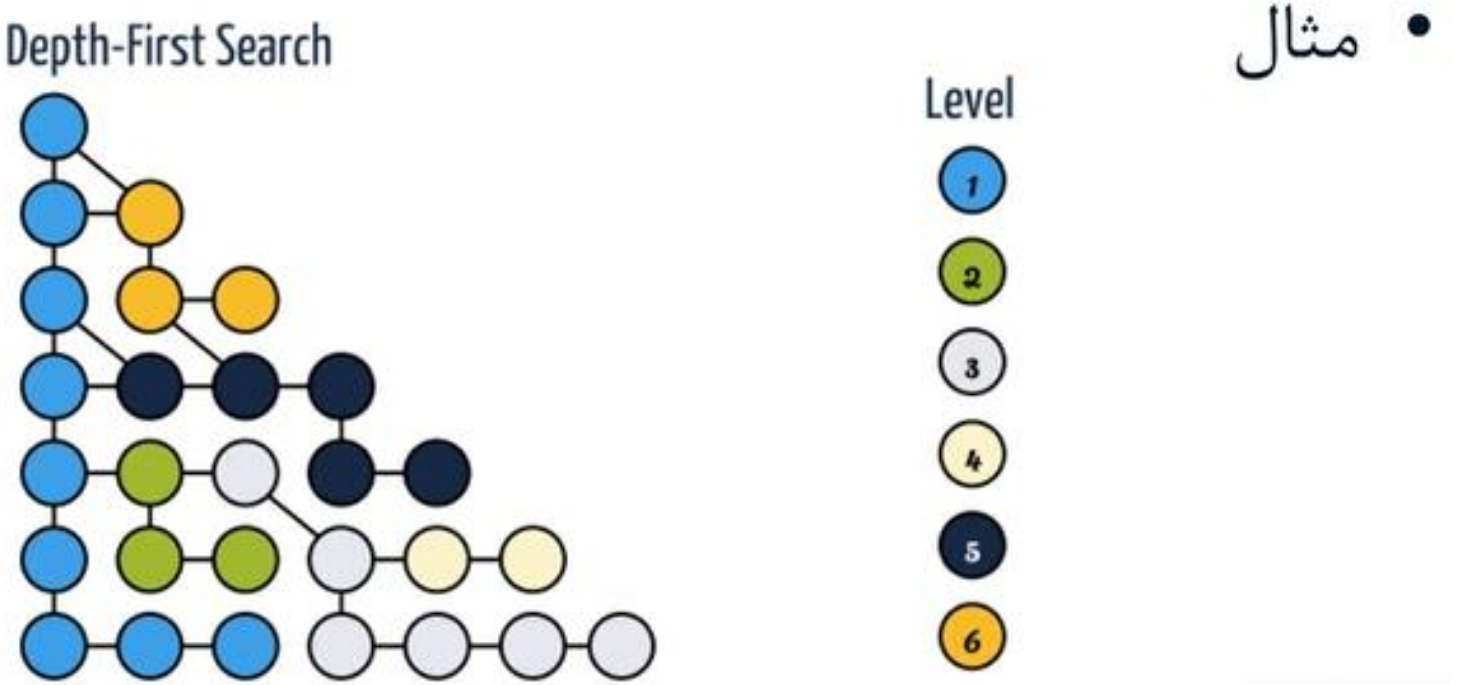
یکی از پیمایش‌های DFS گراف زیر را به دست آورید.

A B F I H G C D E



مثال

- مثال



تابع پیمایش عمقی

```
dfs (v)
{
    visited[v] = TRUE;
    cout << v;
    for ( w = graph[v] ; w ; w = w->link )
        if ( ! Visited [w -> vertex] )
            dfs ( w -> vertex );
}
```

درخت پوشای کمینه

درخت پوشا(فراگیر) برای یک گراف:
یک زیر گراف متصل است که حاوی همه رئوس موجود در گراف بوده و یک درخت باشد یعنی چرخه نداشته باشد.

درخت پوشای کمینه : درخت پوشا با وزن کمینه (MST)

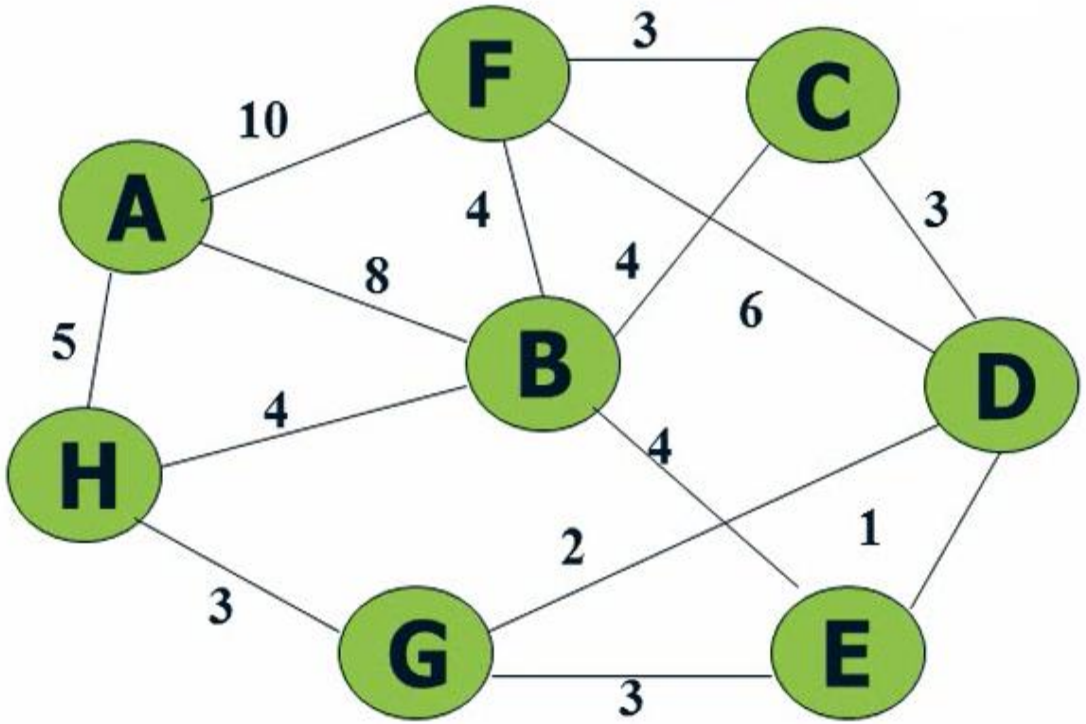
الگوریتم های تعیین MST

۱- کروسکال

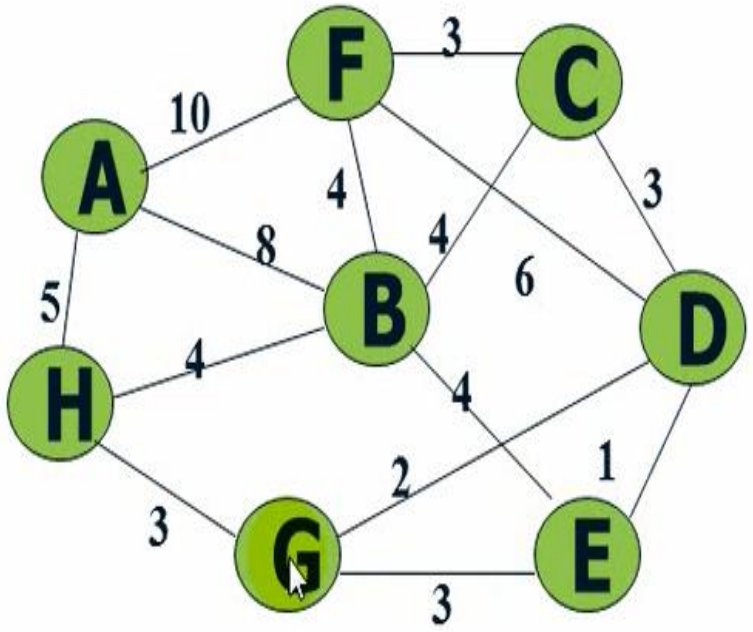
۲- پریم

مثال(کروسکال)

مثال

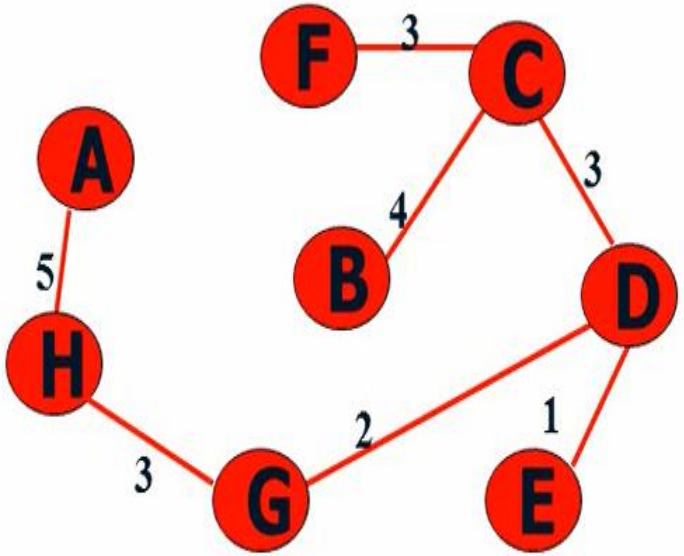


مرتب کردن صعودی یال ها



edge	d_v	
(D,E)	1	
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



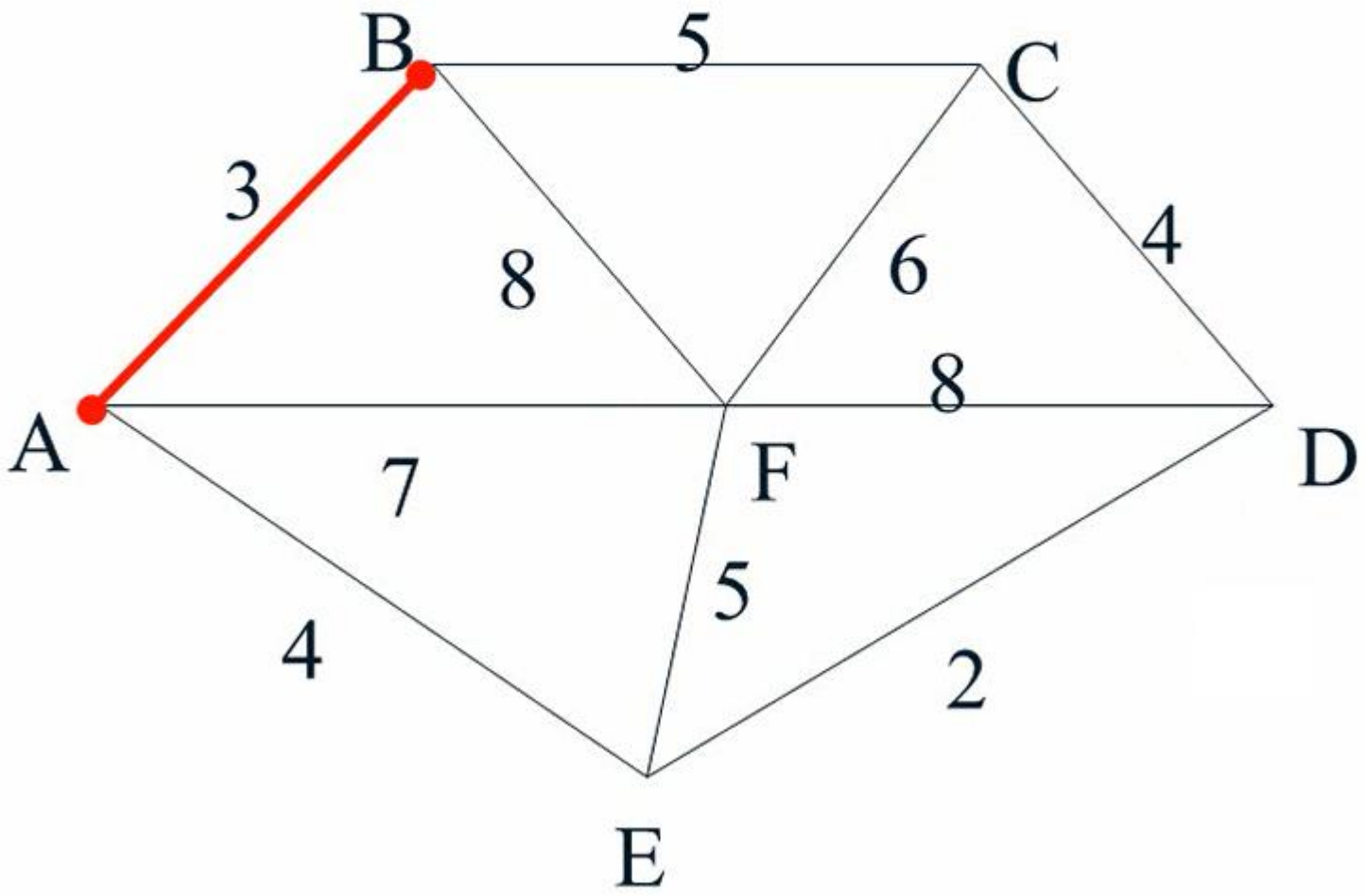
edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

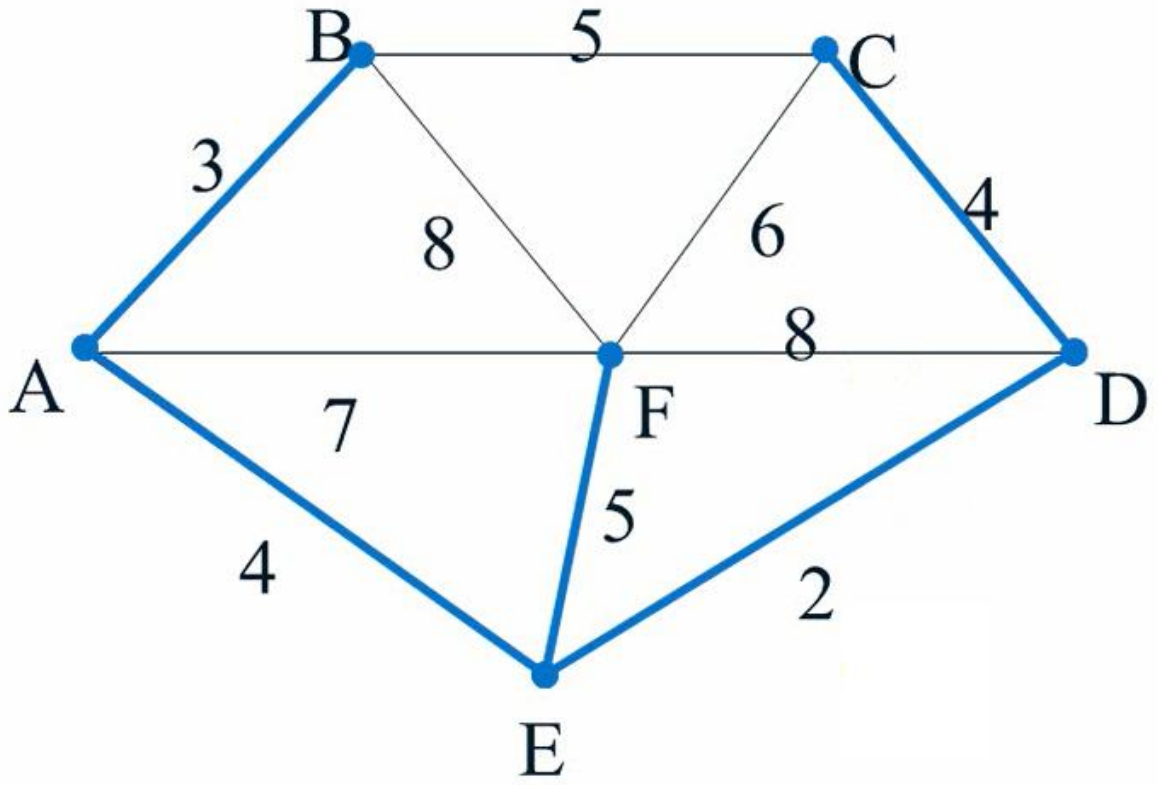
edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	
(A,B)	8	
(A,F)	10	

not considered

Total Cost = 21

مثال (پریم)





Total weight of tree: 18

پایان فصل پنجم