- **روش دوم الگوریتم پریم (prim)** : در این روش از یک رأس شروع می‌کنیم و کمترین یال (یال با کمترین وزن) که از آن می‌گذرد را انتخاب می‌کنیم. در مرحله بعد یالی انتخاب می‌شود که کمترین وزن را در بین یالهایی که از دو گره موجود می‌گذرد داشته باشیم. به همین ترتیب در مرحله بعد یالی انتخاب می‌گردد که کمترین وزن را در بین یالهایی که از سه گره موجود می‌گذرد داشته باشد. این روال را آنقدر تکرار می‌کنیم تا درخت پوشای بهینه حاصل شود. باید توجه کرد که یال انتخابی در هر مرحله در صورتی انتخاب می‌شود که در گراف دور ایجاد نکند. تفاوت روش پریم با روش کراسکال در این است که گراف حاصل در مراحل میانی تشکیل درخت پوشای بهینه در روش پریم همیشه متصل است ولی در الگوریتم کراسکال در آخرین مرحله قطعاً متصل است.
- **روش سوم الگوریتم سولین** : در الگوریتم سولین برای هر گره یال با کمترین هزینه که از آن عبور می‌کند را رسم می‌کنیم. در مرحله بعد ، گراف به مؤلفه‌هایی تقسیم می‌شود و یالی انتخاب می‌گردد که با کمترین هزینه دو مؤلفه گراف را به همدیگر متصل نماید با شرط عدم وجود دور در گراف. آنقدر این مراحل را ادامه می‌دهیم تا درخت پوشای بهینه حاصل شود.

تمرین : درخت پوشای بهینه گراف زیر را به سه حالت رسم نمائید :

✓ ac = 2 , ✓ de = 2 , ✓ gh = 2

✓ be = 3

✓ eh = 4

✓ ab = 5 , ✓ gi = 5

× hi = 6

✓ ef = 7

× bc = 8

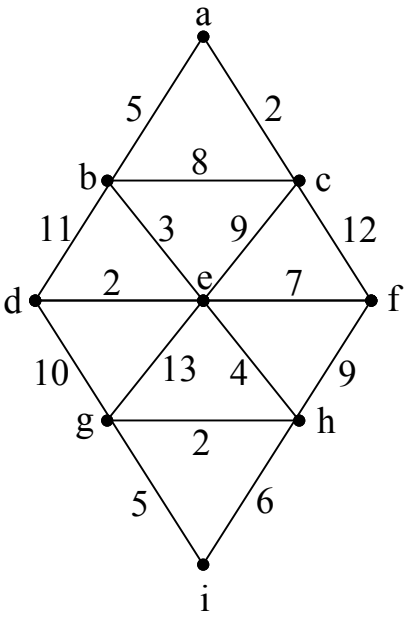
× fh = 9 , × ce = 9

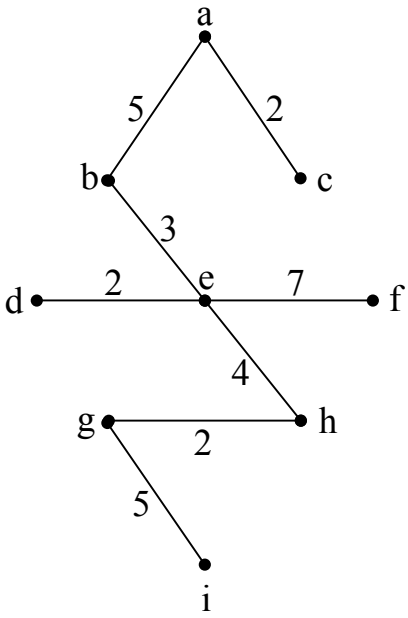
× dg = 10

× bd = 11

× cf = 12

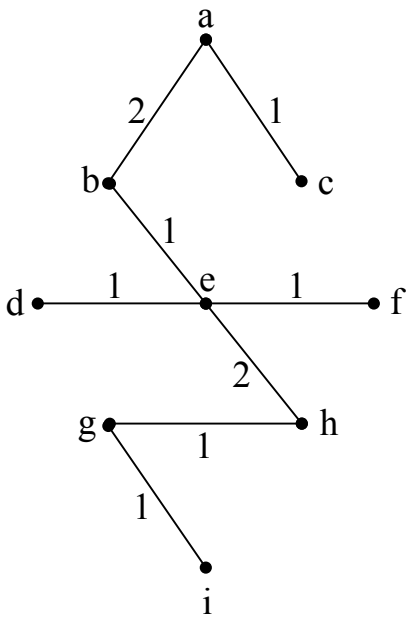
× eg = 13



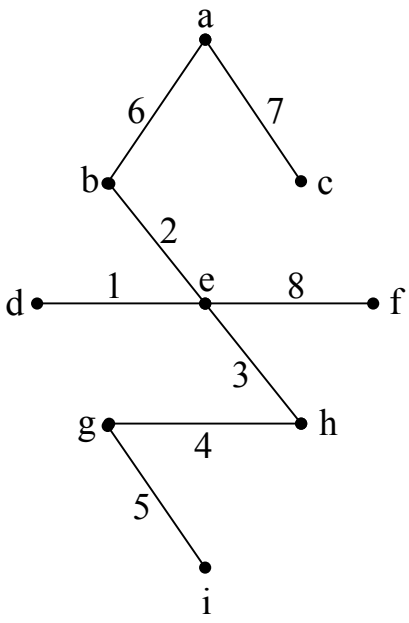


کراسکال

$2 + 2 + 2 + 3 + 4 + 5 + 5 + 7 = 30$



روش سولین (شروع از e)



روش پریم (شروع از e)

حداقل هزینه بین گره‌های گراف (الگوریتم دایکسترا)

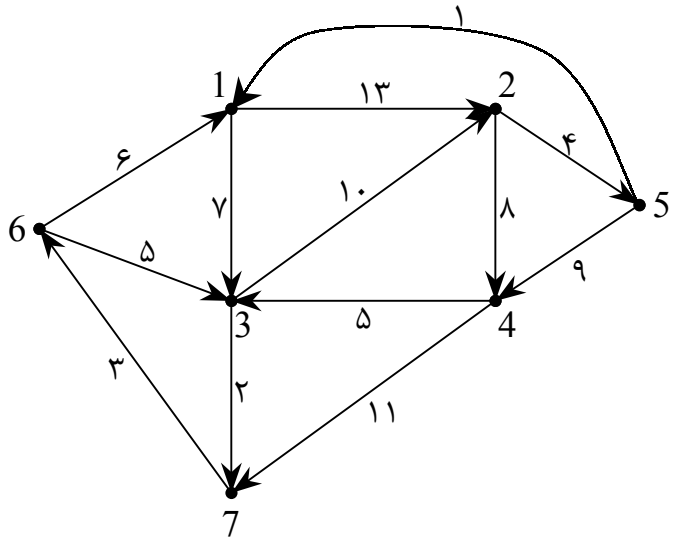
بـرای محاسـبه حداقـل هزیـنه‌هـا از یک گره به گره‌های دیگر در گراف وزن‌دار ، از الگوریتم دایکسترا اسـتفاده مـی‌کنیم. بدین منظور باید ابتا ماتریس هزینه‌های گراف را تشکیل دهیم و سپس با شروع از گـره مفـروض ، هزینه آن گره تا سایر گره‌ها را بدست آوریم. برای بدست آوردن هزینه حداقل بین دو گره دو انتخاب کلی وجود دارد :

۱- مسیر مستقیم بین دو گره W_{ij}

۲- استفاده از یک گره میانی $W_{ik} + W_{kj}$

آنقدر این روال را ادامه می‌دهیم تا تمام گره‌های گراف ملاقات شوند.

مثال : گراف زیر را در نظر بگیرید. از گره شماره ۱ شروع کرده و حداقل هزینه بین گره‌ها را نسبت به گره ۱ بدست آورید.



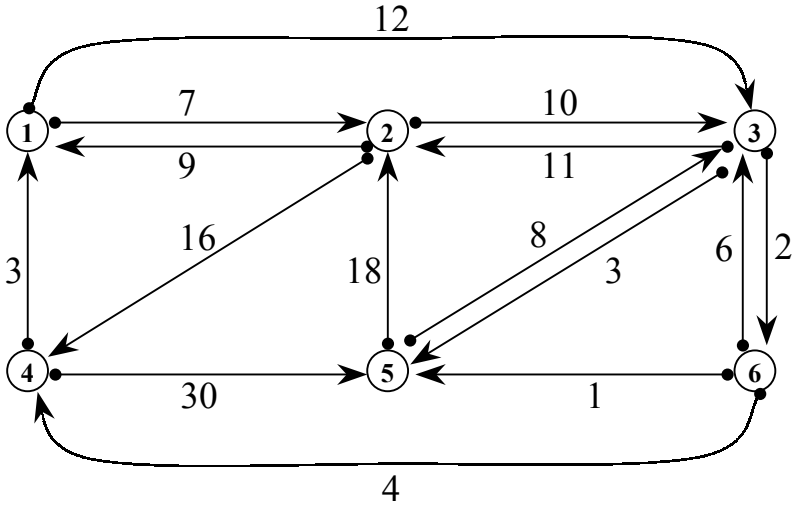
	1	2	3	4	5	6	7
1	0	13	7	∞	∞	∞	∞
2	∞	0	∞	8	4	∞	∞
3	∞	10	0	∞	∞	∞	2
4	∞	∞	5	0	∞	∞	11
5	1	∞	∞	9	0	∞	∞
6	6	∞	5	∞	∞	0	∞
7	∞	∞	∞	∞	∞	3	0

ماتریس

	✓ 1	✓ 2	✓ 3	✓ 4	✓ 5	✓ 6	✓ 7
1	0	13	**7**	∞	∞	∞	∞
3	0	13	7	∞	∞	∞	**9**
7	0	13	7	∞	∞	**12**	9
6	0	**13**	7	∞	∞	12	9
2	0	13	7	21	**17**	12	9
5	0	13	7	**21**	17	12	9
4							

جواب

مثال : حداقل هزینه بین گره‌های گراف زیر را در خصوص گره شماره ۲ محاسبه نمائید.



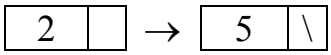
	1	2	3	4	5	6
1	0	7	12	∞	∞	∞
2	9	0	11	16	∞	∞
3	∞	10	0	∞	8	6
4	3	∞	∞	0	30	∞
5	∞	18	3	∞	0	∞
6	∞	∞	2	4	1	0

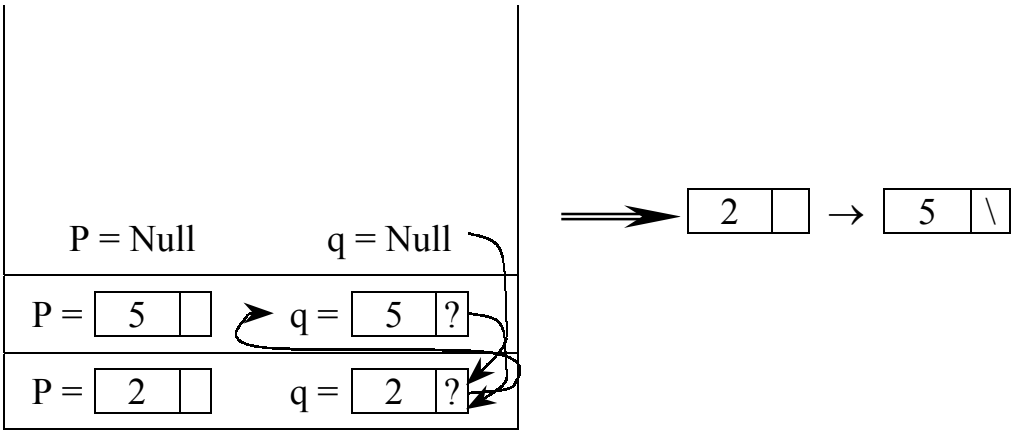
	✓ 1	✓ 2	✓ 3	✓ 4	✓ 5	✓ 6
2	**9**	0	11	16	∞	∞
1	9	0	**11**	16	∞	∞
3	9	0	11	**16**	19	17
4	9	0	11	16	19	**17**
5	9	0	11	16	**18**	17

سؤال : تابع f چه عملی انجام می‌دهد؟

```
node * f( node  *p )
{
        node  *q ;
        q = Null ;
        if ( p )
        {
                q = New ( node ) ;
                q → data = p → data ;
                q = Next = f( p → Next ) ;
        }
        return q ;
}
```

جواب :





تابع بالا از لیست پیوندی کی کپی تهیه می‌کند.

سؤال : عبارت prefix زیر را بصورت postfix بنویسید.

+ + a / b – c d / – a b – + c × d 5 / a – b c

جواب :

a b c d – / + a b – c d 5 × + a b c – / – / +

سؤال : حاصل عبارت postfix زیر را بنویسید.

6 , 2 , 3 , + , – , 3 , 8 , 2 , / , + , × , 2 , ↑ , 3 , +

جواب :

(6 – (2 + 3)) × (3 + (8 / 2)) ↑ 2 + 3 = 52

سؤال : عبارت زیر را بصورت postfix و prefix بنویسید.

(a / (b – c + d)) × (e – a) × c

جواب :

Postfix = a b c – d + / e a – × c ×

Prefix = × × / a + – b c d – e a c

سؤال : خروجی تابع f را بنویسید.

```
void  f( node  *x )
{
        node  *p ;
        int i ;
        i = 0 ;
        if ( x ! = Null )
        {
                p = x ;
                do
                {
                        i + + ;
                        p = p →Next ;
                }
                while ( p ! = x )
        }
        C out << i ;
}
```

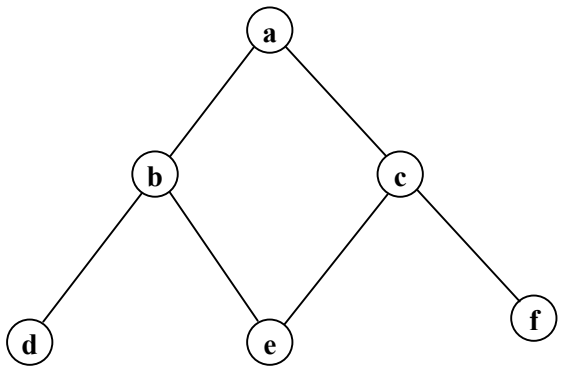
جواب : تعداد ندهای لیست پیوندی چرخشی را محاسبه و چاپ می‌کند.

سؤال : خروجی تابع g را بنویسید.

```
node  *g (node  *p )
{
        node  *m , *L ;
        m = Null ;
        while ( p )
        {
                L = m ; m = p ;
                p = p → Next ;
                m → Link = L ;
        }
        return m ;
}
```

جواب : لیست پیوندی را معکوس می‌کند.

سؤال : الف) از گره a شروع کرده و پیمایش‌های dfs گراف زیر را بنویسید.



a b d e c f

a b e c f d

a c f e b d

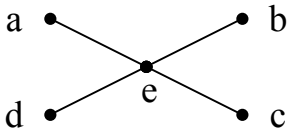
a c e b d f

ب) آیا می‌توان یک dfs و یک bfs در این گراف نوشت که با هم یکی باشند؟

جواب : خیر

ج) در حالت کلی گراف‌ها آیا می‌توان یک dfs و bfs نوشت که با هم برابر باشند یا خیر؟

جواب : بله مثلاً اگر در گراف زیر از گره e شروع کنیم dfs و bfs آن با هم یکی خواهد شد.



dfs = e a b c d

bfs = e a b c d

سؤالات میان ترم

۱- آرایـه‌ای ۱۱ عنصـری بشـکل زیـر موجود است. می‌خواهیم آن را به روش درجی مرتب کنیم. آرایه در مرحله پنجم پویش آن چگونه خواهد بود. (۱۰ نمره)

14	7	3	20	18	4	17	9	11	30	25
1	2	3	4	5	6	7	8	9	10	11

جواب :

3	4	7	14	18	20	17	9	11	30	25
1	2	3	4	5	6	7	8	9	10	11

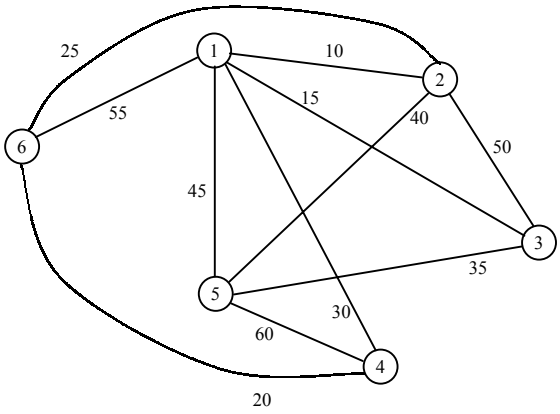
۲- زیربرنامه‌ای بنویسید که آدرس شروع دو لیست پیوندی مرتب را گرفته و آدرس شروع لیست پیوندی مرتب حاصل از ترکیب دو لیست پیوندی داده شده را برگرداند. (۲۰ نمره)

```
node ordermerg ( node * start 1 , node * start 2 )
{
        node * p , * q , * start , * s ;
        s = new ( node ) ; s → next = Null ;
        start = s ;
        p = start 1 → next ;
        q = start 2 → next ;
        while ( p && q )
        if ( p → data < q → data )
        {
                s → next = p ;
                s = p ;
                p = p → next ;
        }
        else
        {
                s → next = q ;
                s = q ;
                q = q → next ;
        }
        if ( p ) s → next = p ;
        else     s → next = q ;
        return start ;
}
```

۳- دو لیست پیوندی با آدرس‌های شروع Start 1 و Start 2 داریم. زیربرنامه‌ای بنویسید که آدرس شروع لیست پیوندی ترکیب این دو لیست را برگرداند. (۱۰ نمره)

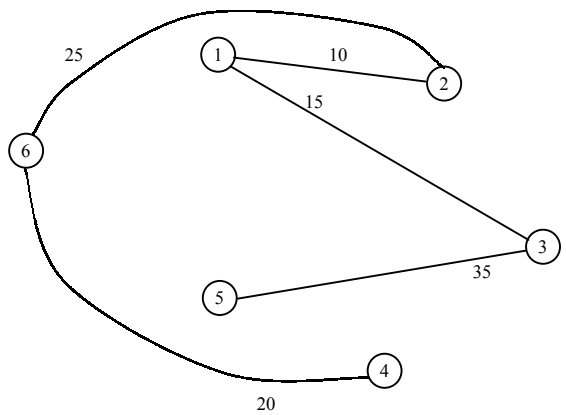
```
node * concatlist ( node * start 1 , node * start 2 )
{
        node * p ;
        p = start 1 → next ;
        while ( p → next )  p = p → next ;
        p → next = start 2 → next ;
        return start 2 ;
}
```

۴- هزینه درخت پوشای مینیمم گراف زیر را از روش پریم بدست آورید. درخت حاصل چند یال دارد؟ ترتیب رسم یالهای درخت پوشای بهینه را با استفاده از الگوریتم پریم ذکر کنید. (۱۵ نمره)



4 – 5 : 60
4 – 6 : 20
4 – 1 : 30
1 – 5 : 45
1 – 2 : 10
1 – 3 : 15
1 – 6 : 55
5 – 5 : 40
2 – 6 : 25
2 – 3 : 50
3 – 5 : 35

جواب :



1) 1 – 2 : 10
2) 2 – 3 : 15
3) 2 – 6 : 25
4) 6 – 4 : 20
5) 3 – 5 : 35

} = 105

۵- زیربرنامه‌ای بنویسید که آرایه‌ای از اعداد را به صورت انتخابی (selection sort) مرتب کند. یک آرایه مرتب شده را توسط کدام یک از روشهای مرتب‌سازی گفته شده مجدداً مرتب کنیم تا کندتر مرتب‌سازی انجام شود. (۲۰ نمره)

```
for (i = n ; i > 1 ; – – 1)
{
        max = A[1] ;
        index = 1 ;
        for (i = 2 ; j < = i ; + + j)
        if (A[j] > max)
        {
                max = A[j] ;
                index = j ;
        }
        A[index] = A[i] ;
        A[i] = max ;
}
```

جواب قسمت دوم : اگر یک آرایه مرتب شده داشته باشیم و با مرتب‌سازی درجی یا حبابی آن را مجدداً مرتب کنیم بهترین حالت مرتب‌سازی را انتخاب کرده‌ایم ولی اگر مرتب‌سازی سریع را انتخاب کنیم کندترین حالت را انتخاب کرده‌ایم. حال اگر یک آرایه نامرتب داشته باشیم بهترین حالت برای مرتب‌سازی حالت مرتب‌سازی سریع و یا ادغامی است.

۶- یک آرایه دو بعدی ۲۰ × ۱۰ را بصورت ستونی در حافظه از محل ۱۰۰۰ حافظه ذخیره کرده‌ایم. در صورتیکه هر عنصر آرایه ۲ بایت فضا مصروف کند آدرس عنصر [7 , 6] آرایه در حافظه چیست؟ (۵ نمره)

$[6,7]=[(6\times 10)+5]\times 2+1000=1130$ روش ستونی

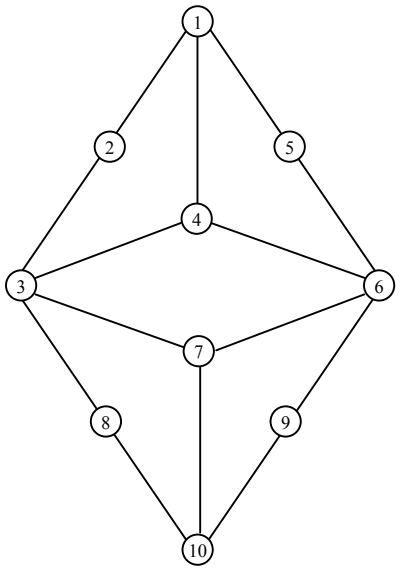
$[6,7]=[(5\times 20)+6]\times 2+1000=1212$ روش سطری

۷- آرایه اعداد در سؤال ۱ را با استفاده از الگوریتم مرتب‌سازی سریع (Quick sort) مرتب می‌کنیم. در مرحله اول مرتب‌سازی (پویش اول آرایه) ، آرایه به چه شکل خواهد بود؟ (۱۰ نمره)

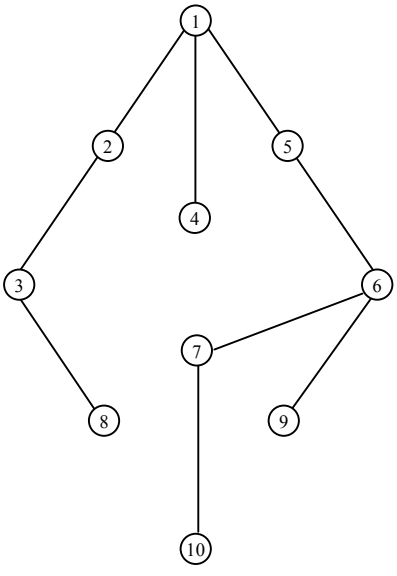
14	7	3	20	18	4	17	9	11	30	25
محور			i_1	i_2	J_3	i_3	j_2	j_1		
4	7	3	11	9	14	17	18	20	30	25

۸- پـیمایش اول عمـق و اول سـطح گراف زیر را بنویسید و درخت پوشای bfs و dfs هر یک را تشـکیل دهـید. پـیمایش اول سـطح را از گره ۵ و پیمایش اول عمق را از گره ۹ شروع کنید. (۲۰ نمره)

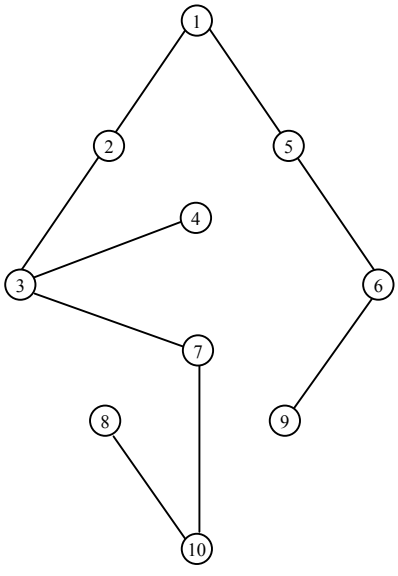
نکته : از بین سؤالات ۱ و ۷ تنها به یکی پاسخ دهید. همه مرتب‌سازی‌ها بصورت صعودی است.



bfs = 5 1 6 2 4 7 9 3 10 8



dfs = 9 6 5 1 2 3 4 7 10 8

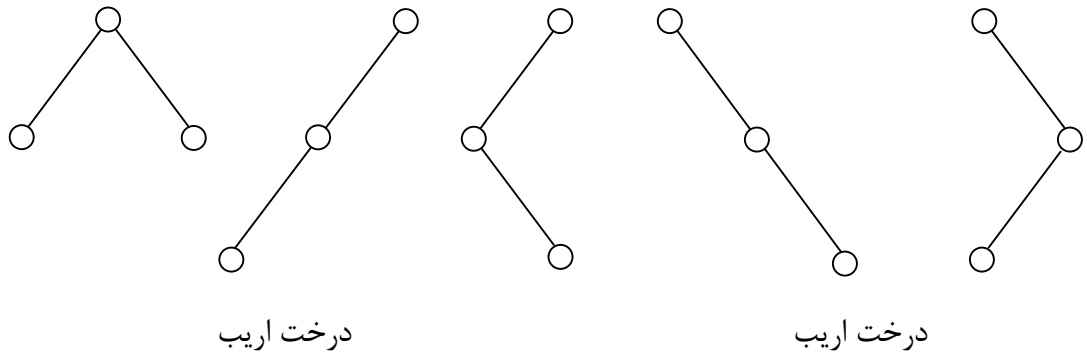


درخت (Tree)

درخت مجموعه‌ای است متناهی از یک یا چند گره که یک گره خاص را بنام ریشه مشخص کرده‌ایم و سایر گره‌ها به مجموعه‌های مجزایی تقسیم می‌شوند که هر مجموعه خود یک درخت است و زیر درخت ریشه نامیده می‌شود. تعداد زیر درخت‌های هر گره درجه آن گره است. فاصله هر گره تا ریشه درخت را سطح آن گره می‌نامند. بزرگترین درجه گره در درخت ، درجه درخت نامیده می‌شود. اگر درجه درخت m باشد درخت را m تایی می‌گویند. به گره‌هایی که درجه آنها صفر است برگ (Leaf) گفته می‌شود. برگها زیر درخت ندارند. برگها را گره‌های خارجی درخت و سایر گره‌ها غیر از برگها را گره‌های داخلی درخت می‌نامند. دو گره که دارای پدر مشترک هستند را گره‌های همزاد گویند. حداکثر سطح یک گره در درخت را ارتفاع (عمق) درخت گویند. پیش فرض سطح ریشه ۱ است.

درخت دودویی طبق تعریف درختی است که درجه آن ۲ باشد یعنی هر گره حداکثر ۲ فرزند داشته باشد. یکی فرزند سمت راست و یکی فرزند سمت چپ که با هم متفاوت (متمایز) هستند.

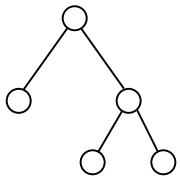
مثال : با سه گره چند درخت دودویی می‌توان ساخت.



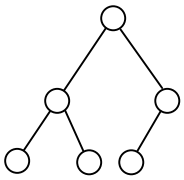
- **درخت Perfect (کاملاً پر)** : درختی است که همه گره‌ها بجز گره‌های سطح آخر (برگها) دارای حداکثر فرزندان بوده (حداکثر درجه درخت) و برگها هم سطح نیز باشند.
- **درخت Complete (کامل)** : درختی است که اگر گره‌های آنرا شماره‌گذاری کنیم ،

شـماره‌هـا بر درخت Perfect متناظرش منطبق باشند. یعنی می‌توان گفت درختی است که اگـر ارتفـاع آن d باشـد تـا ارتفـاع d - 1 ، درخـت Perfect بوده و برگها در سطح d تا حد ممکن در سمت چپ باشند.

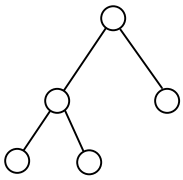
- **درخت Full (پر)** : درختی که در آن گره‌ها یا برگ هستند و یا به تعداد درجه درخت فرزند دارند را درخت full گویند.



Full است
Complete نیست
Perfect نیست



Complete است
Full نیست
Perfect نیست

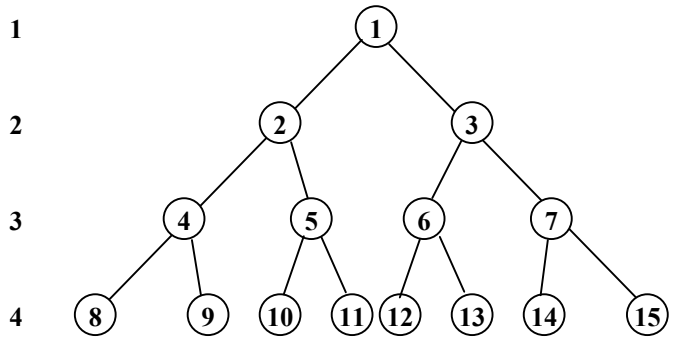


Perfect نیست
Full است
Complete است

- درختی که Perfect باشد حتماً Complete و Full هم هست.
- درختی که Complete است لزوماً Perfect نیست.
- درختی که Complete است لزوماً Full نیست.
- درختی که Full است لزوماً Complete نیست.

- حداکثر تعداد گره‌ها در یک درخت دودویی به ارتفاع d براب با $2^d - 1$ خواهد بود.
- حداثر تعداد برگها در یک درخت دودویی به ارتفاع d برابر با 2^{d-1} خواهد بود.
- حداکثر تعداد گره‌های غیر برگ یک درخت با ارتفاع d برابر با $2^{d-1} - 1$ خواهد بود.
- درختی دودویی با n گره دارای ارتفاع Log_2^{n-1} خواهد بود.

سؤال : درخت دودویی زیر را در نظر بگیرید به سؤالهای آن پاسخ دهید.



۱- حداکثر چند برگ وجود دارد؟

حداکثر تعداد برگها از رابطه 2^{d-1} بدست می‌آید. اگر بعنوان مثال ارتفاع 4 را در نظر بگیریم داریم :

حداکثر تعداد برگهای موجود در این درخت $2^{d-1}=2^{4-1}=2^3=8$

۲- حداکثر چند گره غیر از برگ داریم؟ (گره‌های داخلی)

حداکثر گره‌های داخلی از رابطه $2^{d-1}-1$ بدست می‌آید. باز هم در ارتفاع 4 داریم :

$$2^{d-1}-1=2^{4-1}-1=2^3-1=8-1=7$$

۳- حداکثر چند گره وجود دارد؟

حداکثر تعداد گره‌ها از رابطه 2^d-1 بدست آمده و بعنوان مثال باز هم در ارتفاع 4 داریم :

$$2^d-1=2^4-1=16-1=15$$

۴- چند درخت کامل متمایز به ارتفاع d داریم؟

حداکثر تعداد درخت کامل متمایز نیز از همان رابطه تعداد برگها بدست می‌آید. پس در ارتفاع 4 داریم :

$$2^{d-1}=2^{4-1}=2^3=8$$

سؤال : اگر n تا گره داشته باشیم :

الف) حداکثر عمق چقدر است؟

حداکثر عمق برابر با n خواهد بود. در این حالت درخت بصورت کاملاً اریب خواهد بود. یعنی تمام فرزندان از یک سمت (چپ یا راست) رشد می‌کنند.

ب) حداقل عمق چقدر است؟

حداقل ارتفاع یک درخت دودویی با n گره از رابطه زیر بدست می‌آید :

$$\left[\mathrm{Log}_2^n\right]+1 \Rightarrow \left[\mathrm{Log}_2^8\right]+1=3+1=4$$

نکته : همه روابط گفته شده برای درخت‌های m تایی نیز قابل تعمیم است.

اگر n_0 تعداد برگهای در یک درخت دودویی و n_2 تعداد گره‌های دو فرزندی باشند رابطه $n_0 = n_2 + 1$ برقرار است.

روشهای پیمایش درخت

۱- آرایه

برای نمایش درخت‌های دودویی می‌توان از آرایه‌ها استفاده کرد. بدین منظور به تعداد گره‌های درخت کامل متناظر با درخت مفروض برای یک آرایه حافظه نیاز داریم . در آنصورت داریم :

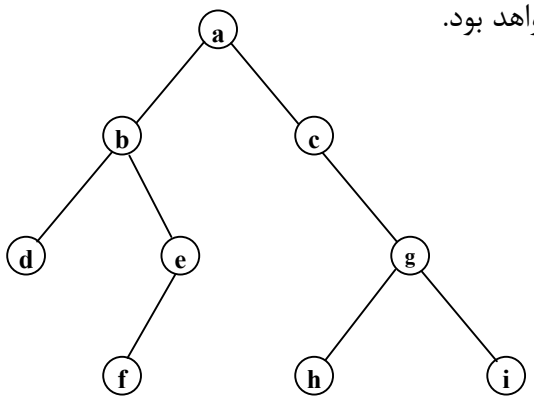
- ریشه در خانه اول آرایه قرار می‌گیرد.
- فرزند سمت چپ گره‌ای با اندیس i در آرایه درون خانه 2i قرار می‌گیرد که $2i \le n$ خواهد بود. اگر $2i > n$ بود یعنی گره i فرزند سمت چپ ندارد.

فرزند سمت چپ = 2i

- فرزند سمت راست گره‌ای با اندیس i در آرایه درون خانه 2i + 1 آرایه قرار می‌گیرد که $2i+1 \le n$ خواهد بود. اگر $2i+1 > n$ باشد یعنی گره i فرزند سمت راست ندارد.

فرزند سمت راست = 2i + 1

در نمایش درخت‌های دودویی بوسیله آرایه‌ها اگر درخت کامل نباشد اتلاف حافظه خواهیم داشت ولی اگر درخت کامل باشد روش خوبی خواهد بود.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	c	d	e		g			f				h	I

فرزند سمت چپ = 2i

فرزند سمت راست = 2i + 1

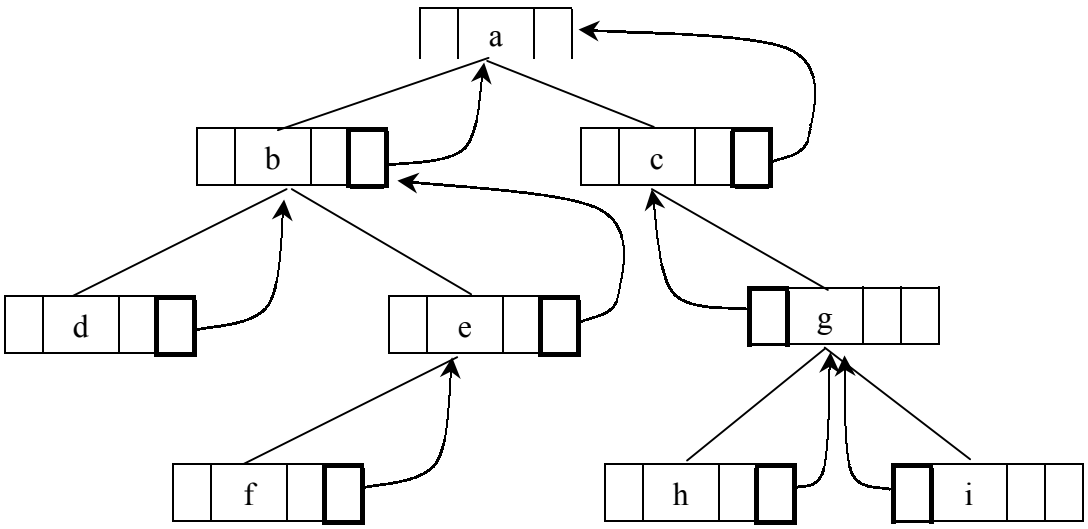
$\Rightarrow$ پدر هر گره i = $\frac{i}{2}$

۲- لیست‌های پیوندی

برای نمایش درخت‌ها به روش لیست پیوندی باید گره‌هایی با ساختار زیر تعریف کنیم.

هر گره یک فرزند سمت چپ و یک فرزند سمت راست و یک داده دارد. در ساختار تعریف شده مشخص کردن پدر هر گره به سادگی امکان پذیر نیست. برای رفع این مشکل می‌توان در ساختار هر گره یک فیلد جدید به نام Parent که به پدر آن گره اشاره می‌کند تعریف نمود.

```
Struct Treetype
{
        int data ;
        struct Treetype * Lchild ;
        struct Treetype * Rchild ;
        struct Treetype * Parent ;
}
typedef struct Treetype Tree ;
```

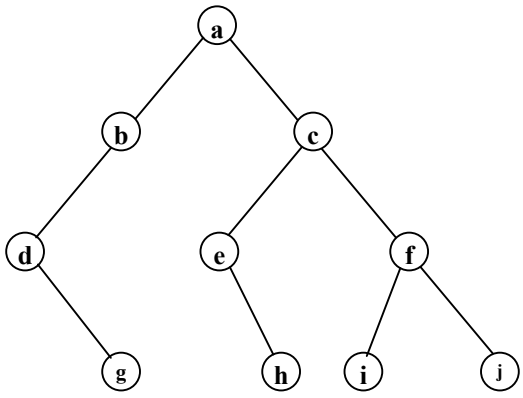


روشهای پیمایش درخت

۱- پیمایش اول عمق (عمقی)

سه روش پیمایش عمقی به شرح ذیل می‌باشد :

		سوم	دوم	اول
1.	Preorder (پیش ترتیب)	فرزند سمت راست	فرزند سمت چپ	ریشه
2.	Postorder (پس ترتیب)	ریشه	فرزند سمت راست	فرزند سمت چپ
3.	Inorder (میان ترتیب)	فرزند سمت راست	ریشه	فرزند سمت چپ



Preorder = a b d g c e h f i j

Postorder = g d b h e i j f c a

Inorder = d g b a e h c i f j

الگوریتم پیمایش عمقی به روش Inorder

```
Void  Inorder  ( Tree * T )
{
      if ( T ! = Null )
      {
            Inorder ( T → Lchild ) ;
            C out << T → data ;
            Inorder ( T → Rchild );
      }
}
```

الگوریتم پیمایش عمقی به روش Preorder

```
Void  Preorder  ( Tree * T )
{
        if ( T ! = Null )
        {
                C out << T → data ;
                Preorder  ( T → Lchild ) ;
                Preorder  ( T → Rchild );
        }
}
```

الگوریتم پیمایش عمقی به روش Postorder

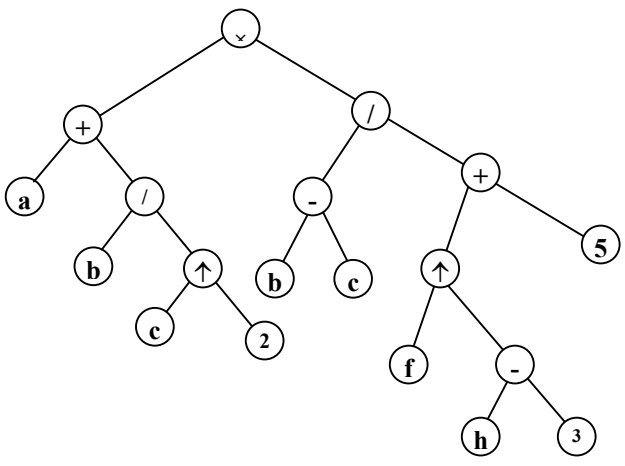
```
Void  Postorder ( Tree * T )
{
        if ( T ! = Null )
        {
                Postorder ( T → Lchild ) ;
                Postorder ( T → Rchild );
                C out << T → data ;
        }
}
```

درخت متناظر با عبارت infix

عبارت infix زیر را در نظر می‌گیریم.

$$\left(a+b/\left(c\uparrow 2\right)\times\left(b-c\right)/\left(f\uparrow\left(h-3\right)+5\right)\right)$$

هر عبارت infix یک درخت دودویی دارد.



$Inorder = a + b / c \uparrow 2 \times b - c / f \uparrow h - 3 + 5$

این همان عبارت infix بدون در نظر گرفتن پرانتزها است.

$Preorder = \times + a / b \uparrow c2 / - bc + \uparrow f - h35$

این همان عبارت prefix بدون در نظر گرفتن پرانتزها است.

$Postorder = abc2 \uparrow / + bc - fh3 - \uparrow 5 + / \times$

این همان عبارت postfix بدون در نظر گرفتن پرانتزها است.

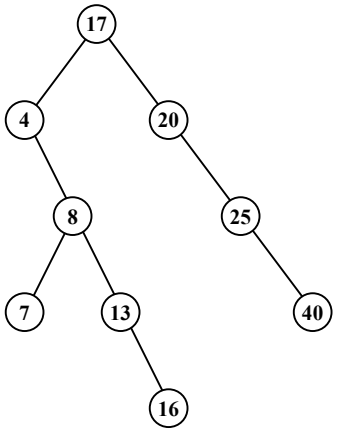
۲- پیمایش اول سطح (سطحی)

```
Void Levelorder ( Tree * T )
{
    while ( T )
    {
        C out << T → data ;
        if ( T → Lchild ) addqueue ( T → Lchild ) ;
        if ( T → Rchild ) addqueue ( T → Rchild ) ;
        T = delqueue ( ) ;
    }
}
```

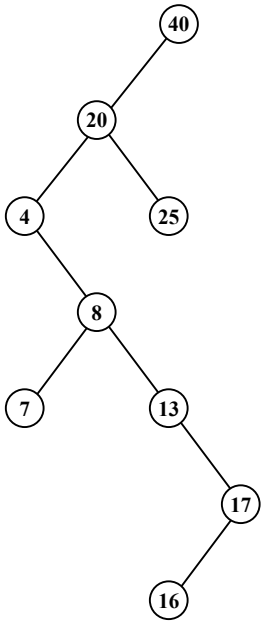
درخت جستجوی دودویی BST (Binary Search Tree)

ساختمان داده‌هایی که تا کنون بررسی شده‌اند هر یک دارای نقاط ضعفی هستند. مثلاً درج در آرایه مرتب مستلزم شیفت دادن داده‌ها و در نتیجه کندتر شدن الگوریتم است. پیمایش‌های مختلف روی لیست‌های پیوندی نیز بصورت خطی انجام می‌شود که هزینه انجام اعمال را بالا می‌برد. درخت‌های جستجوی دودویی راهکاری پیشنهاد می‌کنند که هزینه انجام اعمال اصلی مانند حذف ، اضافه و جستجو با زمان متوسط بهتری انجام می‌شود. این زمان برابر است با ارتفاع درخت که از Log_2^n تا n متغیر است. ترتیب ورود عناصر یا کلیدها برای تشکیل درخت BST از آنها کاملاً مؤثر است. کلیدهای یکسان با ترتیب متفاوت ، درخت‌های BST متفاوتی ایجاد می‌کنند.

17 20 25 40 4 8 7 13 16



40 20 4 8 13 7 17 25 16



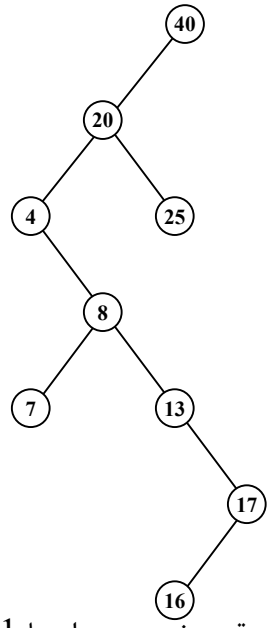
اگر درخت BST را بصورت inorder پیمایش کنیم در خروجی لیست مرتب صعودی خواهیم داشت.

هزینه ساخت یک درخت دودویی BST از یک آرایه n تایی ورودی (نامرتب) $n \times Log_2^n$ است.

الگوریتم جستجو در یک درخت دودویی BST

```
int  BSTSearch ( node * T , int * x )
{
        int founded = 0 ;
        if ( T )
        {
                if ( T → data < x )
                        founded = BSTSearch ( T → Right , x ) ;
                else if ( T → data > x )
                        founded = BSTSearch ( T → Left , x ) ;
                else founded = 1 ;
        }
        return founded ;
}
```

مثال : می‌خواهیم ببینیم عدد ۱۳ در درخت زیر وجود دارد یا خیر؟



(13) ← T	x = 13	F = ~~0~~ = 1
(8) ← T	x = 13	F = ~~0~~ = BSTSearch (13 , 13)
(4) ← T	x = 13	F = ~~0~~ = BSTSearch (8 , 13)
(20) ← T	x = 13	F = ~~0~~ = BSTSearch (4 , 13)
(40) ← T	x = 13	F = ~~0~~ = BSTSearch (20 , 13)

خروجی = founded = 1

وقتی خروجی برابر با 1 باشد یعنی داده پیدا شده و در صورتیکه 0 باشد یعنی داده پیدا نشده است.

الگوریتم اضافه کردن داده به درخت دودویی BST

```
Void  insertBST ( node * T , int x )
{
        node * p , * q , * S ;
        p = new (node) ; p → data = x ;
        p → Right = Null ; p → Left = Null ;
        S = T ;
        While ( S && S → data != x )
        {
                if ( S → data > x ) { q = S ; S = S → Left ; }
                else if ( S → data < x ) { q = S ; S = S → Right ; }
        }
        if ( !S ) if ( q → data > x ) q → Left = p ;
                else q → Right = p ;
}
```

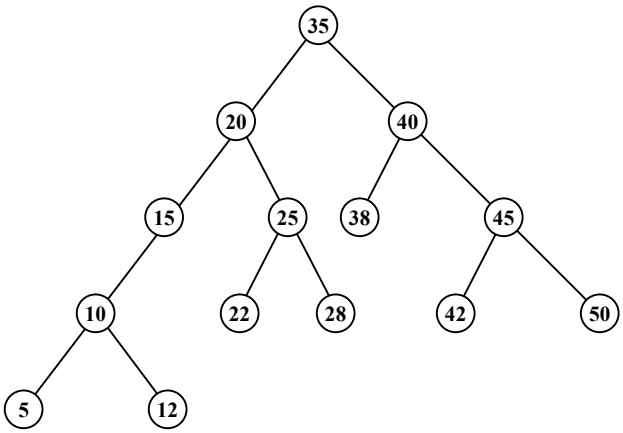
حذف

برای حذف یک گره از درخت جستجو دودویی ابتدا باید آن گره را در درخت BST پیدا کنیم. حال یکی از وضعیتهای زیر رخ می‌دهد :

۱- اگر گره مورد نظر برگ باشد حذف می‌شود یعنی حافظه گرفته شده برای گره آزاد شده و اشاره‌گر پدرش Null می‌شود.

۲- اگر گره حذف شدنی فقط یک فرزند داشته باشد فرزند آن گره جایگزین گره حذف شدنی می‌گردد و یا می‌توان مورد بعدی را انجام داد.

۳- اگر گره دارای دو فرزند باشد یک قدم به راست و سپس آنقدر به چپ می‌رویم تا به Null برسیم و یا برعکس یک قدم به چپ و سپس آنقدر به راست می‌رویم تا به Null برسیم. با دنبال کردن هر یک از حالات فوق گره آخر را جایگزین گره حذف شدنی کرده و حافظه آنرا آزاد می‌کنیم.

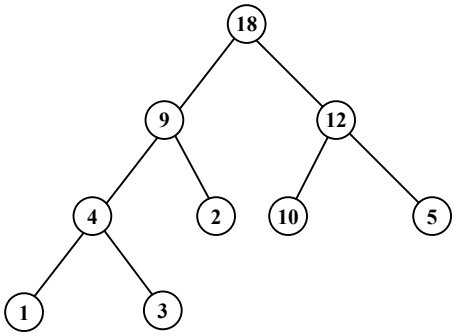


- اگر بخواهیم گره شماره ۴۰ را حذف کنیم هم می‌توانیم گره شماره ۳۸ و هم می‌توانیم گره شماره ۴۲ را جایگزین آن کنیم.
- اگر بخواهیم گره شماره ۲۰ را حذف کنیم هم می‌توانیم گره شماره ۱۵ و هم می‌توانیم گره شماره ۲۲ را جایگزین آن کنیم.
- اگر بخواهیم گره شماره ۱۵ را حذف کنیم هم می‌توانیم گره شماره ۱۰ و هم می‌توانیم گره شماره ۱۲ را جایگزین آن کنیم.

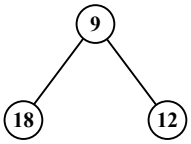
درخت heap (کپه)

درختی است دودویی کامل (Complete) که تعداد موجود در هر گره از مقدار موجود در گره‌های فرزندانش کوچکتر نباشد. این کپه ، کپه بیشترین (maxheap) است. در صورتیکه در درخت دودویی کامل مقدار هر گره از مقدار گره فرزندانش بیشتر نباشد کپه کمترین (minheap) خواهیم داشت.

maxheap



minheap



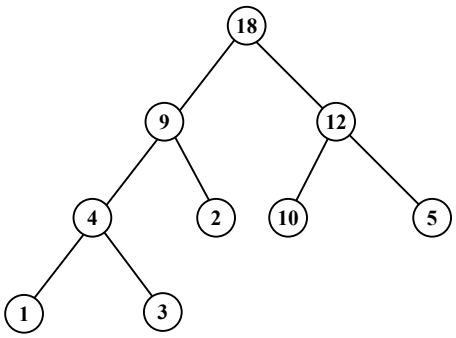
در maxheap بزرگتریـن عنصـر را مـی‌تـوان با مرتبه زمانی $O(1)$ بدست آورد (یعنی بدون محاسبه زیـرا ریشـه درخـت بزرگترین عنصر است) و متقابلاً در minheap کمترین عنصر را می‌توان با مرتبه زمانی $O(1)$ بدست آورد (در این حالت نیز کمترین عنصر همان ریشه درخت است).

افزودن داده به درخت heap

بـرای افـزودن داده جدیـد بـه درخت heap داده جدید را در انتهای لیست آرایه قرار می‌دهیم (توجه ایـنکه درخـت heap را درون آرایه نگهداری می‌کنیم). روال زیر تا رسیدن به ابتدای آرایه و یا بر قرار بودن شرط درخت heap انجام می‌شود :

- ✓ داده موجـود در خانه i (برای اولین بار آخرین عنصر آرایه) با پدر خویش در خانه $\frac{i}{2}$ مقایسه مـی‌شـود. در صـورت جابجایـی مجدداً این مقایسه برای خانه جدید ($\frac{i}{2}$) انجام می‌گردد. این روش را افزودن به طریقه درج در heap می‌نامند.
- ✓ درج در درخت heap برای هر عنصر با مرتبه زمانی Log_2^n انجام می‌شود.

درخت زیر را در نظر بگیرید :



آرایه این درخت به شکل زیر خواهد شد.

1	2	3	4	5	6	7	8	9
18	9	12	4	2	10	5	1	3

حال می‌خواهیم داده شماره 8 را به این آرایه اضافه کنیم و درخت heap نیز برقرار باشد.

1	2	3	4	5	6	7	8	9	10
18	9	12	4	~~2~~	10	5	1	3	~~8~~
18	9	12	4	8	10	5	1	3	2

حال می‌خواهیم ابتدا داده شماره 7 و سپس داده شماره 25 را به آرایه اضافه کنیم.

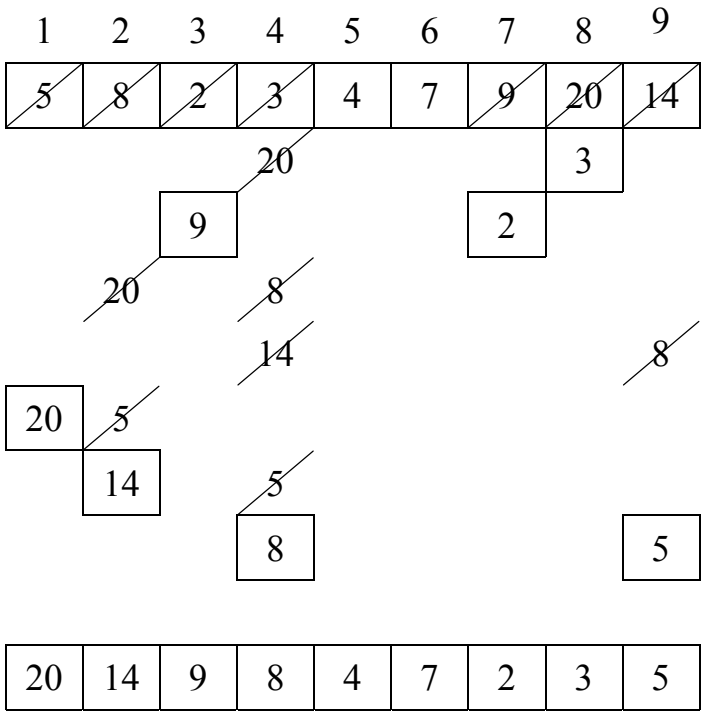
1	2	3	4	5	6	7	8	9	10	11	12
18	9	12	4	8	~~10~~	5	1	3	2	7	~~25~~
18	9	~~12~~	4	8	~~25~~	5	1	3	2	7	10
~~18~~	9	~~25~~	4	8	12	5	1	3	2	7	10
25	9	18	4	8	12	5	1	3	2	7	10

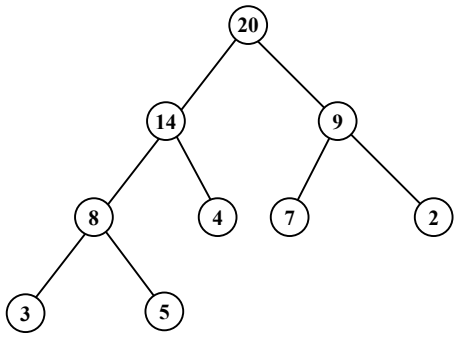
چون داده شماره 7 بر سر جای خود درست قرار گرفته است پس آنرا دست نمی‌زنیم و فقط داده شماره 25 را آنقدر جابجا کرده تا درخت heap برقرار باشد.

روش ساخت درخت دودویی heap به روش جوان‌ترین پدر

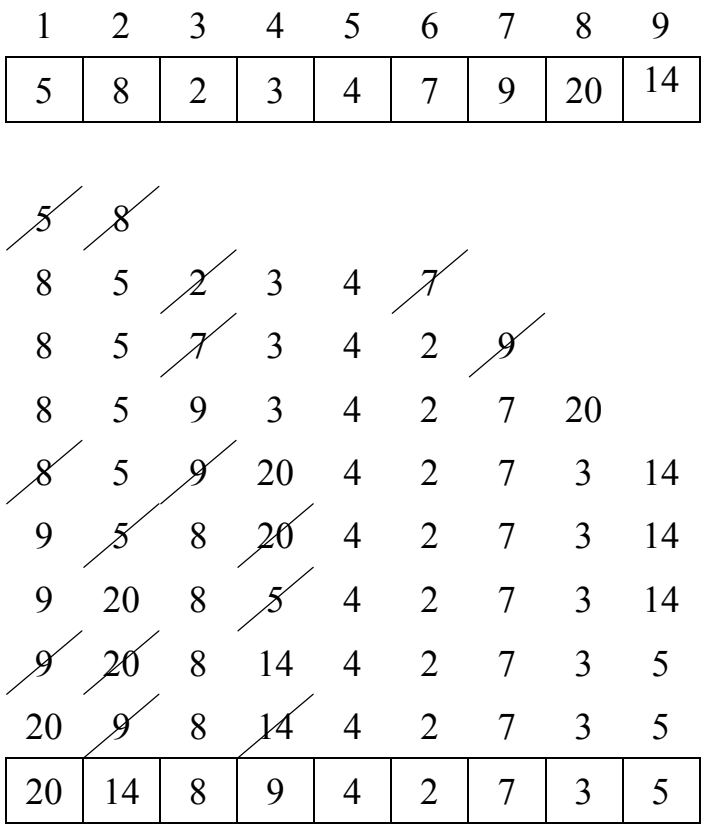
در ایجاد کپه به روش جوان‌ترین پدر ابتدا همه n عنصر ورودی را در یک آرایه قرار دهید. سپس از پائین درخت شروع کرده ، هر پدر و فرزندانش را بصورت کپه تنظیم می‌کنیم و به سمت بالا (ریشه) حرکت می‌کنیم. همینطور که به سمت ریشه می‌رویم زیردرخت‌ها بصورت کپه درآمده‌اند. در این روش چون برگها خودبخود به تنهایی یک heap هستند باید از جوان‌ترین پدر شروع کنیم که اگر عناصر آرایه i تا باشند از عنصر $\frac{i}{2}$ شروع می‌کنیم.

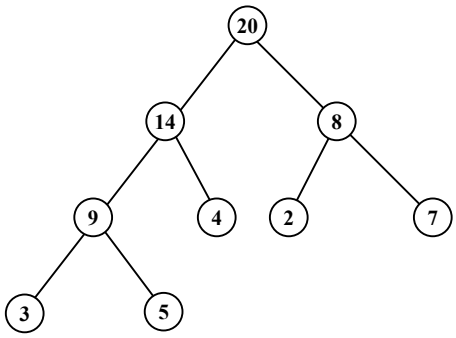
مثال : درخت زیر را به روش جوان‌ترین درخت به صورت درخت دودویی heap درآورید.





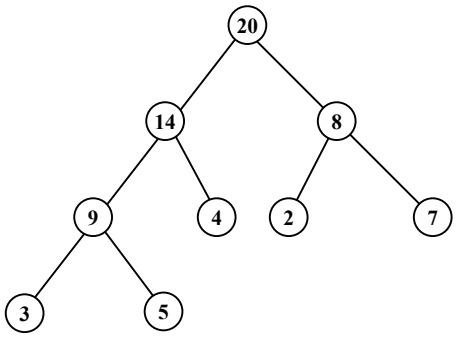
مثال : آرایه زیر را به روش درج بصورت درخت heap بنویسید.





حذف یک عنصر

برای حذف یک عنصر از درخت heap ، ریشه درخت خارج شده و داده آخر به جای آن قرار می‌گیرد. سپس از ابتدای آرایه (i =) شروع می‌کنیم و بین 2i و 2i + 1 عنصر ماکزیمم را در صورت لزوم با عنصر i عوض می‌کنیم. به همین ترتیب جابجایی انجام گرفته تا زمانیکه به انتهای آرایه برسیم (یک گره فرزندی نداشته باشد).



1	2	3	4	5	6	7	8	9
~~20~~	14	9	8	4	7	2	3	~~5~~
~~5~~	~~14~~	9	8	4	7	2	3	
14	~~5~~	9	~~8~~	4	7	2	3	
14	8	9	5	4	7	2	3	
14	8	9	5	4	7	2	3	

در این مثال عنصر 20 را حذف کرده و عنصر 5 را جایگزین کرده و بقیه مراحل ساخت درخت heap را انجام داده‌ایم.

سؤال : تابع f چه کاری انجام می‌دهد؟

```
int f (node * T )
{
        int L , r ;
        if ( T )
        {
                L = f ( T → Left ) ;
                R = f ( T → Right ) ;
                if  L > r  return  L + 1 ;
                else return  r + 1 ;
        }
        return  0 ;
}
```

جواب : ارتفاع درخت را نشان می‌دهد.

سؤال : تابع f چه کاری انجام می‌دهد؟

```
node  * f ( node * T )
{
        node  * r , * s , * q ;
        q = Null ;
        if ( T )
        {
                r = f ( T → Left ) ;
                s = f ( T → Right ) ;
                q = New ( node ) ;
                q → Left = r ;
                q → Right = s ;
                q → data = T → data ;
        }
        return  q ;
}
```

جواب : از یک کپی تهیه می‌کند.

سؤال : تابع g چه کاری انجام می‌دهد؟

```
int  g ( node * T )
{
        if ( T )
        {
                if ( ! T → Left ) && ( ! T → Right )
                        return  1 ;
                else
                        return ( g ( T → Left ) + g ( T → Righe ) + 1 ) ;
        }
        return  0 ;
}
```

جواب : تعداد گره‌ها را چاپ می‌کند.

سؤال : تابع h چه کاری انجام می‌دهد؟

```
int  h ( node * T )
{
        if ( T )
        {
                if ( T → Left = = Null ) && ( T → Right = = Null )
                        return  1 ;
                else
                        return ( h ( T → Left ) + h ( T → Righe ) ) ;
        }
        return  0 ;
}
```

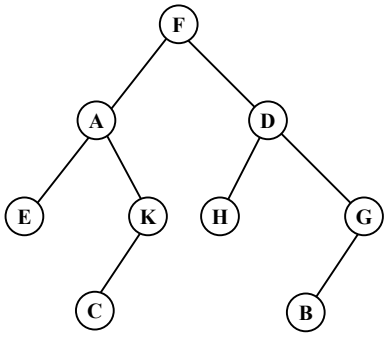
جواب : تعداد برگها را چاپ می‌کند.

سـؤال : پـیمایش Inorder و Preorder زیـر را داریـم. درخت دودویی آنرا تشکیل داده و پیمایش Postorder آنرا بنویسید.

Inorder : E A C K F H D B G

Preorder : F A E K C D H G B

Postorder : E C K A H B G D F

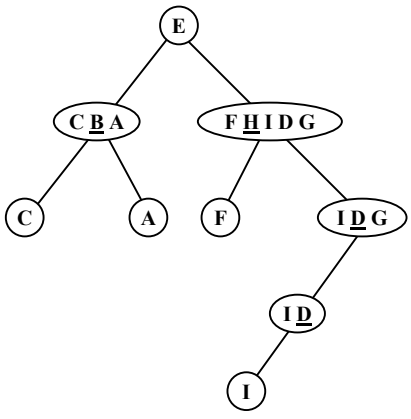


سـؤال : پـیمایش Inorder و Postorder زیر را داریم. درخت دودویی آنرا تشکیل داده و پیمایش Preorder آنرا بنویسید.

Inorder : C B A E F H I D G

Postorder : C A B F I D G H E

Preorder : E B C A H F G D I



سؤال : درخت maxheap آرایه زیر را از دو روش درج و جوان‌ترین پدر بدست آورید.

1	2	3	4	5	6	7	8
~~10~~	15	~~22~~	~~4~~	11	~~23~~	~~19~~	~~14~~

به روش جوان‌ترین پدر

14 4
~~23~~ ~~22~~ ~~19~~
23 15 ~~10~~
22 10 19

23	15	22	14	11	10	19	4

1	2	3	4	5	6	7	8
10	15	22	4	11	23	19	14

به روش درج

~~10~~ ~~15~~
~~15~~ 10 ~~22~~
22 ~~10~~ 15 4 ~~11~~
22 11 ~~15~~ 4 10 ~~23~~
22 11 23 ~~4~~ 10 15 19 ~~14~~
~~22~~ 11 ~~23~~ 14 10 15 19 4
23 ~~11~~ 22 ~~14~~ 10 15 19 4
23 14 22 11 10 15 19 4

23	14	22	11	10	15	19	4