فصل پنجم: درخت(Tree)

بخش ها

- آشنایی با درخت
- درخت های دودویی
- پیمایش درختان
- هرم
- جنگل

مفهوم درخت

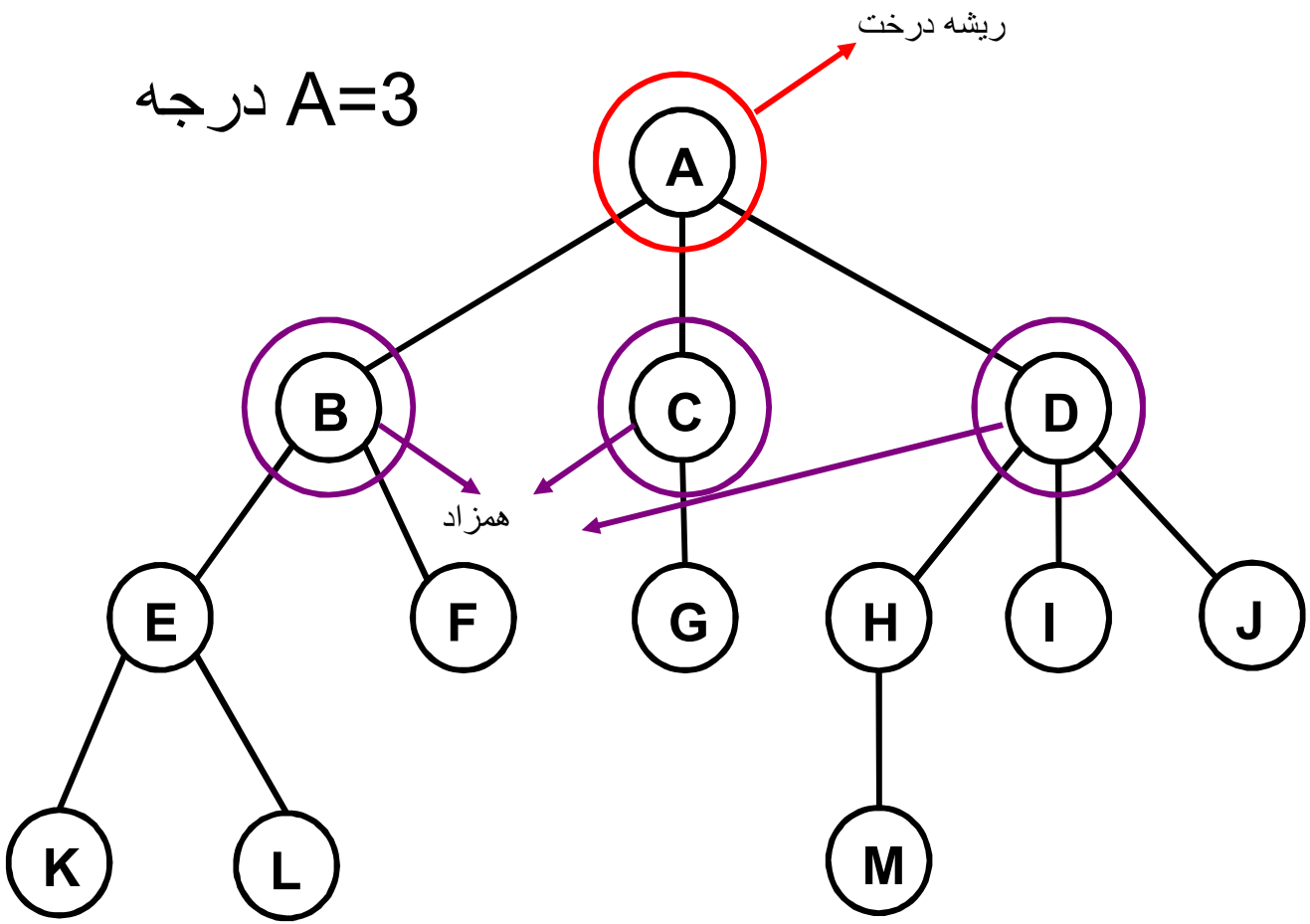
مجموعه ای مجزا و محدود از یک یا چند گره با شرایط زیر می باشد:

- دارای گره خاصی به نام ریشه باشد.
- بقیه عناصر درخت به ریشه متصل هستند.
- بقیه گره ها به $n \geq 0$ مجموعه مجزا $T_1,....T_n$ تقسیم شده که هر یک از این مجموعه ها خود یک درخت هستند. $T_1,....T_n$ زیردرختان ریشه نامیده می شوند.
- نکته: عبارت «مجموعه مجزا» از هرگونه اتصال زیردرخت ها به هم جلوگیری میکند یعنی در درخت حلقه نداریم

تعاریف

- به عنصر حاوی اطلاعات و انشعابات به دیگر عناصر، **گره** اطلاق می شود.
- تعداد زیر درختهای یک گره، **درجه** آن نامیده می شود.
- حداکثر درجه گره های یک درخت را **درجه درخت** می نامند.
- گره ای که درجه آن صفر باشد، **برگ** یا **گره پایانی** نامیده می شود و در مقابل بقیه گره‌ها **گره های غیرپایانی** نامیده می شوند.
- گره ریشه را در هر زیر درخت **گره پدر(والد)** و به گره‌های متصل شده به آن **گره فرزند** می گویند.
- فرزندانی که پدر یکسان دارند **برادر (یا همزاد)** نامیده می شوند.

مثالی از یک درخت



A والد B، C، D است . B، C، D همزادند.

اصطلاحات درخت

- **اجداد گره:** اجداد یک گره ، گره هایی هستند که در مسیر طی شده از ریشه تا آن گره وجود دارند.
- **سطح گره:** سطح یک گره بدین صورت تعریف می شود که ریشه در سطح یک قرار می گیرد. برای تمامی گره های بعدی ، سطح گره برابر است با سطح والد به اضافه یک .
- **ارتفاع درخت:** ارتفاع یا عمق یک درخت به بیشترین سطح گره های آن درخت گفته می شود.
- **یال و مسیر** : خطی که از یک گره به گره بعدی رسم می شود **یال** و دنباله ای از یال های متوالی **مسیر** نامیده می شود.
- **شاخه:** مسیری که به یک برگ ختم می شود **شاخه** نام دارد.

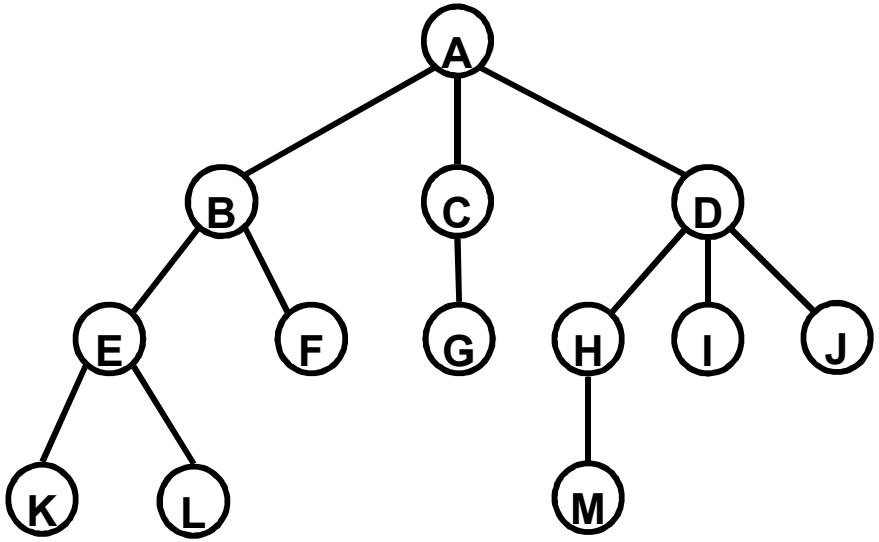
اصطلاحات درخت

- **درخت مرتب:** درختی که ترتیب زیردرخت ها در آن مهم باشد.
- **درخت های مشابه:** درخت های T1 و T2 مشابه هستند اگر دارای ساختمان یکسان باشند. به بیان دیگر شکل مساوی داشته باشند.
- **درخت Kتایی:** درختی که تعداد فرزندان هر گره در آن حداکثر K باشد.
- **درخت متوازن:** درختی که اختلاف سطح برگ های آن حداکثر یک باشد.
- **درخت کاملا متوازن:** درختی که اختلاف سطح برگ های آن صفر باشد.

نمایش درخت با لیست

- یک راه نمایش درخت ، استفاده از لیست است .
- لیست زیر معادل درخت نمایش داده شده است:

(A(B(E(K ،L)،F)،C(G)،D(H(M)،I،J)))



درخت دودویی(Binary Tree)

یک درخت دودویی یا تهی است یا حاوی مجموعه ای محدود از گره ها شامل یک ریشه و دو زیر درخت دودویی است. این درخت ها زیر درخت های چپ و راست نامیده می شوند.

- مشخصه اصلی یک درخت دودویی این است که هر گره آن حداکثر دو انشعاب دارد یعنی گره ای با درجه بیشتر از دو وجود ندارد.
- در درخت های دودویی، زیر درخت سمت چپ و راست با یکدیگر متفاوت است.

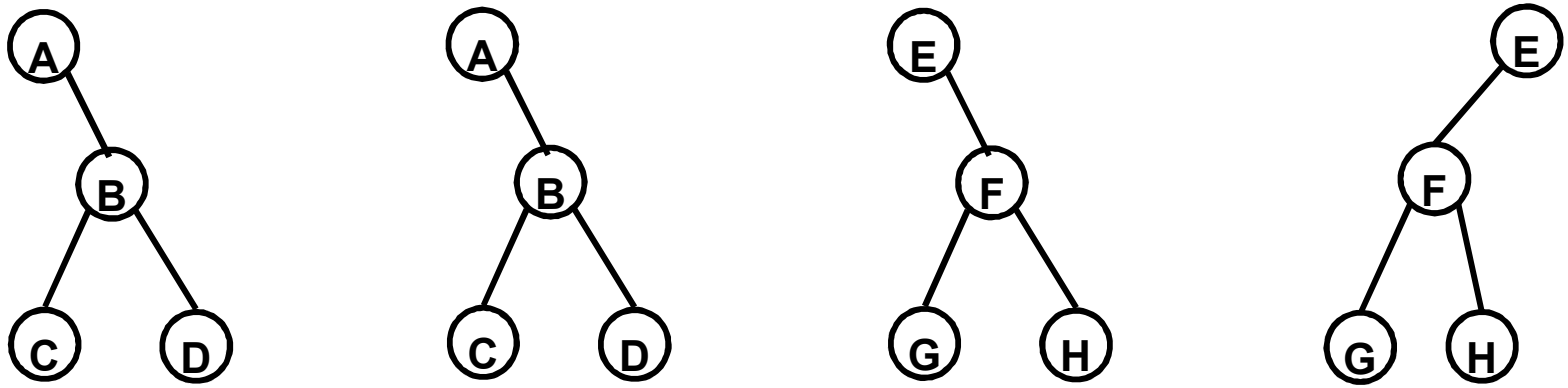
درخت دودویی

تفاوت درخت عادی با درخت دودویی:

- درخت عادی نمی تواند تهی باشد، اما درخت دودویی تهی وجود دارد.
- در درخت دودویی ترتیب فرزندان دارای اهمیت بوده در حالی که در درخت عادی این طور نیست.

درخت دودویی

نمونه هایی از درخت دودویی:



(الف) **(ب)** **(پ)** **(ت)**

سه درخت ب و پ و ت، مشابه هستند.

درخت های پ و ت کپی هستند.

درخت های ت و الف نه کپی هستند و نه مشابه

انواع درخت دودویی

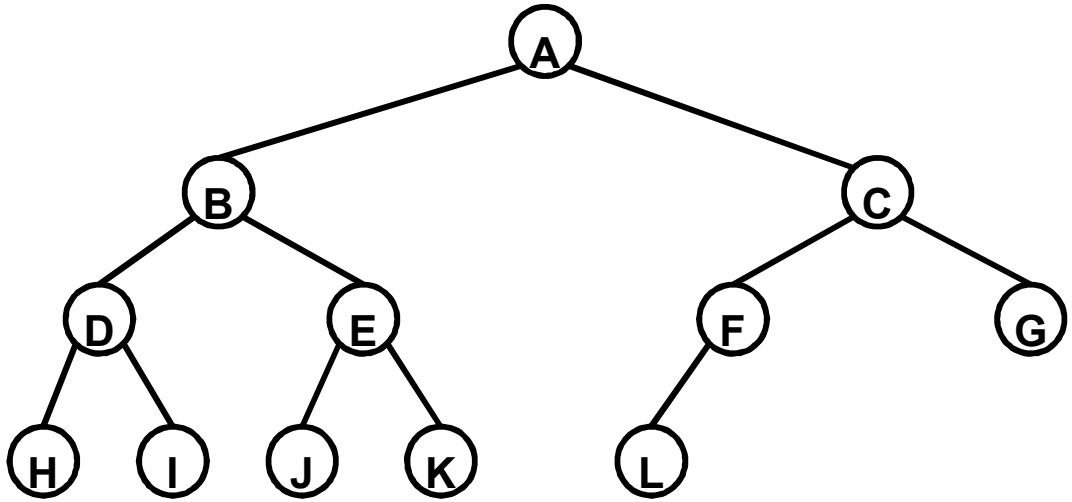
درخت مورب:

- درخت مورب چپ: هر گره، فرزند چپ والد خود باشد.
- درخت مورب راست: هر گره، فرزند راست والد خود باشد.

انواع درخت دودویی

درخت کامل:

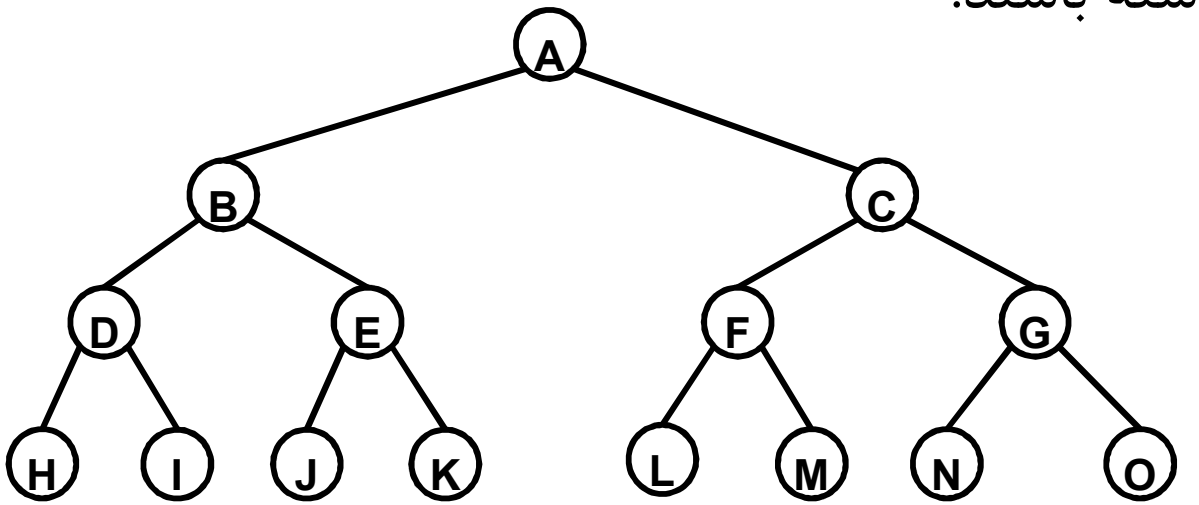
درخت T را کامل گویند اگر تمام سطح های آن به جز احتمالا آخرین سطح حداکثر تعداد گره ممکن را داشته باشند. همچنین تمام گره های آخرین سطح در سمت چپ ترین مکان باشند.



انواع درخت دودویی

درخت پر:

درختی دودویی که همه گره ها به جز گره های سطح آخر دقیقا دو فرزند داشته باشند.



نکته: درخت پر هم کامل است و هم کاملا متوازن.

خواص درخت دودویی

حداکثر تعداد گره ها:

- حداکثر تعداد گره ها در سطح iام یک درخت دودویی 2^{i-1} است.
- حداکثر تعداد گره ها در یک درخت دودویی به عمق k، $2^{k}-1$ است.

مثال:

- حداکثر تعداد گره ها در سطح چهارم یک درخت دودویی ۸ است.
- حداکثر تعداد گره ها در یک درخت دودویی به عمق ۴، ۱۵ است

خواص درخت دودویی

رابطه بین تعداد گره های برگ و گره های درجه دو:

- برای هر درخت دودویی غیر تهی مانند T ، اگر n_0 تعداد گره های پایانی و n_2 تعداد گره های درجه ۲ باشد ، آنگاه خواهیم داشت :

$$n_0 = n_2 + 1$$

خواص درخت دودویی

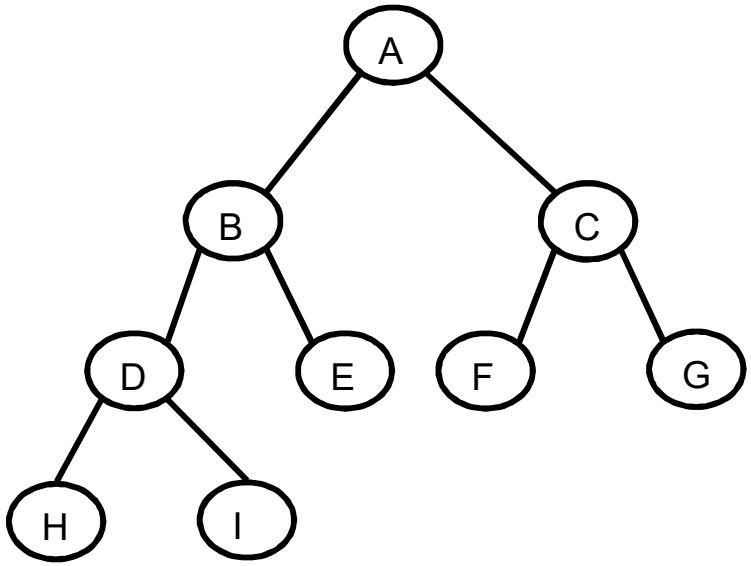
- یک درخت دودویی پر به عمق k، یک درخت دودویی پر است مشروط به اینکه 2k-1 گره داشته باشد.
- یک درخت دودویی با n گره و عمق k کامل است ، اگر و تنها اگر گره هایش مطابق با گره های شماره گذاری شده در یک درخت دودویی پر به عمق k باشد.
- ارتفاع یک درخت دودویی کامل با n گره برابر $\lceil \log_2(n+1) \rceil$ است.

نمایش درخت دودویی

نمایش درخت دودویی به دو صورت است :

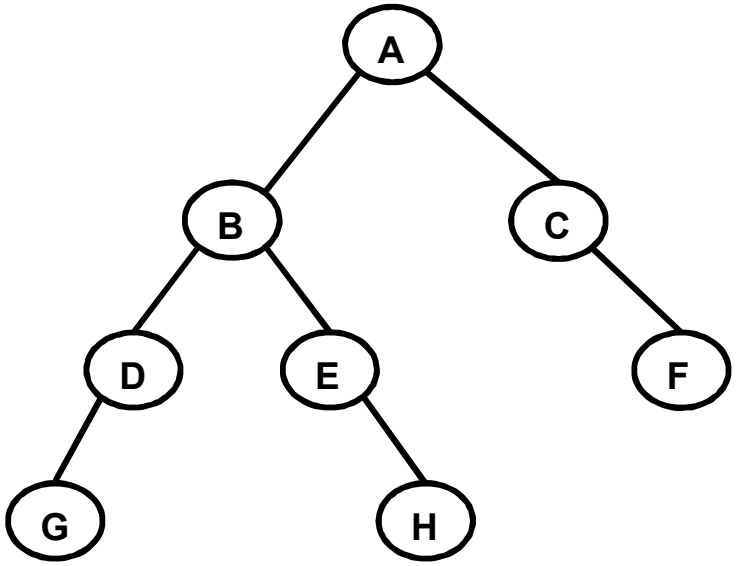
- **آرایه**
- **لیست پیوندی**

نمایش درخت دودویی با آرایه



0	-
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I

نمایش درخت دودویی با آرایه



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E		F	G			H				

نمایش درخت دودویی با آرایه

اگر یک درخت دودویی کامل با n گره (یعنی $=[\log_2 n]+1$ عمق) به ترتیب نمایش داده شده تعریف شده باشد ، آنگاه برای هر گره با اندیس i و $1 \le i \le n$، داریم:

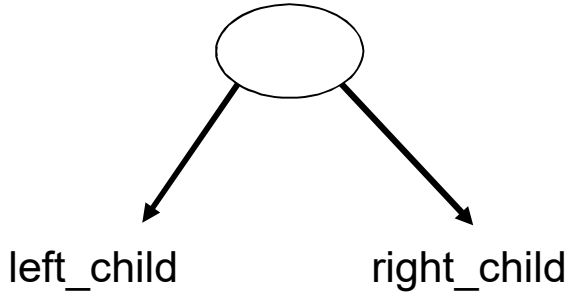
- اگر i≠1 ، آنگاه پدر i در [i/2] است. اگر i=1 ، i ریشه است و پدری نخواهد داشت.
- اگر2i≤n، آنگاه فرزند چپ i در 2i است. اگر 2i>n ، آنگاه i فرزند چپ ندارد.
- اگر 2i+1≤n ، آنگاه فرزند راست i در 2i+1 است. اگر 2i+1>n ، آنگاه i فرزند راست ندارد

نکته: در بدترین حالت ، یک درخت مورب به عمق k، به 2^k-1 خانه آرایه نیاز دارد که از این مقدار، فقط k خانه اشغال می شود.

نمایش درخت دودویی با آرایه

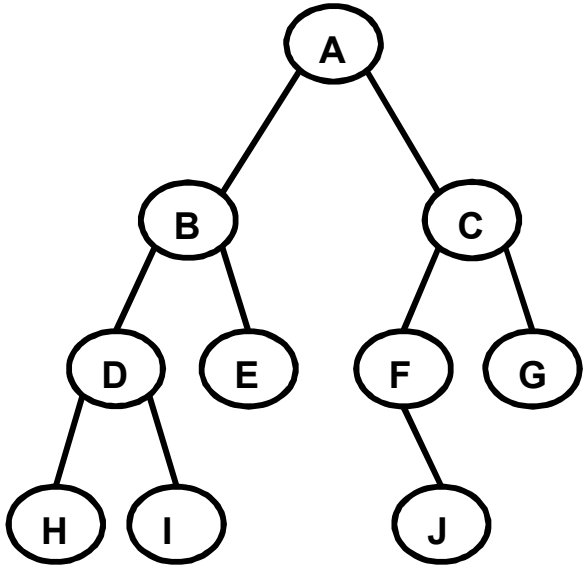
- اگرچه نمایش ترتیبی (آرایه ای) برای درختان دودویی کامل مناسب به نظر می رسد ، اما برای بسیاری از درختان دیگر باعث اتلاف حافظه می شود به علاوه ، این روش از نارسایی های موجود در نمایش ترتیبی نیز برخوردار است.
- درج یا حذف گره های یک درخت ، مستلزم جابه جایی گره هاست که خود باعث تغییر شماره سطح گره ها می شود.

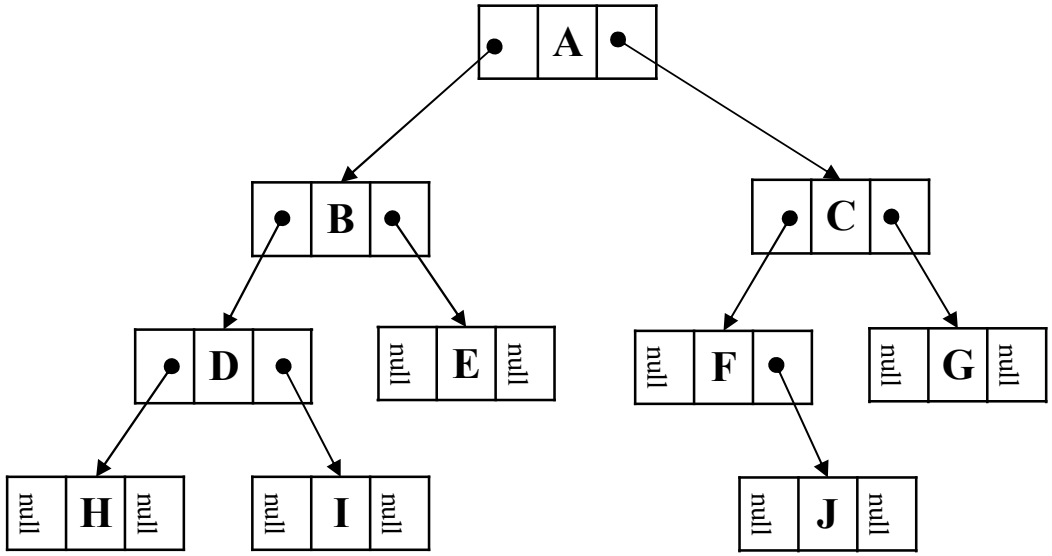
نمایش درخت دودویی با لیست پیوندی



left_child	data	right_child

نمایش درخت دودویی با لیست پیوندی





پیمایش درخت دودویی

- به هنگام پیمایش یک درخت دودویی ، با هر گره و زیردرختان آن به طرز مشابهی رفتار کنیم.

اگر L حرکت به چپ، V ملاقات کردن یک گره (برای مثال ، چاپ فیلد داده آن گره) و R حرکت به راست باشد، آنگاه شش ترکیب ممکن برای پیمایش یک درخت خواهیم داشت :

LVR ،LRV ،VLR ،VRL ،RVL ،RLV

پیمایش درخت دودویی

- اگر تنها حالت هایی را انتخاب کنیم که ابتدا به سمت چپ و بعد به سمت راست برویم ، تنها سه ترکیب VLR، LRV، LVRخواهیم داشت.
- این سه حالت را با توجه به موقعیت ریشه (V) نسبت به پیمایش گره سمت چپ (L) و گره سمت راست(R) به ترتیب Preorder، Postorder، Inorder می نامیم.

پیمایش درخت دودویی

- Postorder: در این پیمایش، یک گره موقعی ملاقات و چاپ می شود که زیردرخت چپ و راست آن قبلا ملاقات شده باشند.(حاصل معادل Postfix)
- Preorder: در این پیمایش ، یک گره قبل از پیمایش زیردرخت چپ و راست ، ملاقات می گردد. (حاصل معادل Prefix)
- Inorder: در این پیمایش، ابتدا زیردرخت چپ، سپس گره و بعد از آن زیردرخت راست ملاقات می گردند.(حاصل معادل Infix)

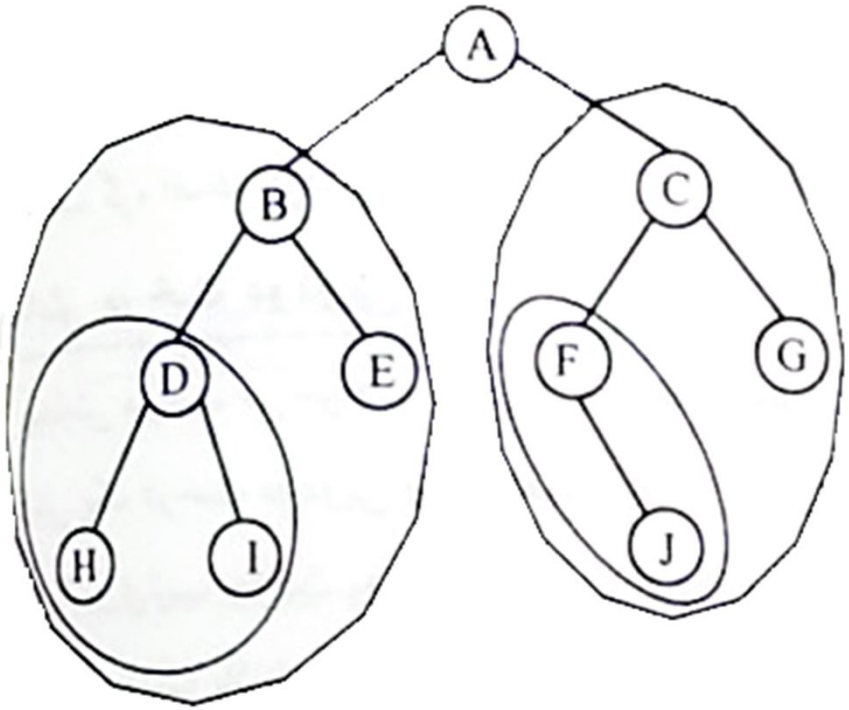
پیمایش Inorder

- در این پیمایش، حرکت به پایین از سمت چپ انجام می شود و تا آخرین گره صورت می گیرد، سپس گره والد را بازیابی کرده و بعد به سمت راست رفته و به همین ترتیب ادامه می دهیم.

```
void  inorder (tree_pointer  ptr )
{
    if(ptr){
        inorder( ptr -> left_child );
        printf(" % d"  ،ptr -> data );
        inorder( ptr -> right_child );
    }
}
```

پیمایش Inorder

- مثال:



H D I B E A F J C G

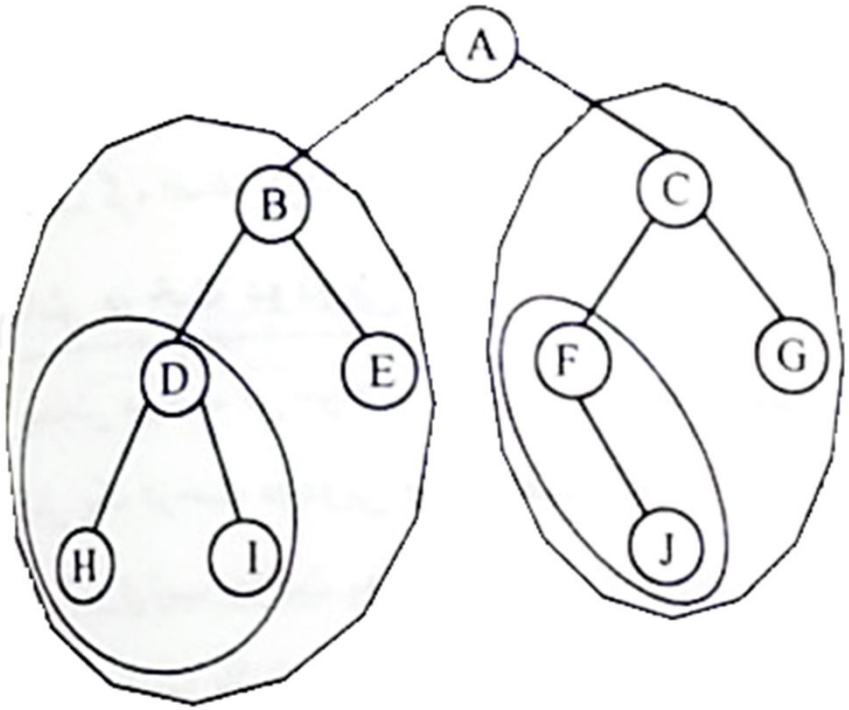
پیمایش Preorder

- در این پیمایش، ابتدا گره را بازیابی و ملاقات نموده و سپس زیردرخت چپ را دنبال و تمام گره ها را بازیابی می کنیم. این فرآیند ادامه پیدا می کند تا به یک گره تهی برسیم. در این نقطه ، به نزدیکترین جدی که دارای یک فرزند راست باشد مراجعه و با این گره شروع خواهیم نمود.

```
void  preorder(tree_pointer ptr )
{
    if(ptr){
        printf(" % d"  ،ptr -> data );
        preorder( ptr -> left_child );
        preorder( ptr -> right_child );
    }
}
```

پیمایش Preorder

- مثال:



A B D H I E C F J G

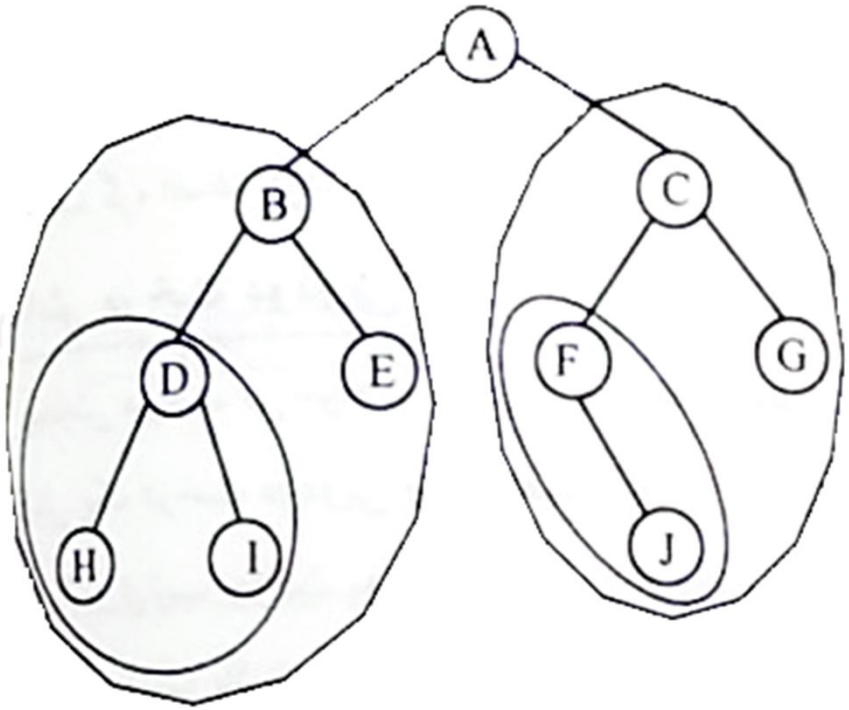
پیمایش Postorder

- این پیمایش دو فرزند یک گره را قبل از بازیابی آن گره ملاقات و چاپ می کند. این مساله بدین مفهوم است که فرزندان یک گره قبل از خود آن گره بازیابی می گردد.

```
void  postorder(tree_pointer ptr )
{
    if(ptr){
        postorder( ptr -> left_child );
        postorder( ptr -> right_child );
        printf(" % d"  ،ptr -> data );
    }
}
```

پیمایش Postorder

- مثال:

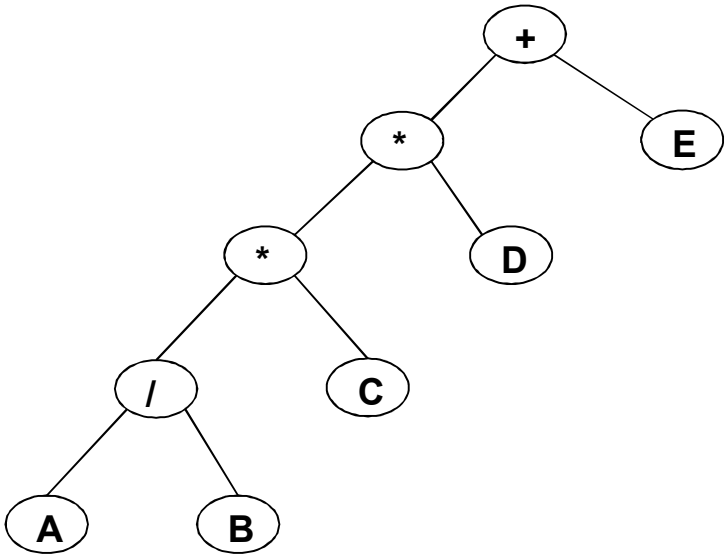


H I D E B J F G C A

نمایش عبارات ریاضی با درخت دودویی

- درخت زیر حاوی یک عبارت ریاضی است :

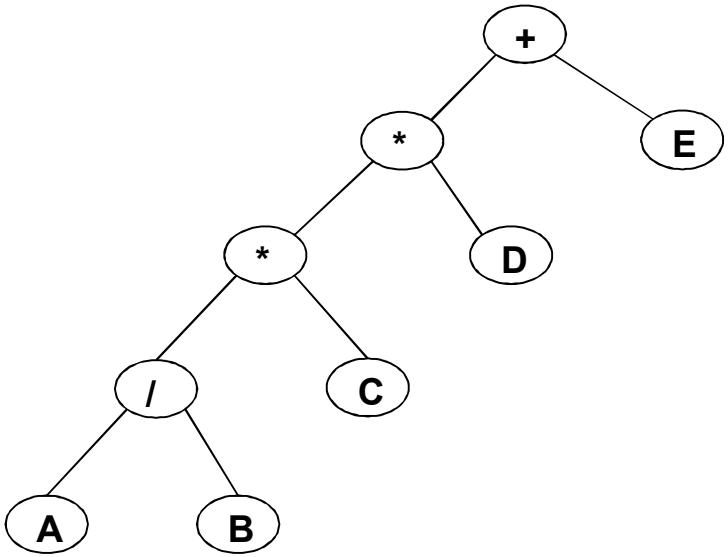
A/B*C*D*+E



پیمایش عبارات ریاضی با درخت دودویی

- پیمایش Preorder

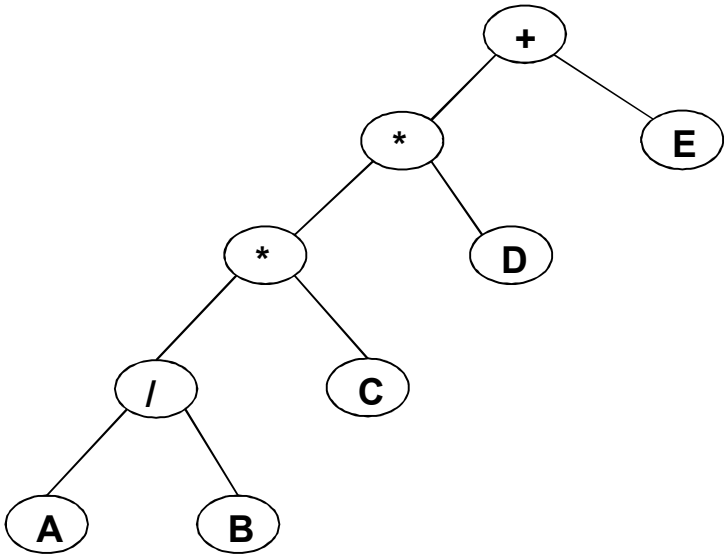
+ * * / A B C D E



پیمایش عبارات ریاضی با درخت دودویی

- پیمایش Postorder

A B / C * D * E +



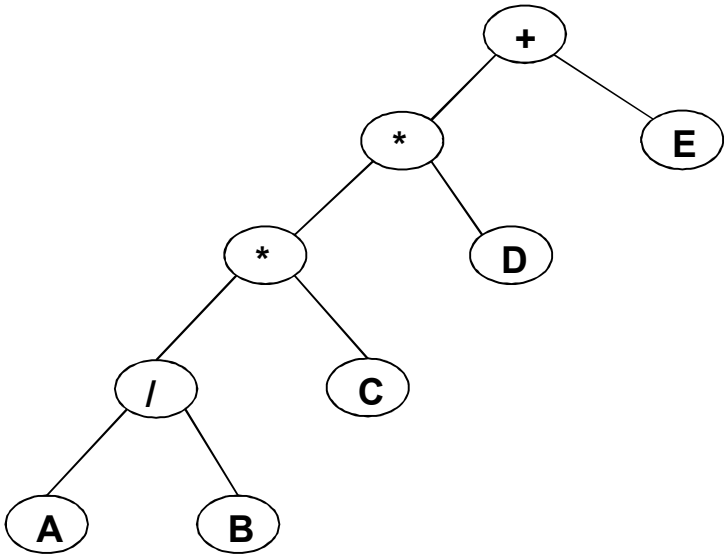
پیمایش ترتیب سطحی

- پیمایش های inorder، preorder، postorder چه به صورت بازگشت پذیری نوشته یا به صورت غیربازگشتی ، همگی نیازمند پشته می باشند.
- در پیمایش ترتیب سطحی ، ابتدا ریشه بازیابی ، سپس فرزند چپ ریشه و به دنبال آن فرزند راست ریشه بازیابی می گردد. این شیوه با بازیابی از گره منتهی الیه سمت چپ به سمت راست هر سطح جدید تکرار می گردد. این پیمایش از صف استفاده می کند.

پیمایش عبارات ریاضی با درخت دودویی

- پیمایش ترتیب سطحی

+ * E * D / C A B



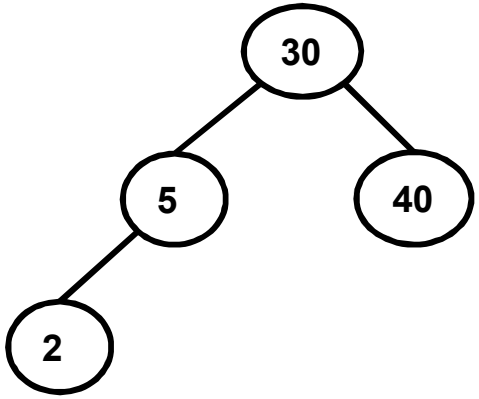
درخت جستجوی دودویی(BST)

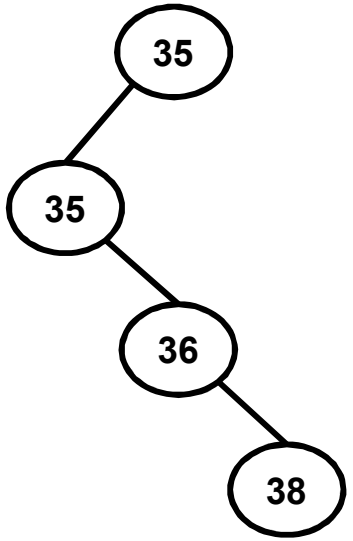
یک درخت جستجوی یک درخت دودویی است که ممکن است تهی باشد. اگر درخت تهی نباشد خصوصیات زیر را برآورده می کند :

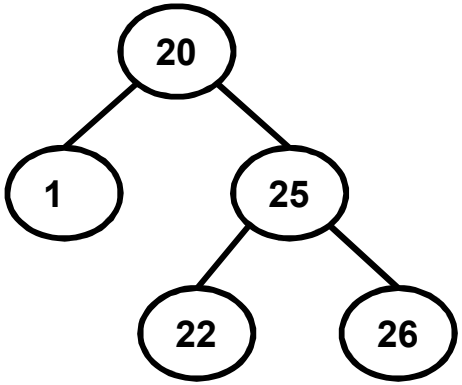
- هر عنصر دارای یک کلید است و دو عنصر نباید دارای کلید یکسان باشند ، در واقع کلیدها منحصر به فرد هستند.
- کلیدهای واقع در زیردرخت غیرتهی چپ باید کمتر از مقدار کلید واقع در ریشه زیردرخت راست باشد.
- کلیدهای واقع در زیردرخت غیرتهی راست باید بزرگتر از مقدار کلید واقع در ریشه زیردرخت چپ باشد.
- زیردرختان چپ و راست نیز خود درختان جستجوی دودویی میباشند.

درخت جستجوی دودویی

مثال:

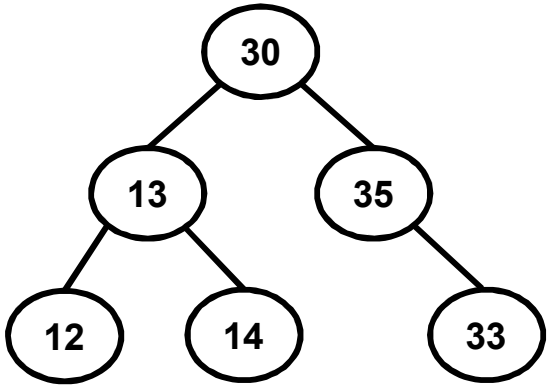






درخت جستجوی دودویی

مثال: آیا درخت زیر، یک BST است؟



یک درخت جستجوی دودوئی نیست زیرا زیردرخت راست گره ۳۵ از آن کوچکتر است

جستجو در درخت جستجوی دودویی

برای جستجوی عنصری با کلید key، ابتدا از ریشه شروع می کنیم ، اگر ریشه تهی باشد ، درخت جستجو فاقد هر عنصری بوده و جستجو ناموفق خواهد بود. در غیر این صورت key را با با مقدار کلید ریشه مقایسه کرده:

- اگر key کمتر از مقدار کلید ریشه باشد ، هیچ عنصری در زیردرخت راست وجود ندارد که دارای کلیدی برابر key باشد ، بنابراین زیردرخت چپ ریشه را جستجو می کنیم.
- اگر key بزرگتر از مقدار کلید ریشه باشد ، زیردرخت راست را جستجو می کنیم.

نکته: اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد ، عمل جستجو را در مدت O(h) انجام می شود.

درج عنصر در درخت جستجوی دودویی

برای جستجوی عنصری با کلید key، ابتدا از ریشه شروع می کنیم ، اگر ریشه تهی باشد ، درخت جستجو فاقد هر عنصری بوده و جستجو ناموفق خواهد بود. در غیر این صورت key را با با مقدار کلید ریشه مقایسه کرده:

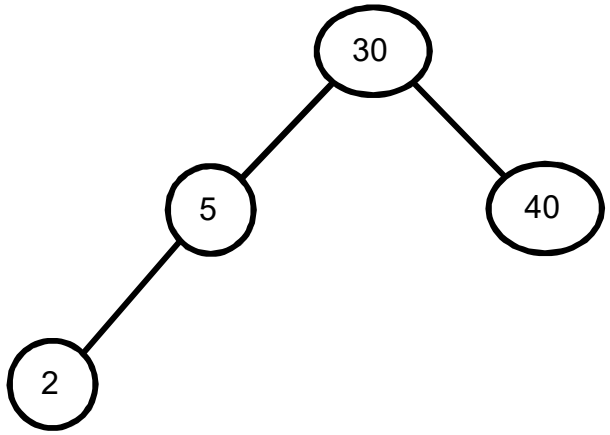
- اگر key کمتر از مقدار کلید ریشه باشد ، هیچ عنصری در زیردرخت راست وجود ندارد که دارای کلیدی برابر key باشد ، بنابراین زیردرخت چپ ریشه را جستجو می کنیم.
- اگر key بزرگتر از مقدار کلید ریشه باشد ، زیردرخت راست را جستجو می کنیم.

نکته: اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد ، عمل جستجو را در مدت O(h) انجام می شود.

درج عنصر در درخت جستجوی دودویی

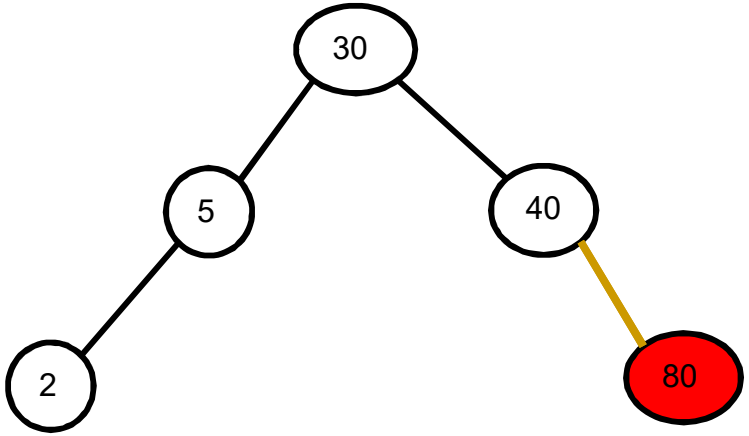
برای درج عنصر جدیدی به نام key، ابتدا باید مشخص نمود که آیا کلید با عناصر موجود متفاوت است یا خیر. برای انجام این کار باید درخت را جستجو کرد، اگرجستجو ناموفق باشد ، عنصر را در محلی که جستجو خاتمه پیدا نموده است ، درج می کنیم.

مثال درج: با فرض درخت روبرو

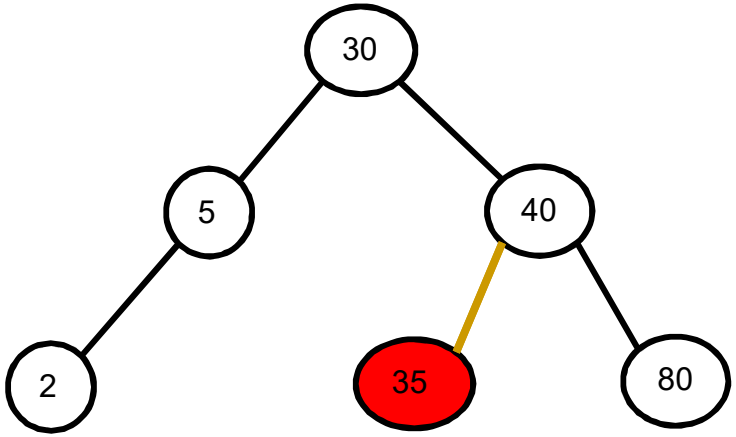


درج عنصر در درخت جستجوی دودویی

مثال درج: درج عدد ۸۰



مثال درج: درج عدد ۳۵

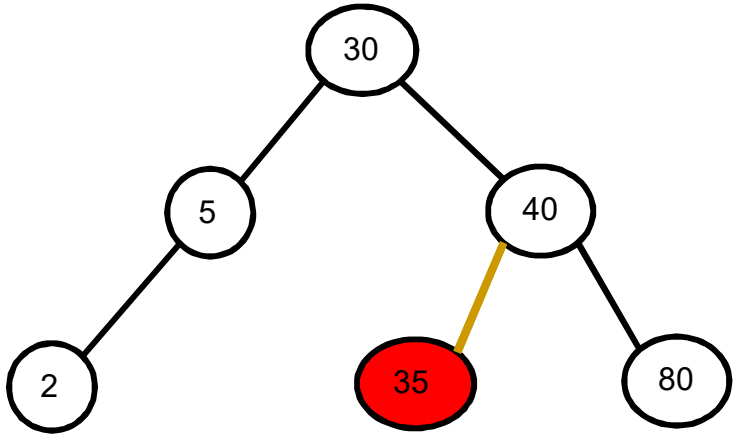


درج عنصر در درخت جستجوی دودویی

- زمان لازم برای جستجوی num در یک درخت برابر O(h) می باشد به نحوی که h برابر با عمق یا ارتفاع درخت است.
- بقیه الگوریتم نیاز به زمان Θ(1) دارد. بنابراین زمان کل مورد نیاز insert_node برابر با Θ(h) می باشد.

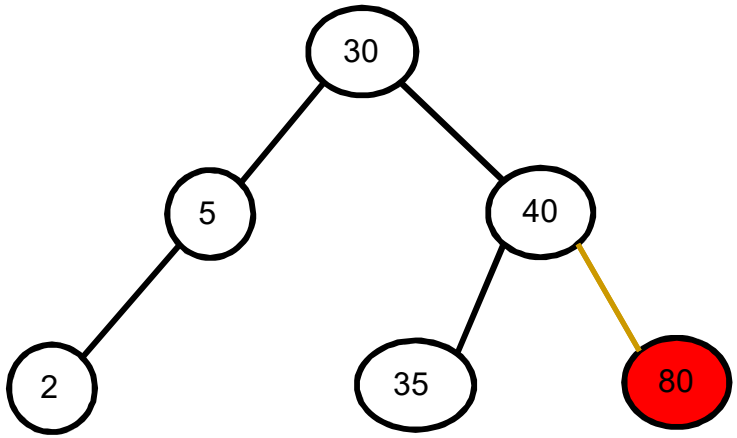
حذف عنصر از درخت جستجوی دودویی

- برای حذف فرزند چپ کافی است اشاره گر فرزند چپ والد حذف شود.
- برای حذف ۳۵ از درخت زیر باید فیلد فرزند چپ والد این گره را برابر NULL قرار داده و گره را آزاد نمود :



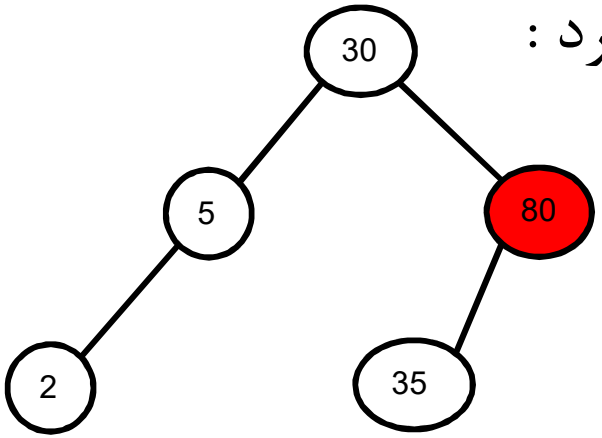
حذف عنصر از درخت جستجوی دودویی

- برای حذف فرزند راست کافی است اشاره گر فرزند راست والد حذف شود.
- برای حذف ۸۰ از درخت زیر باید فیلد فرزند راست والد این گره را برابر NULL قرار داده و گره را آزاد نمود :



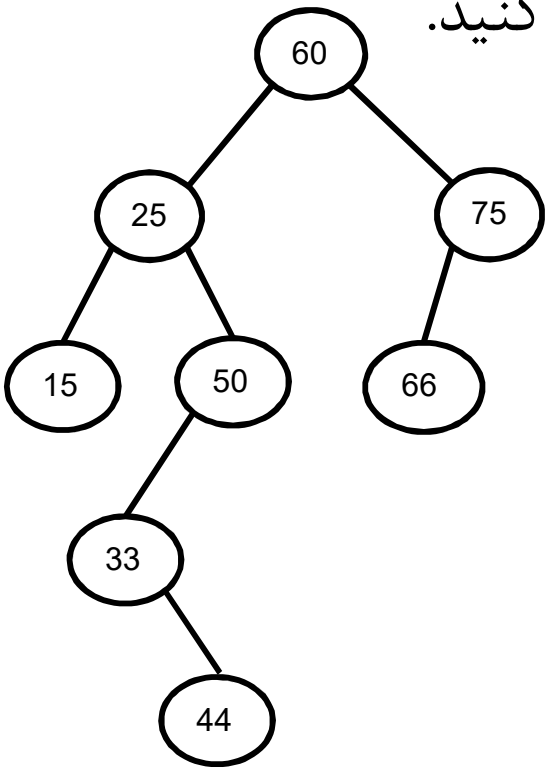
حذف عنصر از درخت جستجوی دودویی

- زمانی که یک گره با دو فرزند حذف می شوند ، گره را با بزرگترین عنصر در زیر درخت چپ و یا کوچکترین عنصر در زیردرخت راست آن گره جایگزین و تعویض کرد.
- برای حذف ۴۰ از درخت زیر به دلیل اینکه این گره والد دو گره ۳۵ و ۴۰ است پس از حذف عنصری که در پیمایش inorder بعد از آن ظاهر می شود(۸۰) به جای آن قرار می گیرد :



حذف عنصر از درخت جستجوی دودویی

- تمرین: عناصر ۴۴، ۷۵ و ۲۵ را به ترتیب از درخت BST زیر حذف کنید. در هر مرحله درخت حاصل را ترسیم کنید.



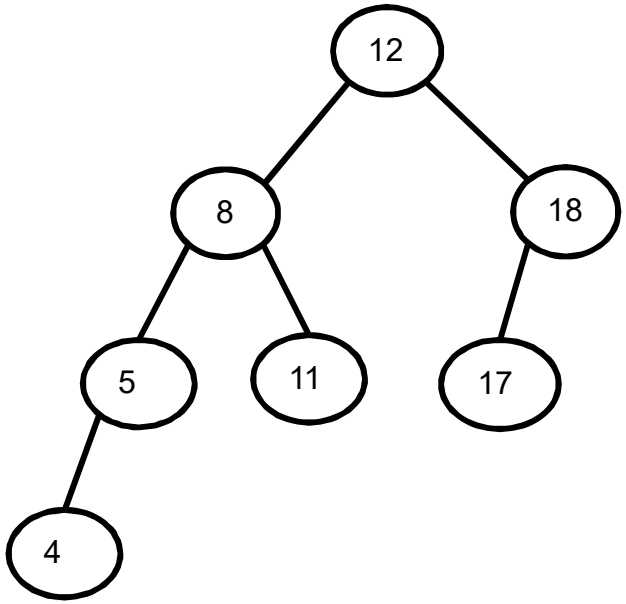
- نکته: عمل حذف در زمان O(h) انجام می گیرد(h عمق درخت)

درخت جستجوی دودوئی متعادل

- درخت جستجوی دودوئی متعادل درختی است که به طور خود کار ارتفاع خود را پس از هر عملیات درج و حذف دلخواه حفظ می کند.
- درختان جستجو با بیشترین عمق $O(\log_2 n)$، درختان جستجوی دودوئی متعادل نامیده می شوند.
- نمونه هایی از آن،درخت 2-3، AA، AVL و ... هستند.
- درختان جستجوی متعادلی وجود دارند که عمل جستجو ، درج و حذف را در زمان O(h) ممکن می سازند از جمله درخت AVL

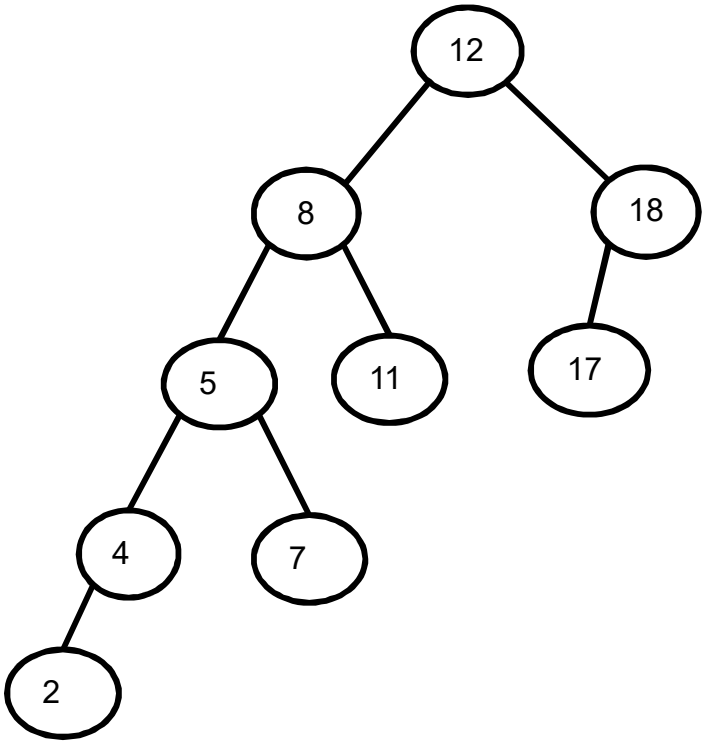
درخت AVL

- درخت جستجوی دودویی است که ارتفاع آن متوازن باشد.
- درخت با ارتفاع متوازن درختی است که حداکثر اختلاف ارتفاع دو زیردرخت چپ و راست هر گره آن یک باشد.
- مثال:



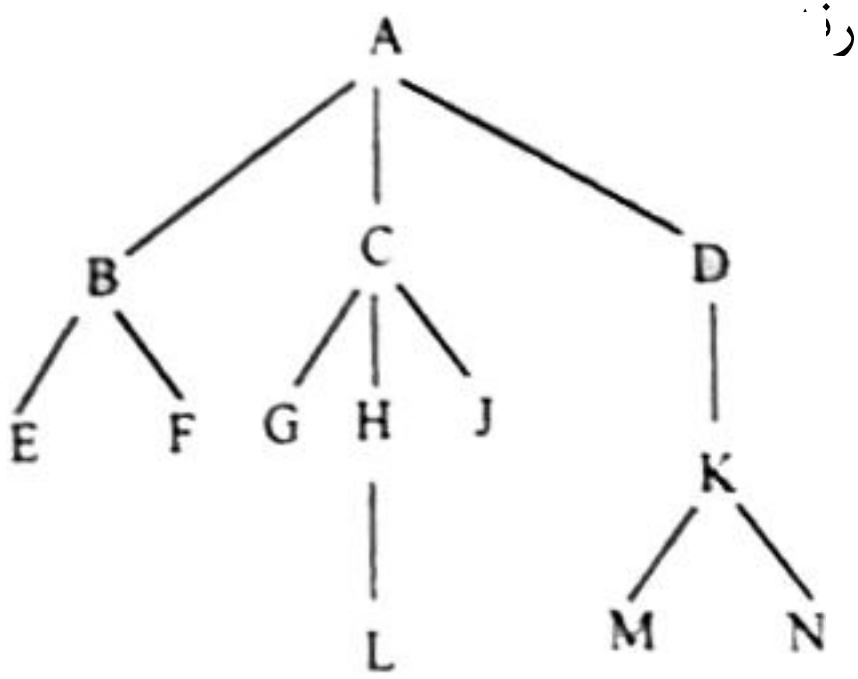
درخت AVL

- مثال: درخت زیر AVL نیست



درخت عمومی (General)

- درخت k تایی است که در آن فقط یک گره به نام ریشه با درجه ورودی صفر وجود دارد و سایر گره ها دارای یک کمان ورودی هستند. برای سهولت فرض می شود درخت عمومی مرتب است یعنی گره های فرزند ترتیب دارند

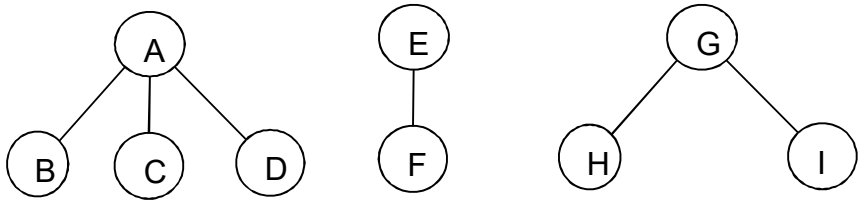


جنگل ها

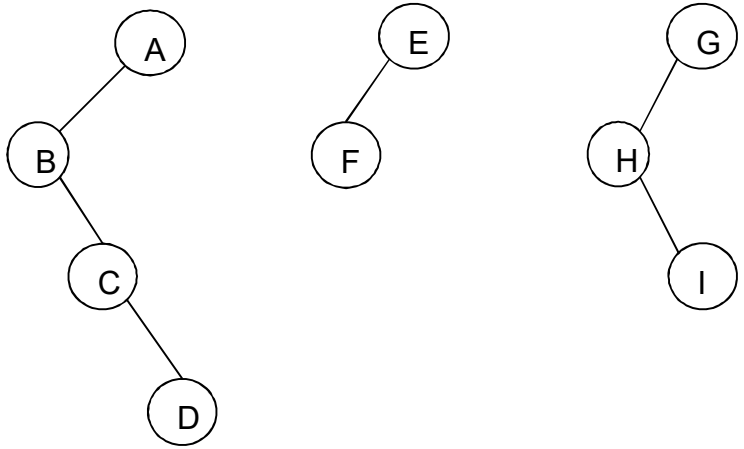
- یک جنگل مجموعه ای مرتب از صفر یا چند درخت متمایز است. به عبارت دیگر جنگل، مجموعه ی $N \geq 0$ درخت مجزاست
- مفهوم جنگل بسیار مشابه مفهوم درخت است. زیرا با حذف ریشه درخت، جنگل به وجود می آید
- تبدیل جنگل به درخت دودویی:
  - ابتدا نمایش دودویی هر یک از درختان جنگل را به دست می آوریم
  - سپس تمام درختان دودویی را از طریق فیلد همزاد گره ریشه به یکدیگر متصل می کنیم.

تبدیل جنگل به درخت دودویی

- مثال:

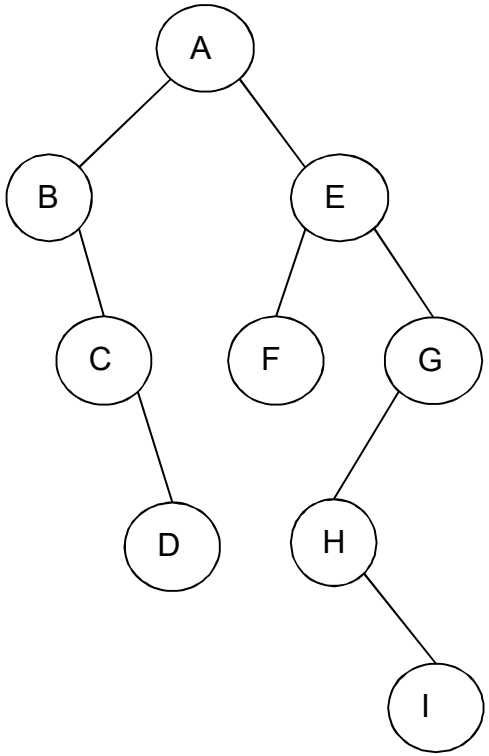


- ابتدا این سه درخت را به درخت دودوئی معادل تبدیل می کنیم:



تبدیل جنگل به درخت دودویی

- سپس از چپ به راست ریشه هر درخت را به عنوان فرزند راست درخت سمت چپ در نظر می گیریم:

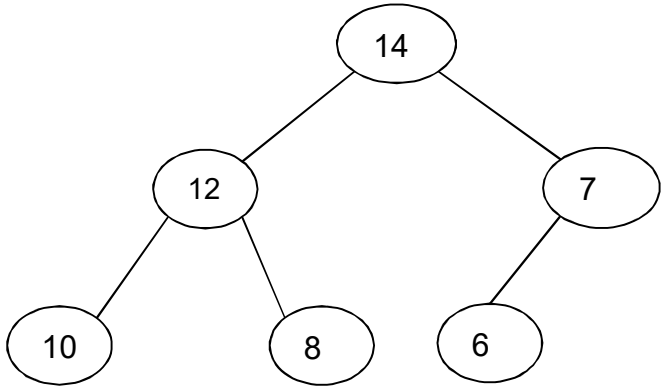


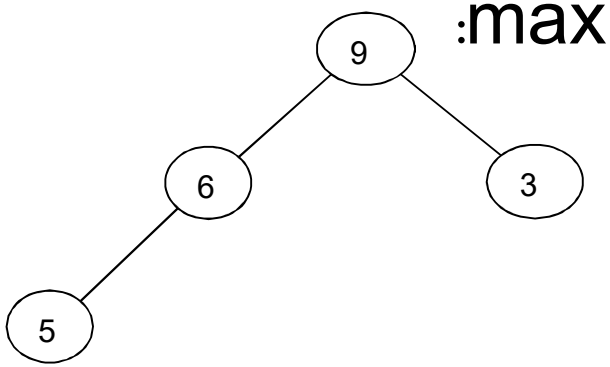
هرم (Heap)

- max tree: درختی است که مقدار کلید هر گره کمتر از مقادیر کلیدهای فرزندانش نباشد.
- max heap: یک درخت دودویی کامل است که یک max tree نیز می باشد.
- min tree: درختی است که مقدار کلید هر گره بیشتر از مقادیر کلیدهای فرزندانش نباشد.
- min heap: یک درخت دودویی کامل است که یک min tree نیز می باشد.

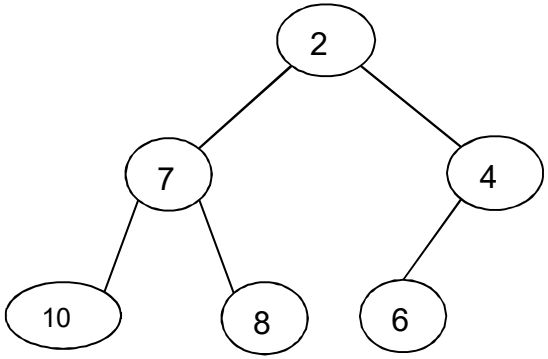
هرم (Heap)

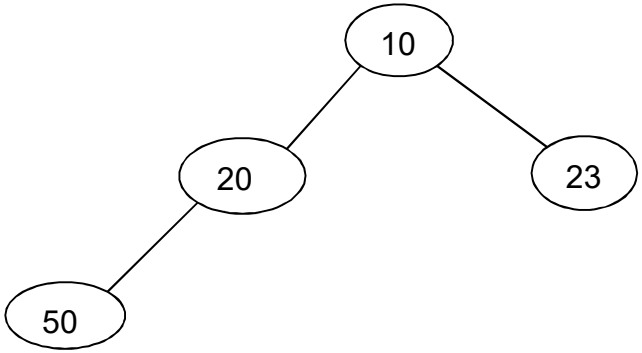
- مثال max heap:





- مثال min heap:





اعمال اساسی بر روی heap

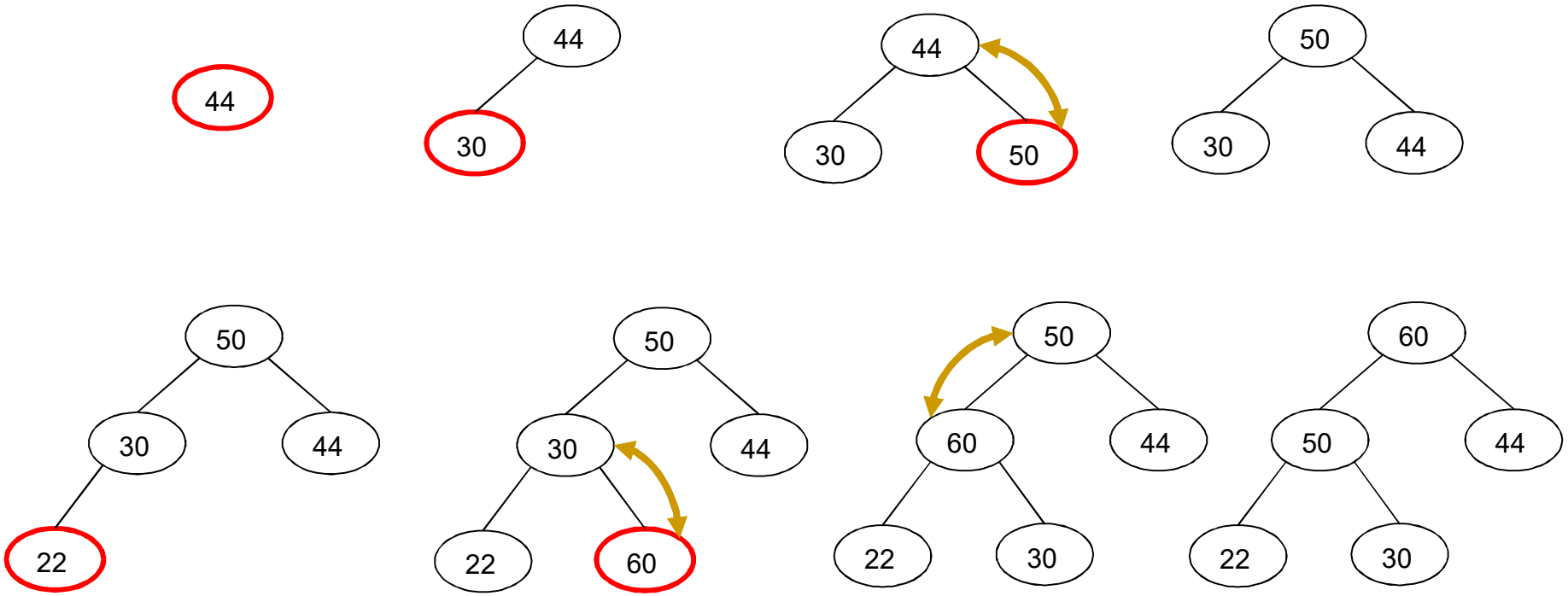
- ایجاد یک هرم(heap)
- جایگذاری عنصر جدید به هرم (heap)
- حذف بزرگترین عنصر از هرم (heap)

درج عنصر در هرم

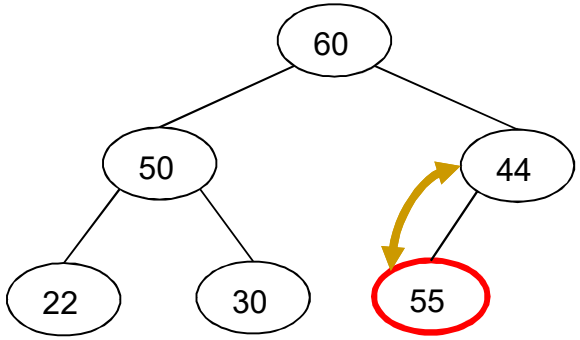
- با اضافه و حذف کردن یک عنصر در هرم باید درخت به صورت هرم باقی بماند در نتیجه این عملیات نیازمند فعالیت های اضافه جهت تنظیم درخت خواهد بود.
- اضافه نمودن گره به هرم:
- همواره درخت از سمت چپ به راست در سطح آخر پرشده و با پر شدن هر سطح به سطح بعدی می رویم. پس از اضافه کردن هر عنصر عمل مرتب سازی انجام می شود. در این عمل یک گره در پایین ترین سطح تا جایی که لازم باشد با گره پدرش جابجا می شود.

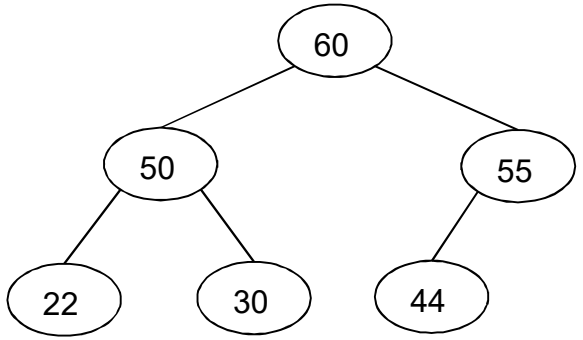
درج عنصر در هرم

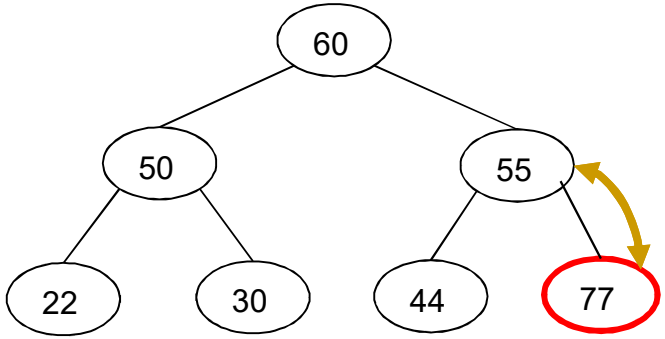
- مثال: با درج اعداد ۴۴، ۳۰، ۵۰، ۲۲، ۶۰، ۵۵، ۷۷ را به ترتیب در یک MaxHeap وارد نمایید.

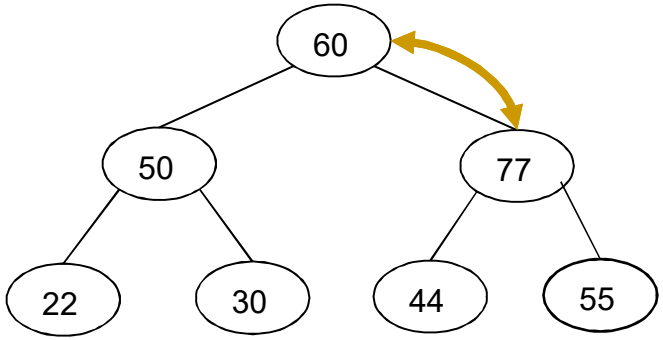


درج عنصر در هرم



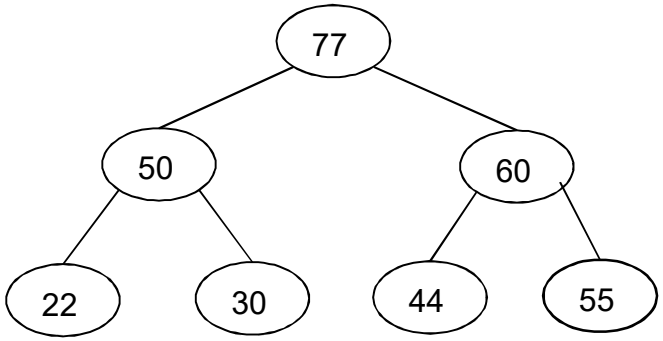






درج عنصر در هرم

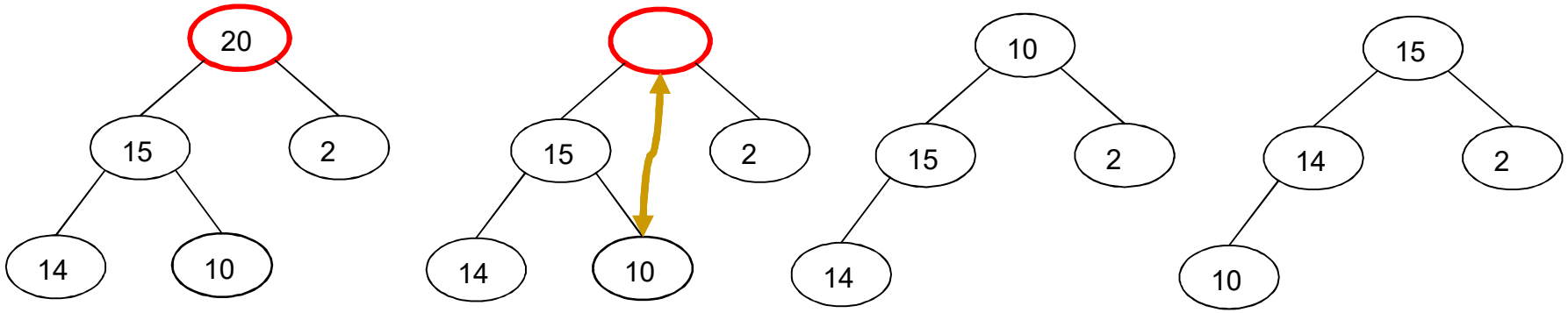
- نتیجه نهایی



- از آنجا که heap یک درخت کامل با n عنصر است، دارای ارتفاع $\log_2(n)$ می باشد. به این معنی که حلقه به میزان $\log_2(n)$ تکرار شود. بنابراین پیچیدگی تابع درج برابر $\log_2(n)$ می باشد.

حذف عنصر از هرم

- در عمل حذف همواره ریشه هرم حذف می شود و سمت راست ترین برگ به جای آن ریشه قرار می گیرد.سپس باید درخت مجددا تنظیم شود.
- مثال:



- پیچیدگی زمانی حذف یک عنصر از هرم برابر $O(\log_2 n)$ می باشد

کاربردهای هرم

- یکی از کاربردهای هرم، پیاده سازی صف اولویت استفاده می شوند.
- در صف اولویت ها عنصری که دارای بالاترین (یا پایین ترین) اولویت است ، حذف می شود.
- آرایه ساده ترین نمایش برای یک صف اولویت می باشد.

نوع نمایش	درج	حذف
آرایه نامرتب	$\Theta(1)$	$\Theta(n)$
لیست پیوندی نامرتب	$\Theta(1)$	$\Theta(n)$
آرایه مرتب	$O(n)$	$\Theta(1)$
لیست پیوندی مرتب	$O(n)$	$\Theta(1)$
هرم	$O(\log_2 n)$	$O(\log_2 n)$