**پیمایش درختان دودویی:**

**هدف از پیمایش ملاقات گره های درخت است  حداکثر برای یک بار.**

پیمایش های مختلفی که روی درخت دودویی می توان انجام داد عبارتند از:

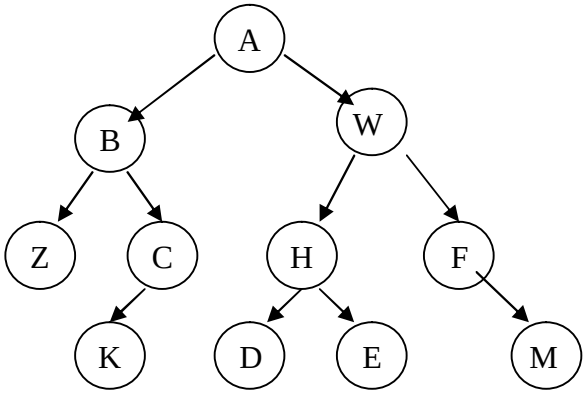
1)`inorder`(پیمایش میانوندی)

2)`preorder`(پیمایش پیشوندی)

3)`postorder`(پیمایش پسوندی)

4) پیمایش سطحی `levelOrder`

مثال: درخت زیر موجود است:




1) پیمایش میانوندی درخت را بنویسید.

2) پیمایش پیشوندی درخت را بنویسید.

3) پیمایش پسوندی درخت را بنویسید.

الگوریتم پیمایش میانوندی به روش بازگشتی :

```
void  Tree::InOrder( TNode * t)
{
   if (t) {
     InOrder( t->LeftChield);
     printf("%d″, t->Data;);
     InOrder( t-> RightChield);
   }
}
```

تمرین: توابع پیمایش پیشوندی و پسوندی را به روش بازگشتی بنویسید.

برای نوشتن الگوریتم های پیمایش به روش غیر بازگشتی باید از یک پشته استفاده کنیم.

تمرین: یکی از این پیمایش ها را به روش غیر بازگشتی بنویسید.

پیمایش سطحی :

در این روش بجای پشته از صف برای ذخیره کردن آدرس گره ها استفاده می شود.

```
void  Tree::LevelOrder (TNode * t)
{
     TNode  * temp;
     Queue q;

     q.addQ (t);
     while ( ! q.isEmpty () ) {
        temp = q.deleteQ ();
        if (temp -> LeftChield) q.addQ (temp->
LeftChield);
        if (temp -> RightChield)q.addQ (temp->
RightChield);
        printf ("%d" , temp -> Data);
```

```
        }
}
```

درخت جستجوی دودویی :

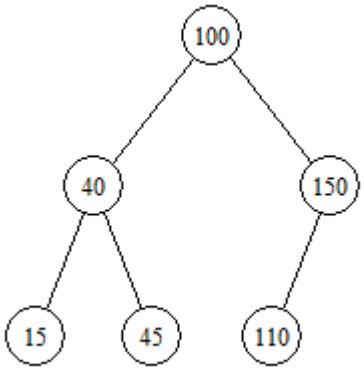
یک درخت دودویی است که خواص زیر را دارد :

(1) هر گره درخت یک فیلد به نام فیلد کلید (شاخص) دارد که منحصر به فرد است.

(2) شاخص گره سمت چپ از شاخص گره ریشه اش کوچکتر است.

(3) شاخص گره سمت راست از شاخص گره ریشه اش بزرگتر است.

در شکل زیر یک درخت جستجوی دودوئی با 6 گره را مشاهده می کنید.




الگوریتم حذف یک گره از درخت جستجوی دودویی :

اگر گره مورد نظر از برگ ها باشد به سادگی حذف میشود. اگر گره مورد نظر از درجه 1 باشد به سادگی حذف میشود و فرزند آن به جای آن قرار میگیرد. اگر گره مورد نظر از درجه 2 باشد آنگاه به یکی از روشهای زیر عمل میکنیم :

الف) کوچکترین عنصر از زیر درخت سمت راست را پیدا کرده و جای آن را با گره مورد نظر عوض می کنیم و حذف را روی آن عنصر کوچک انتخاب شده انجام می دهیم.

ب) بزرگترین عنصر در زیردرخت سمت چپ را پیدا کرده و آن را با گره مورد نظر عوض می کنیم ، سپس آن را حذف می کنیم.

نکته: در هر یک از این دو روش با حذف گره ها ، درخت های یکسانی بدست نمی آید.

## هرم Heap

Max-Tree
Min-Tree

Max-Heap
Min-Heap

درج در Max-heap

جذف از Max-heap

کاربرد Max-heap

# گراف

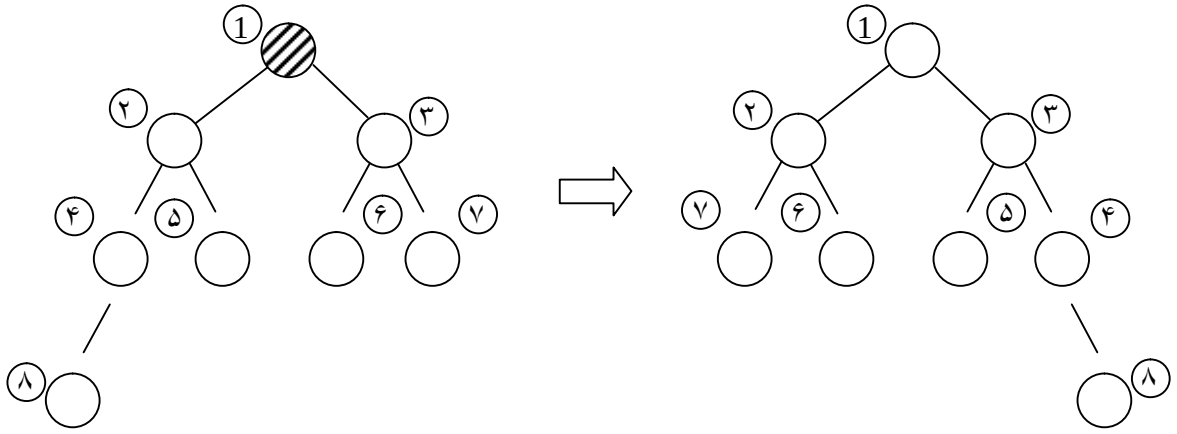
مثال : تابعی به زبان C بنویسید که با استفاده از آن تعداد گره های برگ یک درخت دودویی را مشخص کند. زمان اجرایی این تابع چقدر است؟

(راهنمایی : تعداد برگ های دودویی :

تعداد برگ های زیر درخت سمت راست + تعداد برگ های زیر درخت سمت چپ)

مثال : تابعی به نام swapchields بنویسید که در یک درخت دودویی و فرزندان راست و چپ هر گره را عوض نماید.

با استفاده از پیمایش preorder این کار را انجام می دهیم.




```
void  swapchields (struct tnode & t)
{
    if (t)  {
        struct node * temp;
        temp = t -> left;
        t -> right = temp;
        Swap-tree (t -> left);
        Swap-tree (t -> right);
    }
}
```

مثال : تابعی بنویسید که گره ای که حاوی یک عدد num است از یک لیست پیوندی حلقوی حذف کند.

```
void delete_by_key (struct list * L , int num)
```

مثال : تابعی به زبان C بنویسید که اگر کلمهای حالت آینه ای داشته باشد مقدار true و اگر در غیر این صورت بود false را برگشت دهد. (با استفاده از پشته).

تمرین : تابعی بنویسید که مشخص کند که تعداد Aها با تعداد Bها برابر است. (با استفاده از پشته)

تمرین : تابعی به زبان C بنویسید که درستی پرانتزهای یک عبارت را بررسی کند. (با استفاده از پشته)

مثلا : a + (C+D+(x*y)  عبارت درست

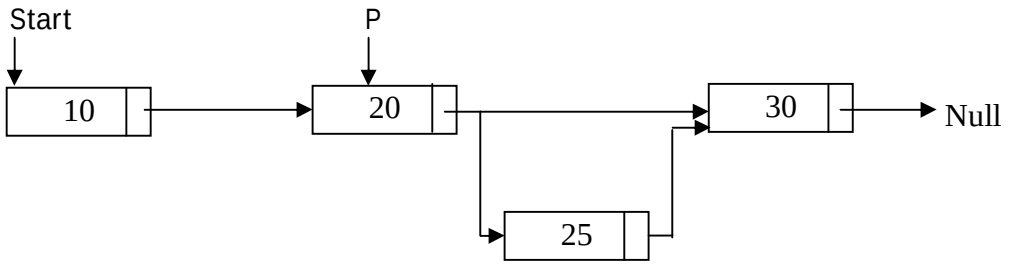
(x+(y+z)  عبارت نادرست

از چپ به راست عبارت را پردازش می کنیم. اگر به پرانتز باز برسیم پرانتز را وارد پشته می کنیم. و اگر به پرانتز برسیم یک پرانتز باز از پشته خارج میکنیم. (اگر پشته خالی نباشد پس تاکنون جمله درست است و اگر پشته خالی باشد و نتوان از آن عنصری خارج کرد ، عبارت نادرست پرانتز گذاری شده است.) و در آخرکار پشته باید خالی باشد. خالی بودن پشته نشان می دهد که تعداد پرانتز باز و بسته برابر است.

مثال : لیست مرتب : لیستی که نودهایش ترتیب داشته باشد (چه نزولی ، چه صعودی). الگوریتم درج در این لیست را بنویسید.

ابتدا لیست را پیمایش می کنیم و عددی را که می خواهیم به لیست اضافه میکنیم.

(باید قبل از عددی قرار بگیرد که از آن کوچکتر باشد.)




مثال : دو لیست z و y مفروضند به طوری که نودهای آنها به ترتیب صعودی مرتب است. تابعی بنویسید که دو لیست x و y را در یک لیست (z) ادغام کند به طوری که نودهای z مرتب باشد. هیچ حافظه جدیدی درخواست نمی شود.