فایل هشتم درس ساختمان های داده

- فایل آماده شده برای جلسه ادامه مبحث ساختار درخت و صرفا برای کلاس آنلاین در این ترم می باشد.

همواره سلامت و موفق باشید- اسبقی

پیمایش های درخت های دودویی

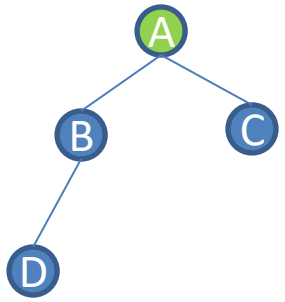
- برای درخت های زیر پیمایش های درخت را استخراج نمایید.

Preorder: A B D C

Inorder:D B A C

Postorder:D B C A

Levelorder:A B C D

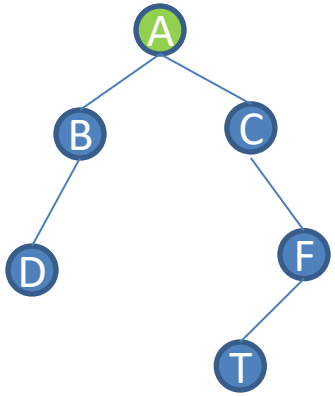


Preorder: A B D C F T

Inorder:D B A C T F

Postorder:D B T F C A

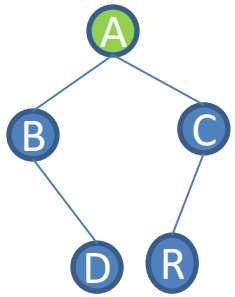
Levelorder:A B C D F T



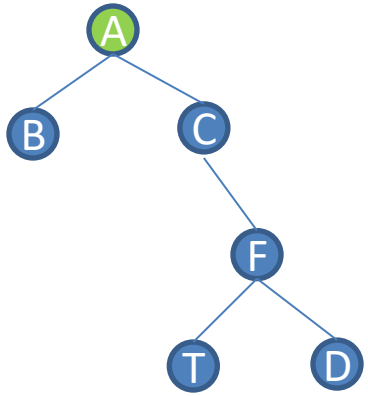
پیمایش های درخت های دودویی

- برای درخت های زیر پیمایش های درخت را استخراج نمایید.

Preorder: A B D C R
Inorder: B D A R C
Postorder:D B R C A
Levelorder:A B C D R



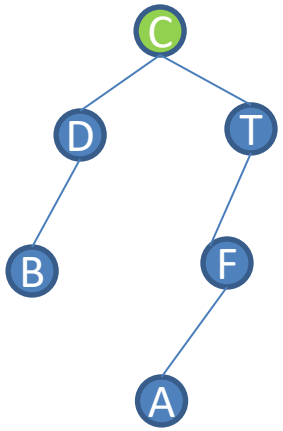
Preorder: A B C F T D
Inorder: B A C T F D
Postorder:B T D F C A
Levelorder:A B C F T D

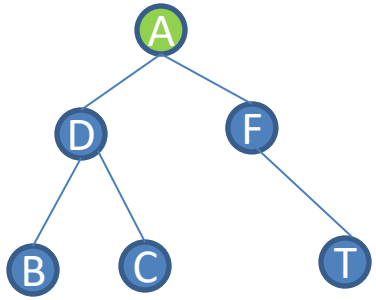


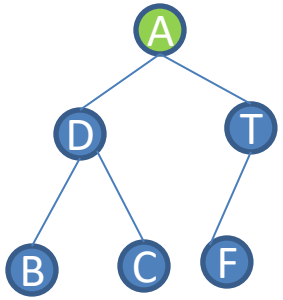
ساخت درخت از روی پیمایش های درخت

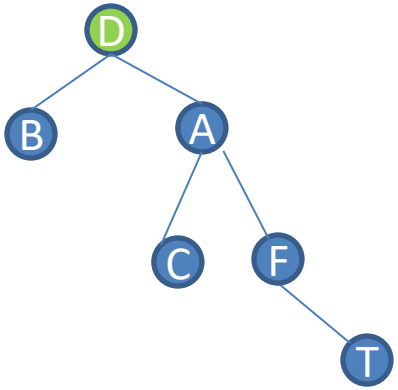
- پیمایش میانوند زیر را در نظر بگیرید:

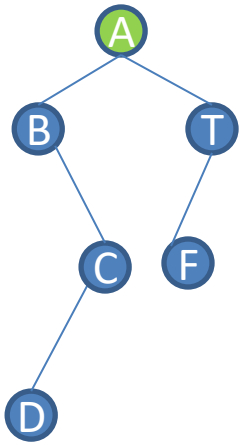
Inorder: BDCAFT







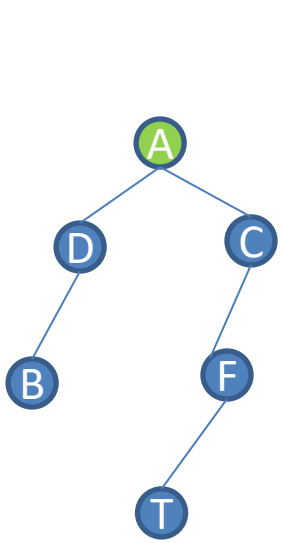


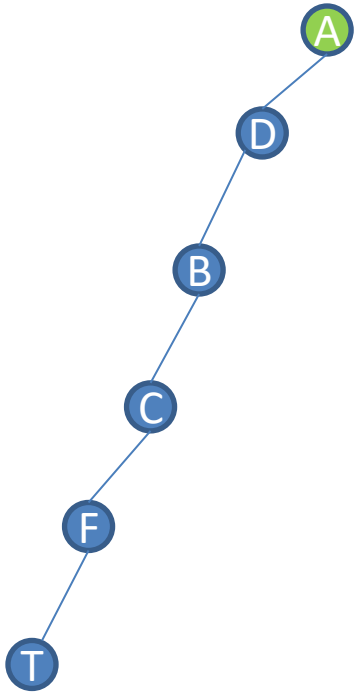


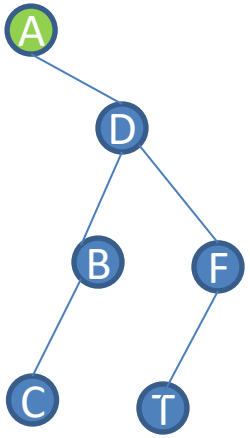
ساخت درخت از روی پیمایش های درخت

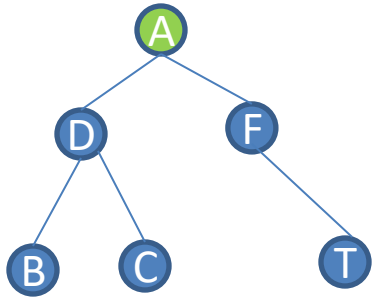
- پیمایش پیشوند زیر را در نظر بگیرید:

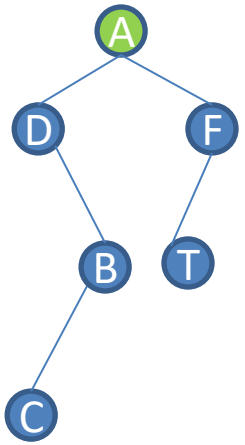
preorder: ADBCFT











ساخت درخت از روی پیمایش های درخت

با توجه به مثال های مطرح شده مشخص گردید:

-از روی هر یک از پیمایش های پیشوند، میانوند و پسوند بصورت جداگانه، می توان درخت های دودویی متعددی را ایجاد نمود، ولی اگر درخت دودویی یکتای اولیه مدنظر باشد، بایستی حداقل دو پیمایش درخت (inorder , preorder) یا (inorder, postorder) در دسترس باشد.

preorder: برای مشخص شدن ریشه درخت(در پیمایش پیشوند، ریشه همواره اولین گره پیمایش شده است.

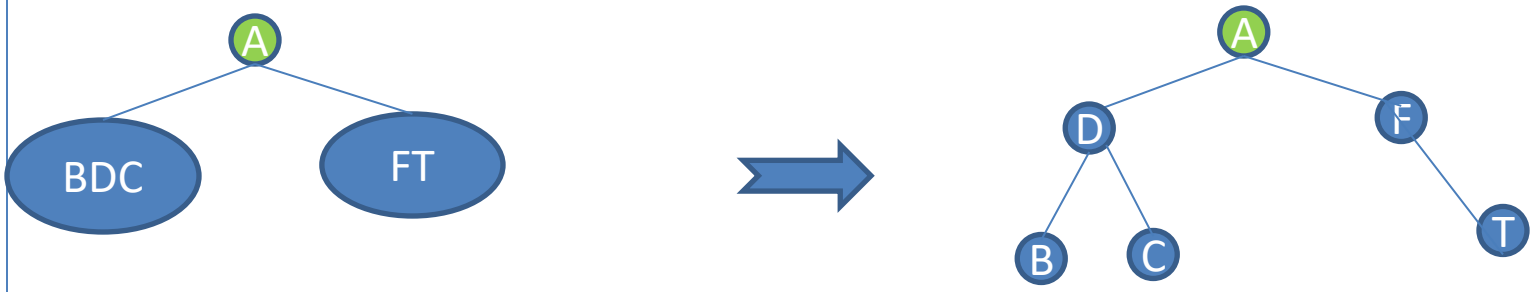
Postorder:برای مشخص شدن ریشه درخت(در پیمایش پسوند، ریشه همواره آخرین گره پیمایش شده است.

Inorder:برای مشخص شدن فرزندان گره(در پیمایش میانوند، یک گره همواره بین فرزندان سمت چپ و فرزندان سمت راست قرار می گیرد.)

ساخت درخت از روی پیمایش های درخت

Preorder: ADBCFT

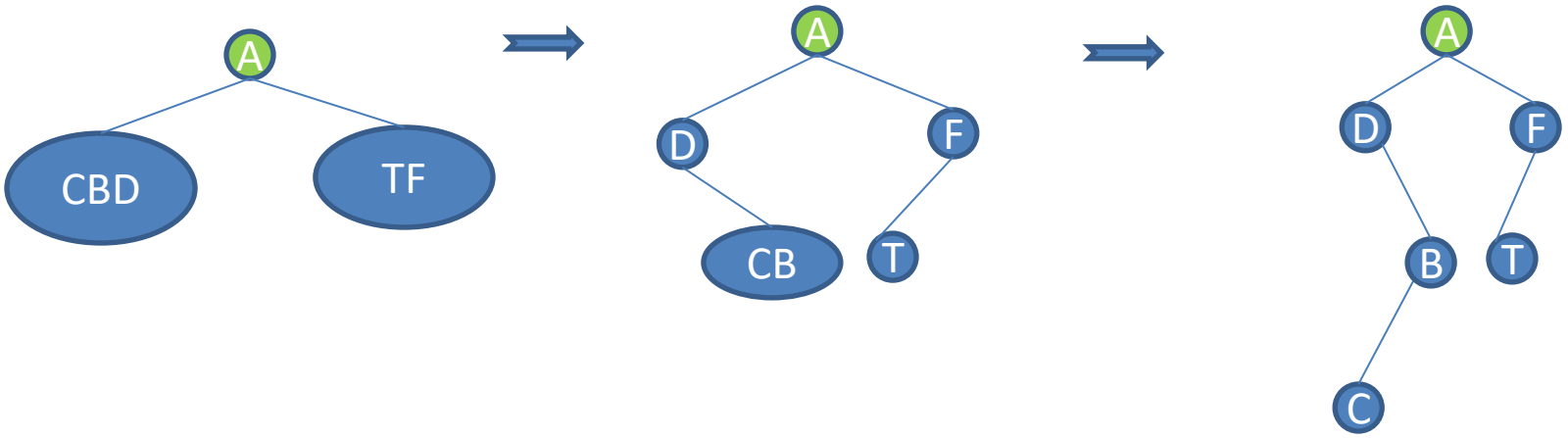
Inorder: BDCAFT



ساخت درخت از روی پیمایش های درخت

Postorder: CBDTFA

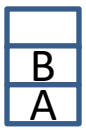
Inorder: DCBATF



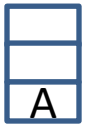
پیمایش های درخت های دودویی

```
Inorder(Tree_Node  *root){
  if (root!=null)
   {  stack    s;
     Tree_Node   *Temp = root;
    while(true)
     {
        while( Temp !=null)
          { push(s, Temp);
           Temp = Temp->Lchild;}
        if (s  isn't  empty)
           { Temp = pop(s);
             cout<<Temp->data;
             Temp = Temp->Rchild;
             }
        else
            break;
     }
}}
```

Temp = A
Temp = B
Temp = null



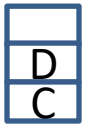
Temp = B
Temp = null



Temp = A



Temp = C
Temp = D
Temp = null

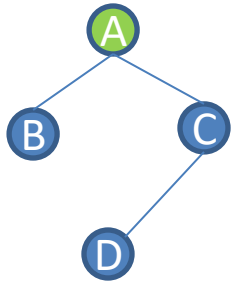


Temp = D
Temp = null



Temp = C
Temp = null





inorder: B A D C

پیمایش سطحی(level order)

- پیاده سازی پیمایش سطحی با استفاده از ساختار صف:

```
levelorder(Tree_Node  *root){
  if (root!=null)
   {  Queue    q;
      Tree_Node   *Temp = root;
     while(Temp!=null)
      {
         cout<<Temp->data;
         if (Temp->Lchild!= null)
            Add_q(q,Temp->Lchild);
          if (Temp->Rchild!= null)
            Add_q(q,Temp->Rchild);
              Temp = Remove_q();
      }
}
```

Temp = A



Temp = B



Temp = C



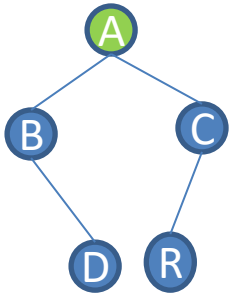
Temp = D



Temp = R



Temp = null



levelorder: ABCDR

درخت جستجویی دودویی(Binary Search Tree)(BST)

آرایه زیر را در نظر بگیرید:

5	7	2	8	4	6	1

برای جستجوی x=10 چندین عمل مقایسه در آرایه بالا لازم است؟

درخت جستجوی دودویی، راهکاری برای بهینه سازی عملیات جستجو و کاهش تعداد مقایسه ها می باشد.

خصوصیات درخت BST:

-برای هر عنصر یک مقدار کلیدی در نظر گرفته می شود.

-کلید فرزندان چپ، کوچکتر از ریشه می باشد.

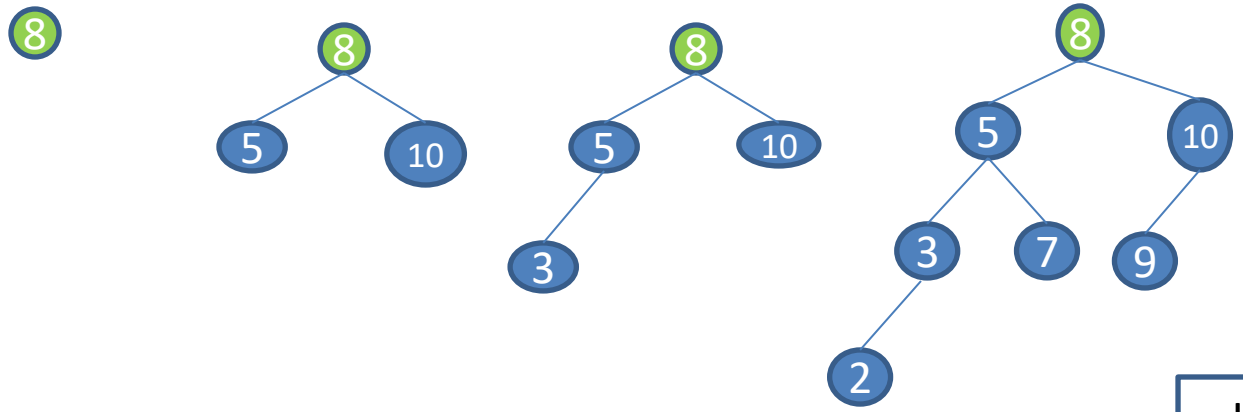
-کلید فرزندان راست، بزرگتر از ریشه می باشد.

درخت جستجویی دودویی(Binary Search Tree)(BST)

- درج عنصر در درخت BST:

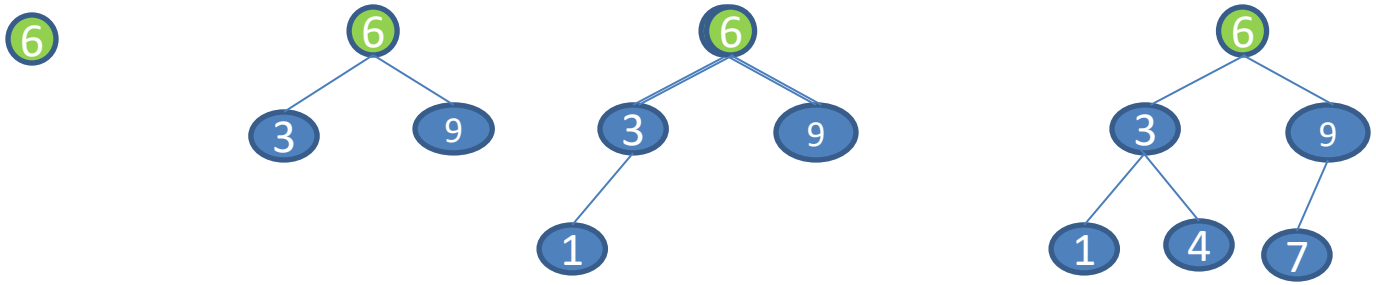
A)8, 5,10, 3, 7, 2, 9

Inorder:2,3,5,7,8,9,10



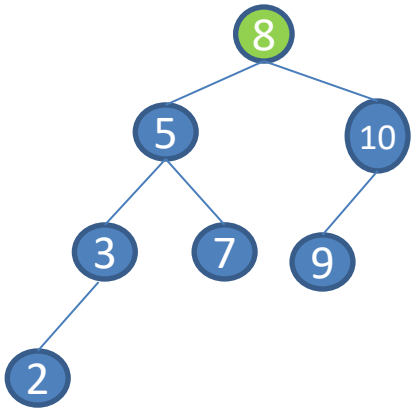
B)6, 9, 3, 1, 4, 7

Inorder:1,3,4,6,7,9



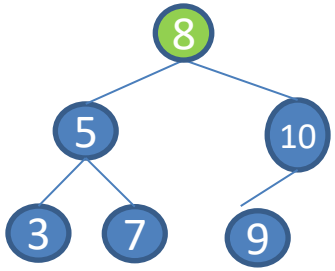
درخت جستجویی دودویی(Binary Search Tree)(BST)

- حذف عنصر از درخت جستجوی BST:

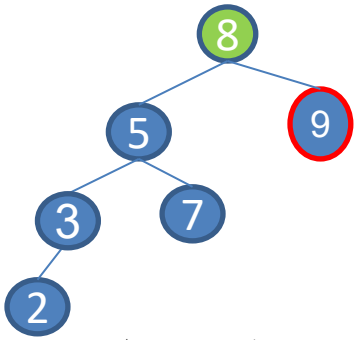


الف)حذف گره برگ (مانند گره 2)

این عمل حذف، بدون هیچ مشکل و تغییری انجام می شود.



ب)حذف گره با یک فرزند(مانند گره 10)



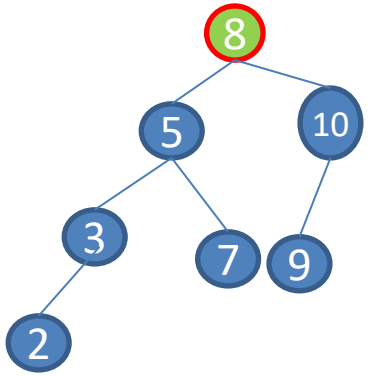
در اینحالت، عمل حذف انجام شده و تک فرزند آن گره، جایگزین گره حذف شده می گردد.

ج)حذف گره با دو فرزند چپ و راست(مانند گره 8)

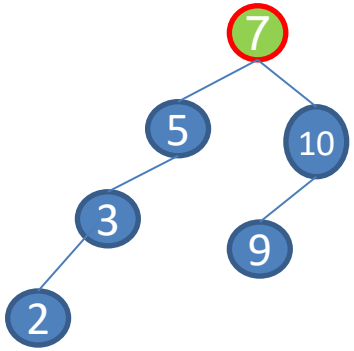
برای حذف یک گره با دو فرزند، لازم است، یا بزرگترین فرزند شاخه سمت چپ و یا کوچکترین فرزند شاخه سمت راست، جایگزین گره حذف شده گردد.

درخت جستجویی دودویی(Binary Search Tree)(BST)

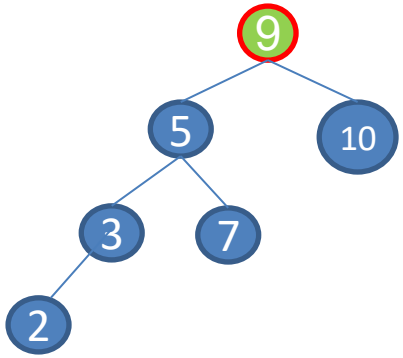
ج)حذف گره با دو فرزند چپ و راست(مانند گره 8)



جایگزینی گره 7 بعنوان بزرگترین فرزند سمت چپ



جایگزینی گره 9 بعنوان کوچکترین فرزند سمت راست



درخت جستجویی دودویی(Binary Search Tree)(BST)

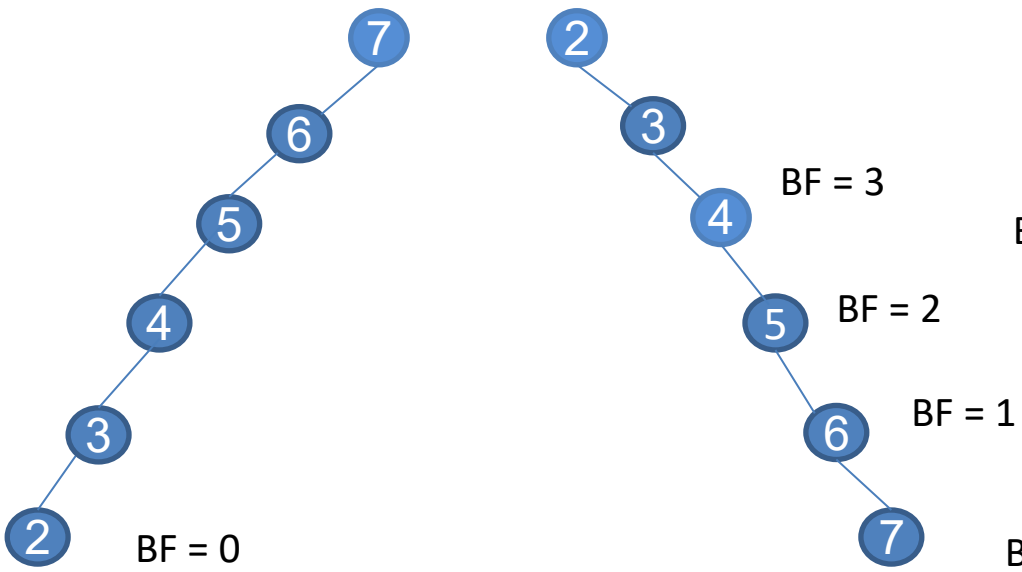
در درخت جستجوی دودویی، برای جستجوی یک مقدار، به اینصورت عمل می شود که اگر مقدار مورد نظر با ریشه درخت برابر باشد که مقدار 1 بعنوان موفقیت آمیز بودن جستجو برگشت داده می شود و اگر مقدارمورد جستجو از ریشه کوچکتر باشد که شاخه سمت چپ و اگر بزرگتر از ریشه باشد که شاخه سمت راست پیمایش خواهد شد.

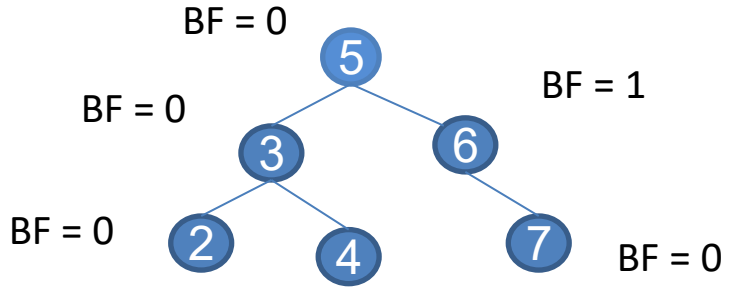
```
int  BST(BST_node  *root,  int  x)
{   if(root == null)
        return  0;
    elseif (root->key == x)
        return  1;
    elseif(root->key>x)
        return  BST(root->Lchid, x);
    else
        return BST(root->Rchild, x);}
```

- بعنوان تمرین، الگوریتم جستجوی دودویی را بصورت غیر بازگشتی بنویسید.

درخت AVL

- درخت های زیر را در نظر بگیرید:





در 3 درخت جستجوی دودویی، فقط درخت سوم، تعداد مقایسه را کمتر و عملیات جستجو را بهینه می کند.
درخت های متوازن AVL با چیدمان متوازن گره های درخت در هر دو شاخه چپ و راست، باعث بهینه سازی عملکرد جستجو می گردند.
معیار توازن (Balance Factor) ⟸ |hr-hl | >=1

درخت دودویی heap

- درخت دودویی heap در دو شکل زیر تعریف می شود:

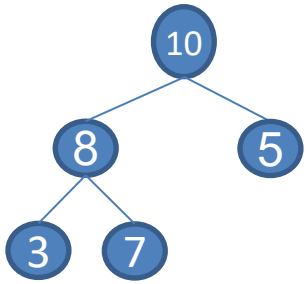
- max heap

- min heap

خصوصیات درخت max heap:

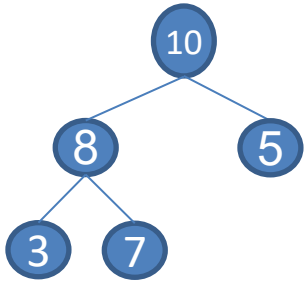
- درخت max heap یک درخت کامل است.

- مقادیر گره ریشه، از تمامی فرزندان بیشتر است، بطوریکه این روند در سطح کلیه گره ها برقرار است.

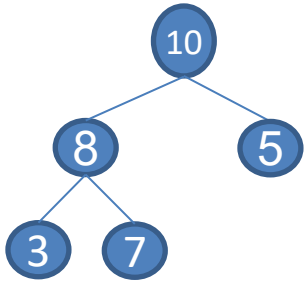


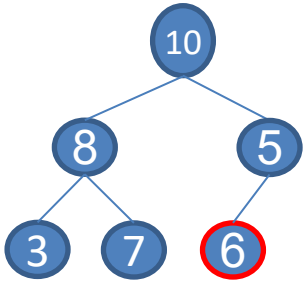
درخت دودویی heap

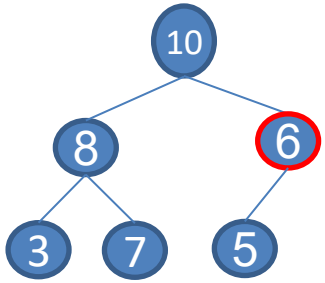
- درج گره جدید در max heap:



- درج گره 6

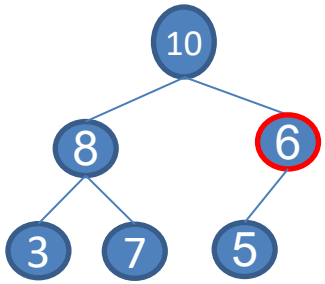


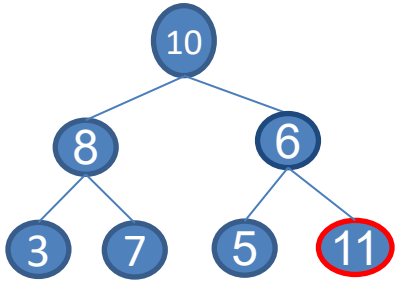


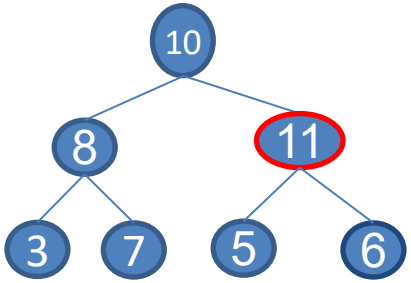


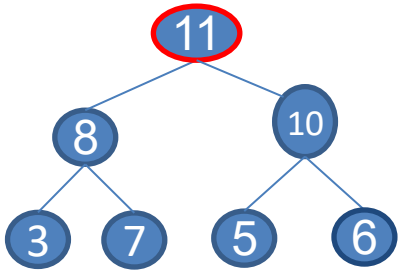
درخت دودویی heap

- درج گره جدید در max heap:



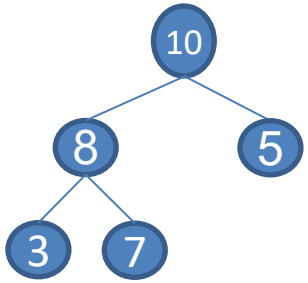


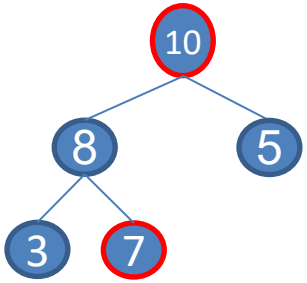


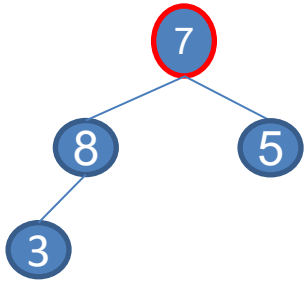


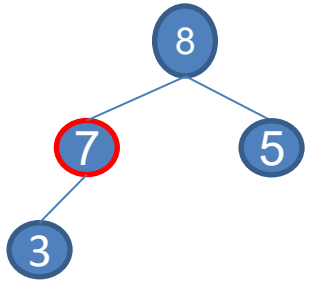
درخت دودویی heap

- حذف گره ریشه maxheap:



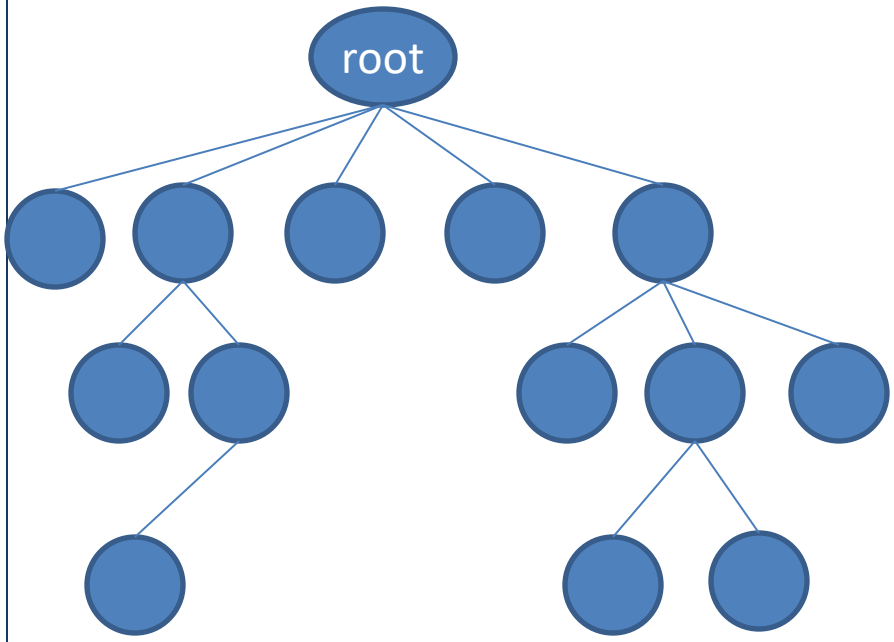






درخت عمومی

- درخت عمومی: درختی را که حداقل دارای یک گره با درجه بیش از 2 باشد، را درخت عمومی گویند.



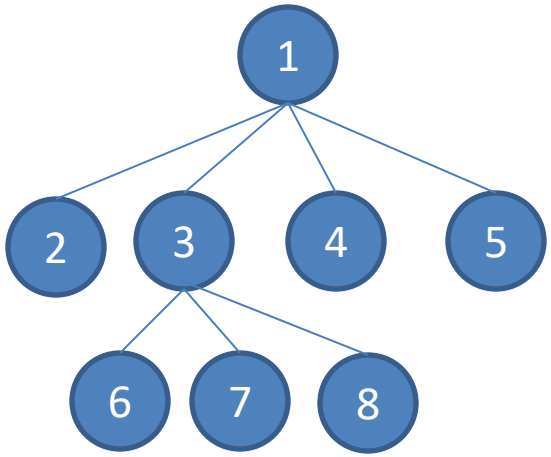
نمایش درخت عمومی

- برای نمایش درخت های دودویی از روش های زیر استفاده می شود:

الف) استفاده از آرایه

ب)استفاده ترکیبی از آرایه و لیست پیوندی

ج)استفاده از لیست پیوندی



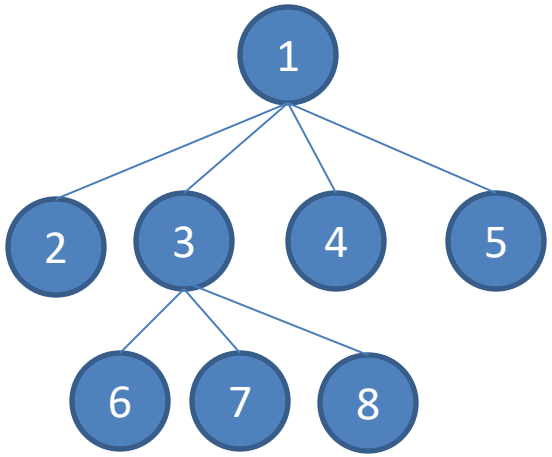
<u>پیاده سازی درخت عمومی با آرایه:</u>

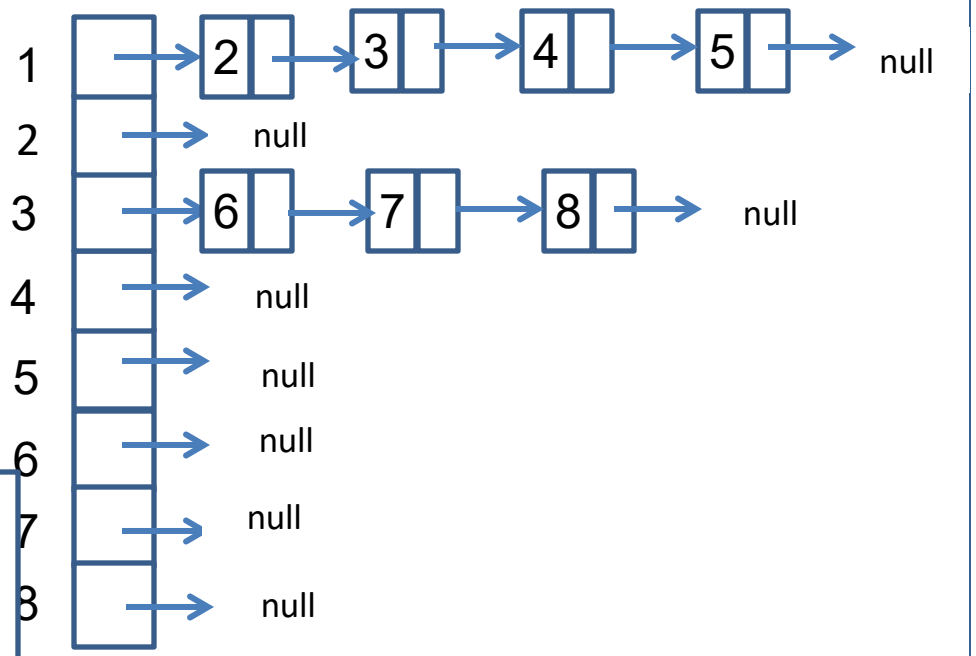
1	2	3	4	5	6	7	8
-	1	1	1	1	3	3	3

در این روش نمایش، دسترسی از فرزندان به گره پدر با مرتبه ثابت ولی دسترسی از پدر به فرزندان با مرتبه کندتری انجام می گردد.

نمایش درخت عمومی

پیاده سازی درخت عمومی با استفاده از آرایه و لیست پیوندی:





در این روش نمایش، محتوای هر محل آرایه، اشاره گری است به شروع یک لیست پیوندی شامل تمامی فرزندان ان گره.

نمایش درخت عمومی

پیاده سازی درخت عمومی با لیست پیوندی(تبدیل درخت عمومی به درخت دودویی)

