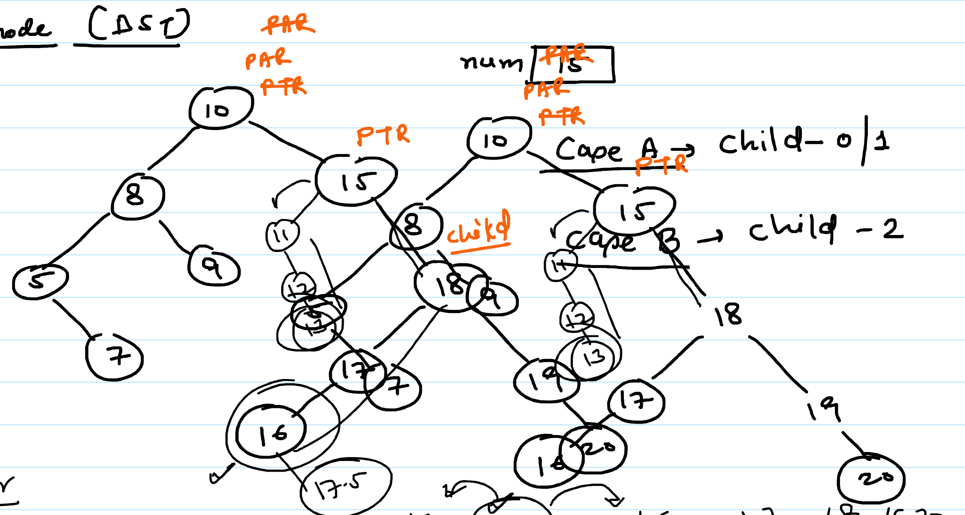
Day 20

09 April 2025 08:48 AM

Del node (BST)

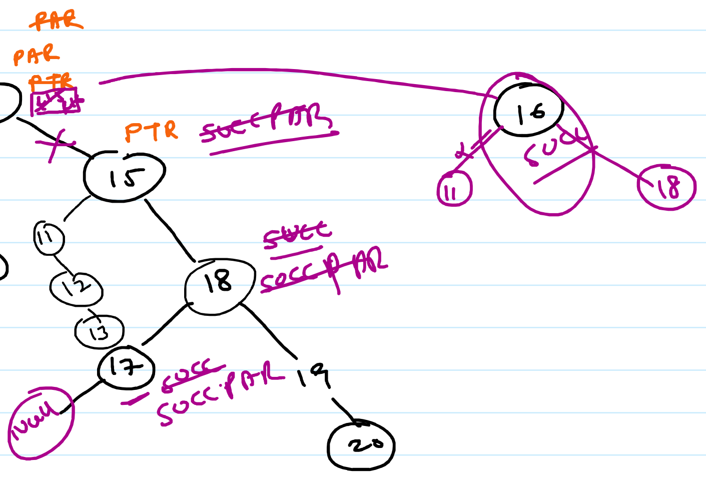


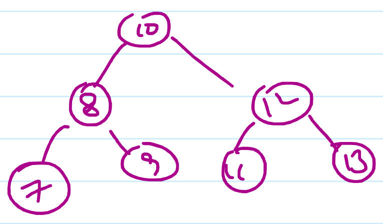
Inorder

5 7 8 9 10 11 12 13 15 16 17 18 19 20

Pre dec
↓
Left child
↓
Right most node child

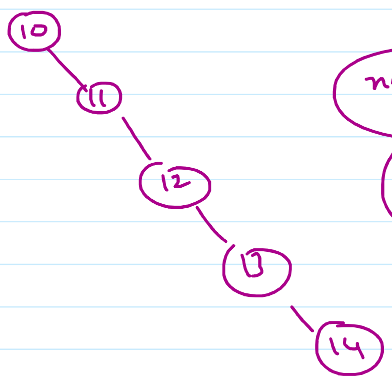
succ
↓
right child
↓
Left most node child





```
#include<iostream>
using namespace std;
class node{
    public:
        int val;
        node *left,*right;
        node(int x)
        {
            val=x;
            left=right=nullptr;
        }
};
class BST{
    node*root;
    pair<node*,node*> pp;
    public:
        BST()
        {
            root=nullptr;
        }
        bool search(int num)
        {
            node *ptr = root;
            node *par = nullptr;
            while(ptr != nullptr)
            {
                if(num == ptr->val)
                {
                    pp=pair<node*, node*>(ptr,par);
                    return true;
                }
                par = ptr;
                if(num < ptr->val)
                    ptr = ptr->left;
                else
                    ptr = ptr->right;
            }
            pp=pair<node*, node*>(ptr,par);
            return false;
        }
        void addBST(int num)
        {
            if(search(num))
            {
                cout<<"Duplicate element\n";
                return;
            }
            node* par = pp.second;
            if(par == nullptr)
            {
                root= new node(num);
            }
            else
            {
                if(num < par->val)
                    par->left = new node(num);
                else
                    par->right=new node(num);
            }
        }
        void inorder(node* root)
        {
            if(root==nullptr)
                return;
            inorder(root->left);
            cout<<root->val<<" ";
            inorder(root->right);
        }
        void traverse()
        {
            inorder(root);
            cout<<endl;
        }
        void delBST(int num)
        {
            if(! search(num))
            {
                cout<<"Element not found\n";
                return;
            }
            node *par = pp.second;
            node *ptr = pp.first;
            if(ptr->left != nullptr and ptr->right != nullptr)
            {
                caseB(ptr,par);
            }
            else
            {
                caseA(ptr,par);
            }
            delete ptr;
        }
        void caseB(node *ptr, node* par)
        {
            node *succ = ptr->right, *succpar = ptr;
            while(succ->left != nullptr)
            {
                succpar = succ;
                succ = succ->left;
            }
            caseA(succ,succpar);
            if(par == nullptr)
            {
                root = succ;
            }
            else
            {
                if(ptr == par->left)
                    par->left = succ;
                else
                    par->right = succ;
            }
            succ->left = ptr->left;
            succ->right = ptr->right;
        }
        void caseA(node *ptr, node* par)
        {
            node *child;
            if(ptr->left != nullptr)
                child = ptr->left;
            else if(ptr->right != nullptr)
                child = ptr->right;
            else
                child = nullptr;
            if(par == nullptr)
            {
                root = child;
            }
            else
            {
                if(ptr == par->left)
                    par->left = child;
                else
                    par->right = child;
            }
        }
};
int main()
{
    BST tree;
    tree.addBST(10);
    tree.addBST(8);
    tree.addBST(9);
    tree.addBST(12);
    tree.addBST(11);
    tree.addBST(13);
    tree.addBST(7);
    tree.traverse();
    tree.delBST(8);
    tree.traverse();
    tree.delBST(9);
    tree.traverse();
    return 0;
}
```

BST
unbalanced



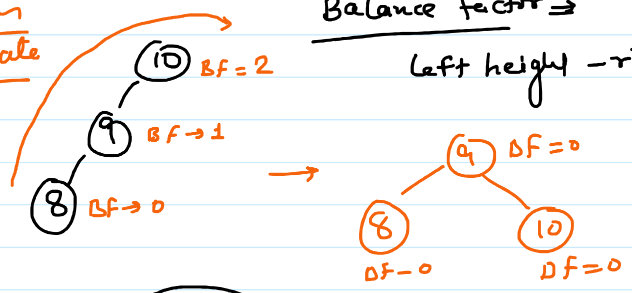
num $\boxed{13}$

O(n) ←

AVL → (Balanced BST)

L Problem

Right Rotate

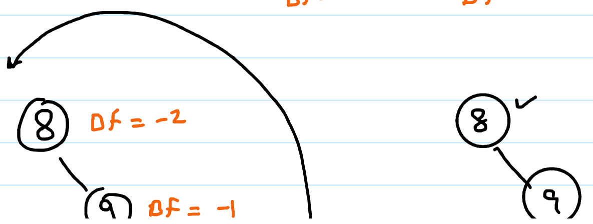


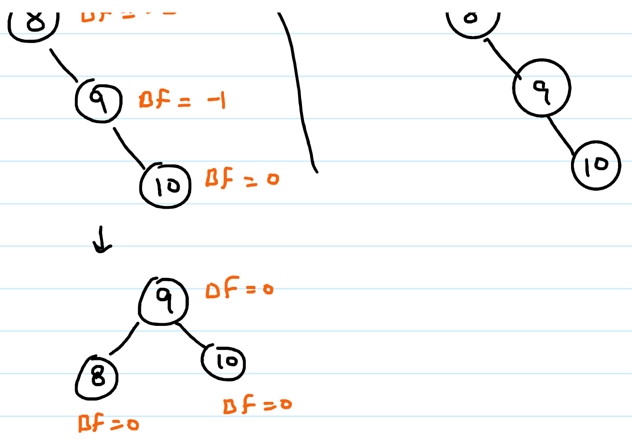
Balance factor ⇒

Left height – right height $\left\{ \begin{matrix} 0 \\ 1 \\ -1 \end{matrix} \right\}$ ✓

R-problem

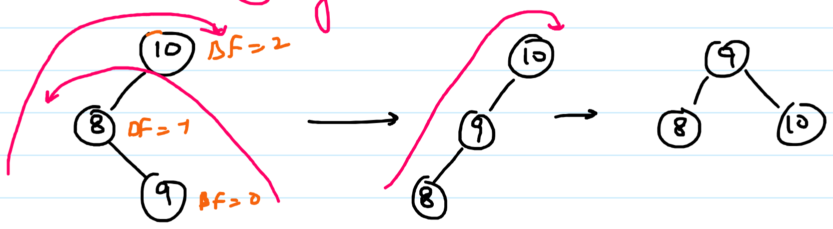
Left Rotate



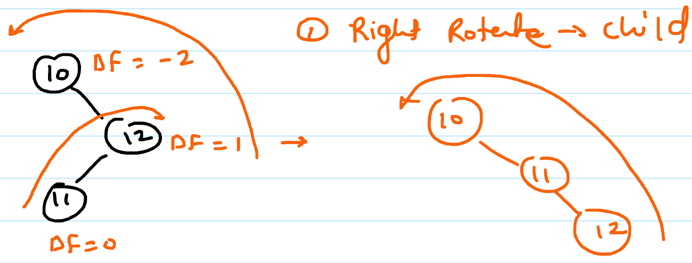


L R Problem

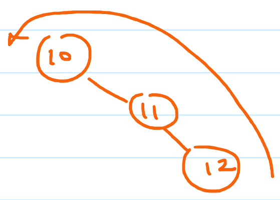
① Left Rotate – child
② Right Rotate → root



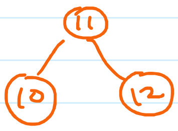
R L



① Right Rotate → child



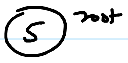
② Left Rotate → Root



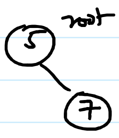
Draw AVL →

5, 7, 15, 10, 8, 12, 19, 18, 17

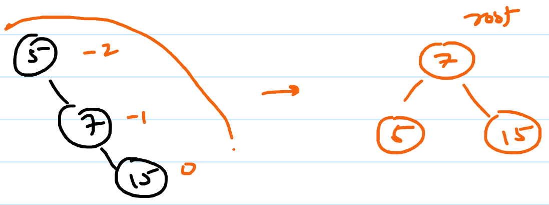
Add – 5



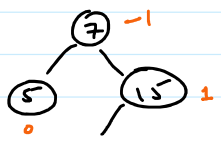
Add – 7



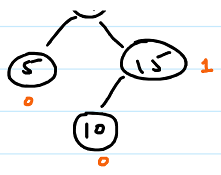
Add – 15



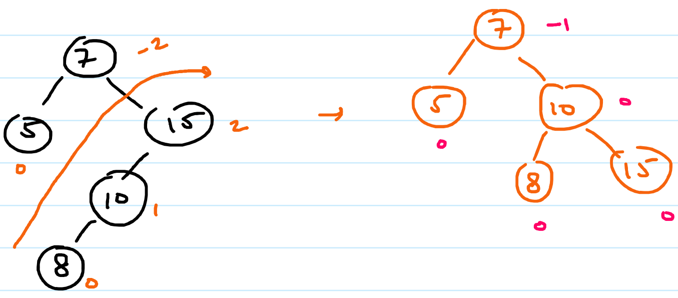
Add – 10



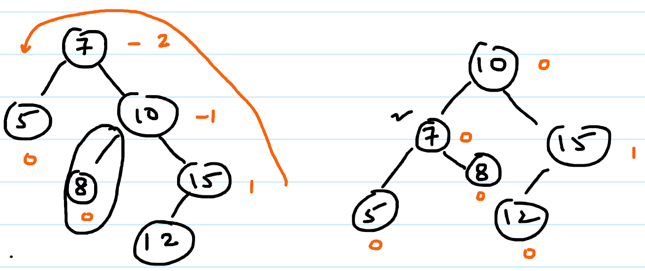
Add - 10



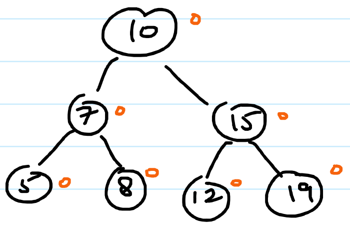
Add - 8



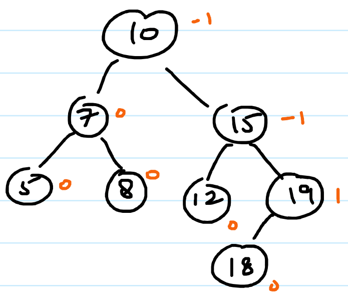
Add - 12



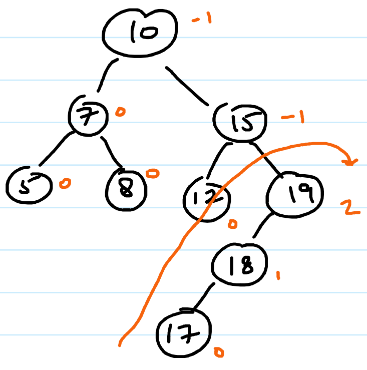
Add - 19

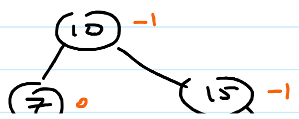


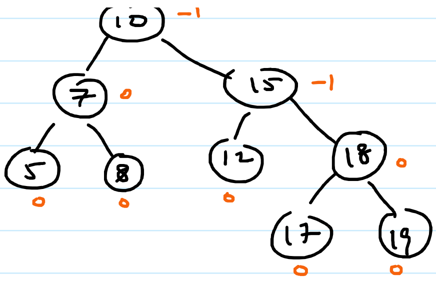
Add - 18



Add - 17

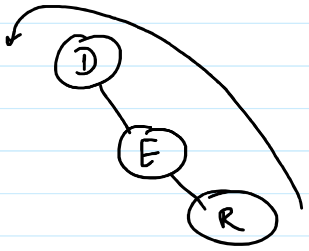


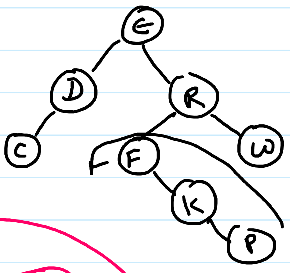


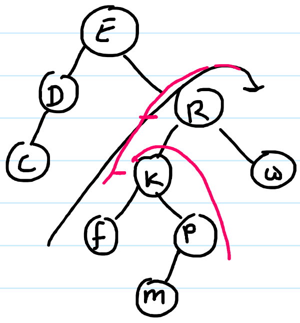


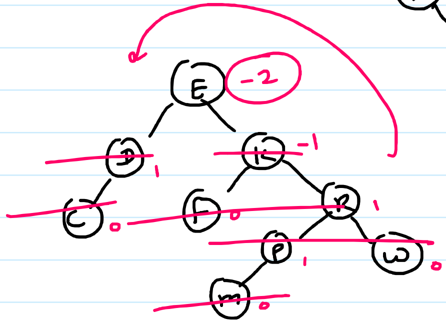
Draw AVL →

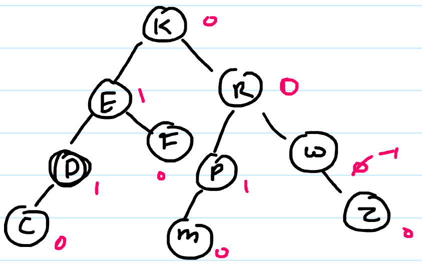
D, E, R, C, F, W, K, P, M, Z







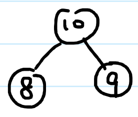




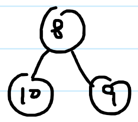
Heap tree → heap Sort ✓

(Binary tree)

1. max child – 2
2. Complete tree
3. max heap | min heap

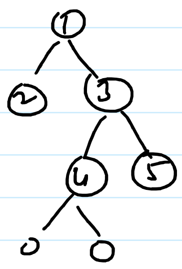


left < root > right

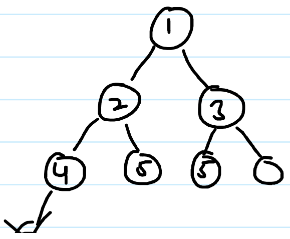


Left > root < right

① Full tree
child < 0 ✓, 2 ✓



② Complete tree



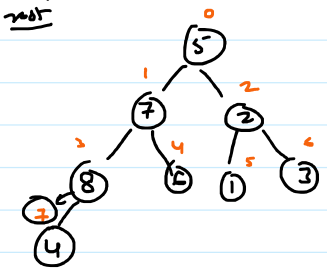
n $\boxed{8}$

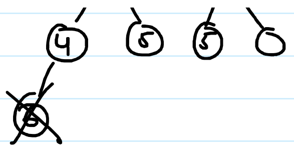
arr	5	7	2	8	6	1	3	4
	0	1	2	3	4	5	6	7

arr

5	7	2	8	6	1	3	4
0	1	2	3	4	5	6	7

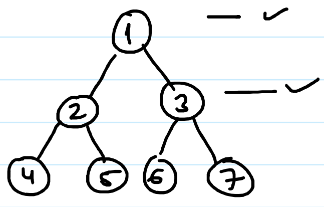
root





Left child = par * 2 + 1

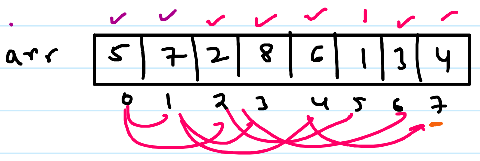
right child = left child + 1



par = (child − 1) / 2

Add → $(n \log_2 n)$

del → $(n \log_2 n)$ → $\cancel{p}o(n \log_2 n)$

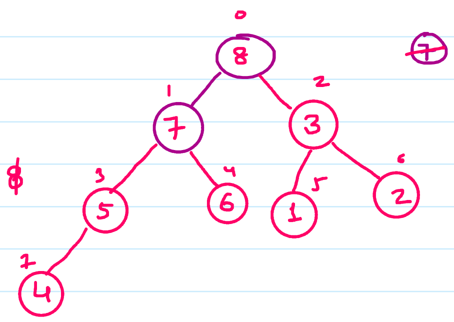


arr

5	7	2	8	6	1	3	4
0	1	2	3	4	5	6	7

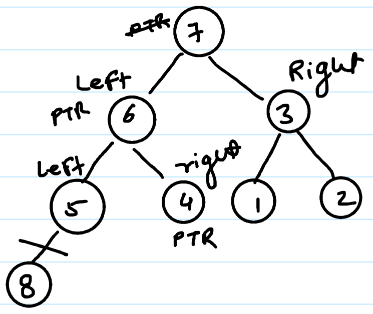
Add →

max heap



8	7	3	5	6	1	2	4
0	1	2	3	4	5	6	7

Del − 8

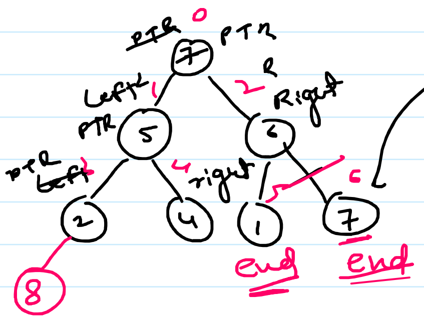


num $\boxed{8}$

Last $\boxed{4}$ ✓

7	6	3	5	4	1	2	8

Del − 7

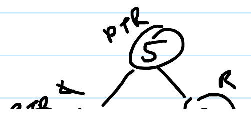


num $\boxed{7}$

Last $\boxed{2}$ ✓

en

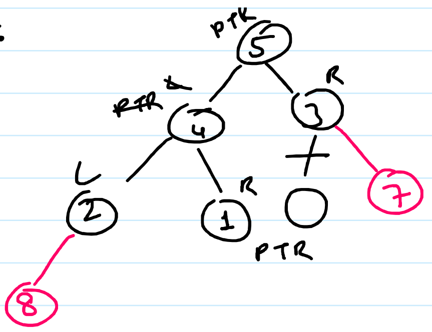
Del − 6



num $\boxed{6}$

Last

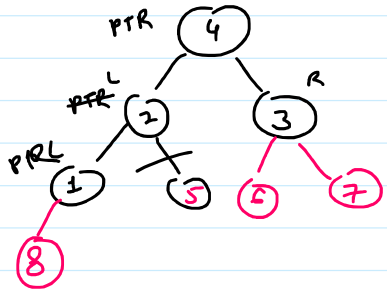
Del-6



num 6

last 1

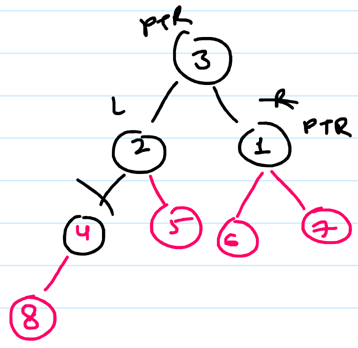
Del-5



num 5

last 1

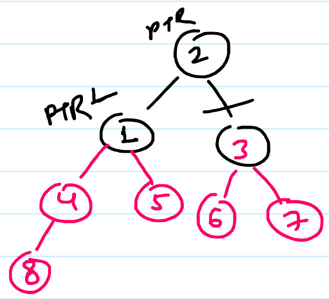
Del-4



num 4

Last 1

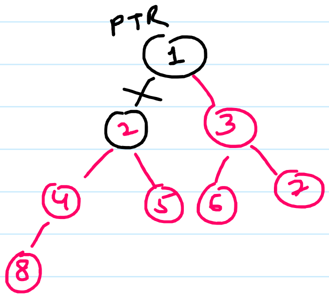
Del-3



num 3

last 1

Del-2



num 2

Last 1

1	2	3	4	5	6	7	8

↑
sorted

```
#include<iostream>
using namespace std;
void addHeap(int arr[],int last,int num)
{
    int ptr = last;
    while(ptr>0)
    {
        int par = (ptr-1)/2;
```

```
            if(num<arr[par])
            {
                arr[ptr] = num;
                return;
            }
            arr[ptr]=arr[par];
            ptr = par;
        }
        arr[0]=num;
    }
    void delHeap(int arr[], int end)
    {
        int num = arr[0];
        int last = arr[end];
        end--;
        int ptr = 0,left = 1,right=2;
        while(right <= end)
        {
            if(last > arr[left] && last>arr[right])
            {
                arr[ptr]=last;
                arr[end+1]=num;
                return;
            }
            if(arr[left]>arr[right])
            {
                arr[ptr] = arr[left];
                ptr = left;
            }
            else
            {
                arr[ptr]= arr[right];
                ptr = right;
            }
            left = ptr*2+1;
            right = left+1;
        }
        if(left == end && arr[left]>last)
        {
            arr[ptr] = arr[left];
            ptr = left;
        }
        arr[ptr] = last;
        arr[end+1]= num;
        return;
    }
    void heapSort(int arr[], int n)
    {
        //add
            for(int i=1;i<n;i++)
            {
                addHeap(arr,i,arr[i]);
            }
        //del
            for(int i=n-1 ; i>0 ; i--)
            {
                delHeap(arr,i);
            }
    }
    int main()
    {
        int arr[]={5,7,2,8,6,1,3,4};
        int n=8;
        heapSort(arr,n);
        //output
        for(int i:arr)
            cout<<i<<" ";
    }
```