DAY 19

15 May 2024 01:04 PM

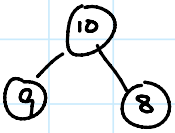
BST → $(\log_2 n)$

Add → $\log_2 n$

Build → $(n \log_2 n)$

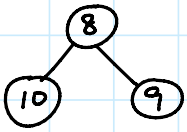
Heap tree → ① Binary tree → max child - 2

② MAX heap

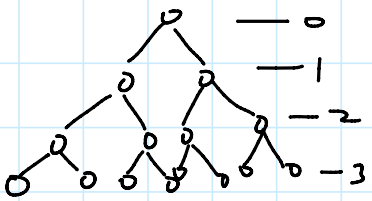


left < root > Right

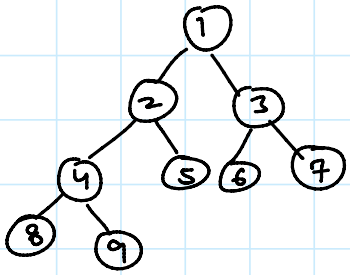
min heap



left > root < Right

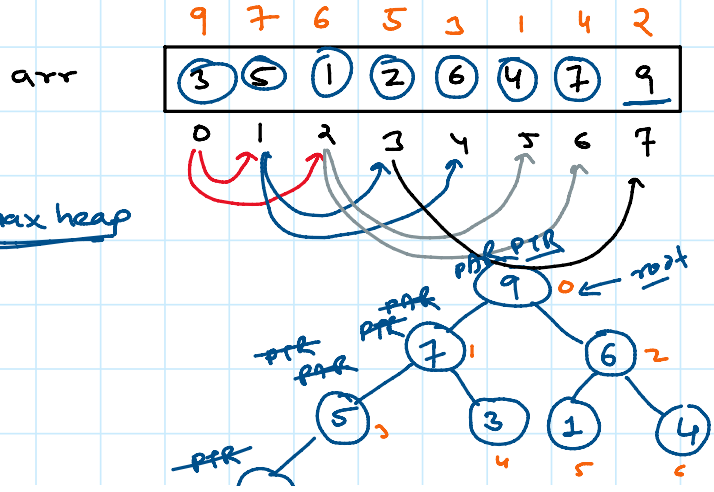


③ Complete tree :-



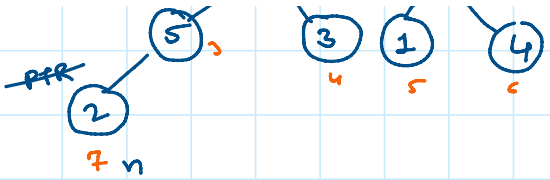
Heap Sorting :-

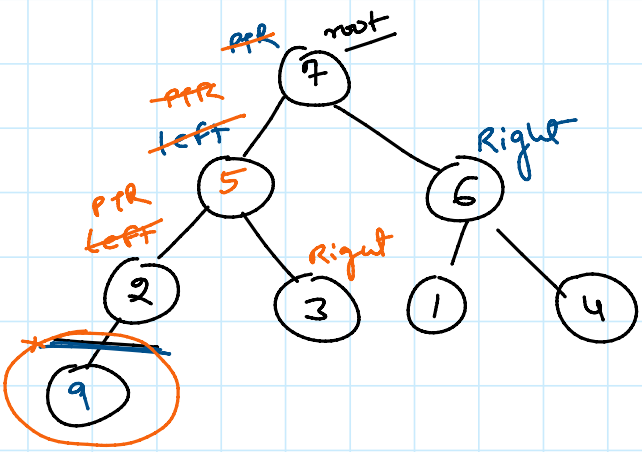
n = 7



PAR = (PTR-1)/2

element 9

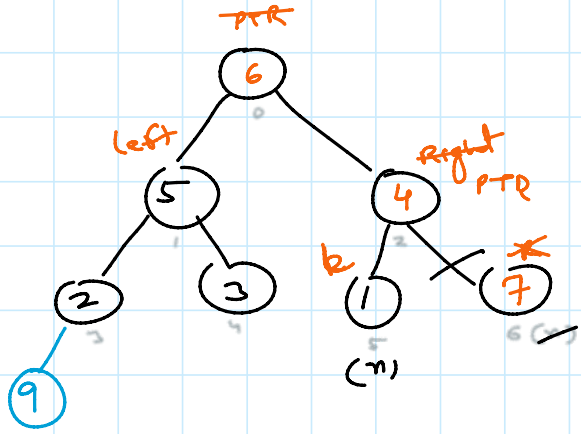




element 9

last 2

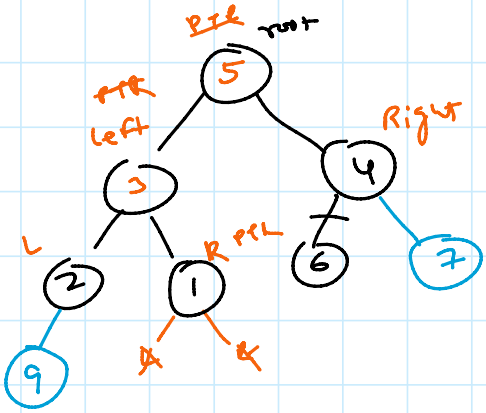
del -7



element 7

last 4

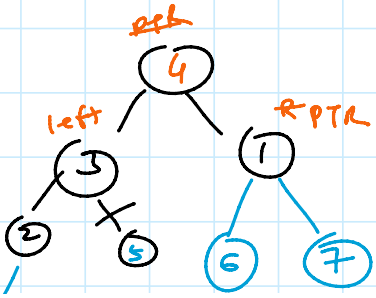
del 6



element 6

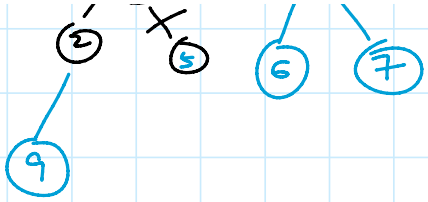
last 1

del -5

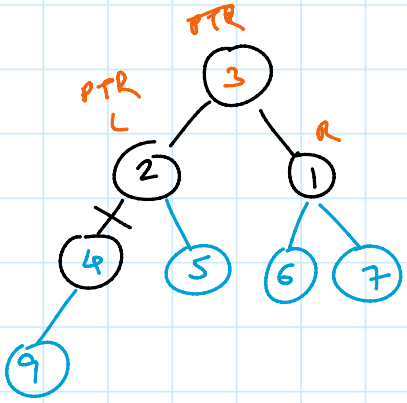


element 5

last 1



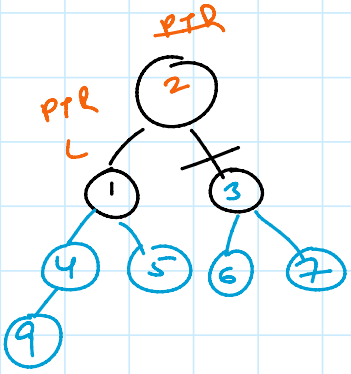
del - 4



element | 4 |

last | 2 |

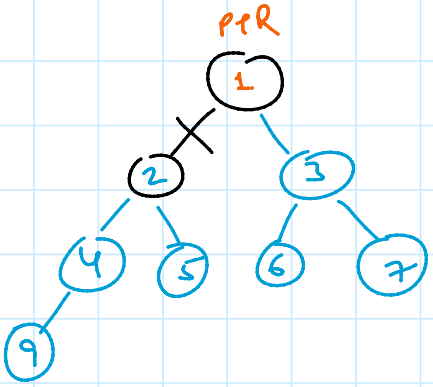
del - 3



element | 3 |

last | 1 |

del - 2

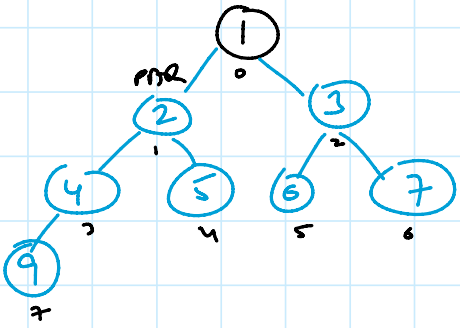


element | 2 |

last | 1 |

arr

1	2	3	4	5	6	7	9



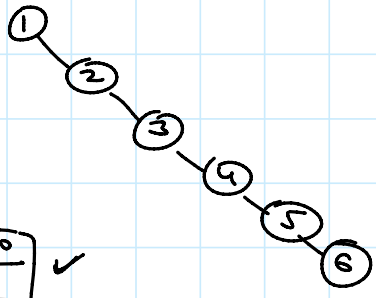
left child = (PTR * 2) + 1

Right child = left child + 1

```cpp
#include<iostream>
using namespace std;
void addHeap(int tree[], int n, int element)
{
    int ptr=n;
    while(ptr!=0)
    {
        int par=(ptr-1)/2;
        if(tree[par]>element)
        {
            tree[ptr]=element;
            return;
        }
        tree[ptr]=tree[par];
        ptr=par;
    }
    tree[ptr]=element;
}
int delHeap(int tree[], int n)
{
    int ptr=0;
    int element=tree[ptr];
    int last=tree[n];
    n--;
    int left=1,right=2;
    while(right<=n)
    {
        if(last > tree[left] and last > tree[right])
        {
            tree[ptr]=last;
            return element;
        }
        if(tree[left]>tree[right])
        {
            tree[ptr]=tree[left];
            ptr=left;
        }
        else
        {
            tree[ptr]=tree[right];
            ptr=right;
        }
        left=ptr*2+1;
        right = left+1;
    }
    if(left==n and tree[left]>last)
    {
        tree[ptr]=tree[left];
        ptr=left;
    }
    tree[ptr]=last;
    return element;
}
void HeapSort(int arr[], int n)
```

```
{
    for(int i=1;i<n;i++)
        addHeap(arr,i,arr[i]);
    for(int i=n-1;i>0;i--)
    {
        int element = delHeap(arr,i);
        arr[i]=element;
    }
}
int main()
{
    int arr[]={3,5,1,2,6,4,7,9};
    int n=sizeof(arr)/sizeof(int);
    HeapSort(arr,n);
    for(int i:arr)
        cout<<i<<" ";
}
```

BST → unbalanced BST

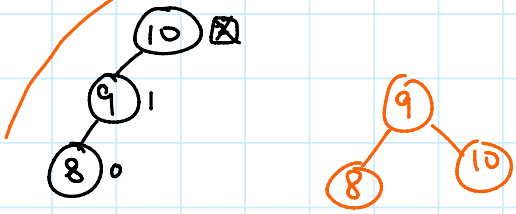


AVL Tree :- (Balanced BST)

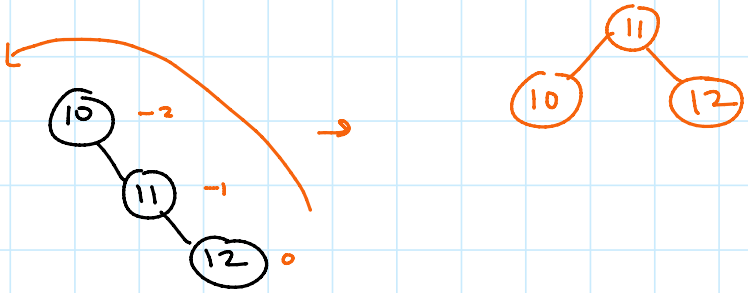
Balance factor = 0, 1, -1 ✓

height of left child - height of right child

LL Problem :- (Right Rotation)



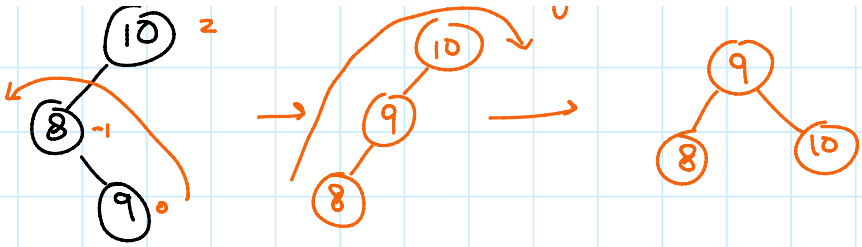
RR Problem → (Left Rotate)



LR Problem

① child → (Left Rotate)

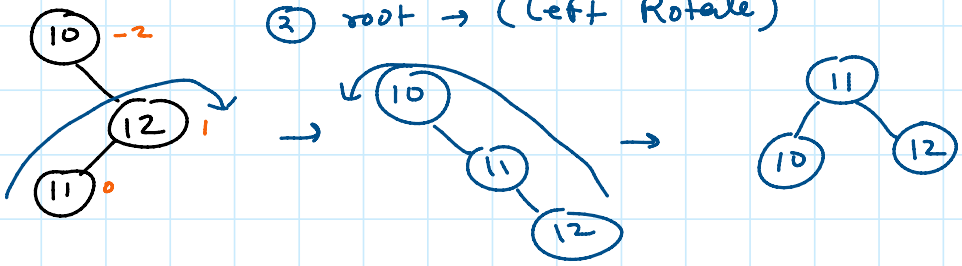
② root → (Right Rotate)



RL problem :→

① child → (Right Rotate)

② root → (Left Rotate)

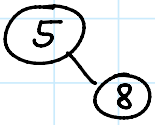


Draw AVL Tree

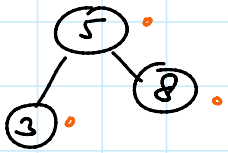
5 8 3 7 2 1 9 6

add - 5

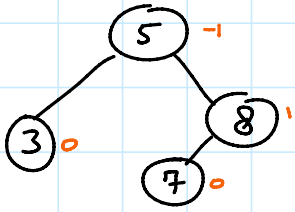
add - 8



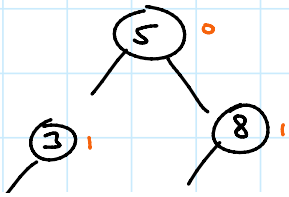
add - 3

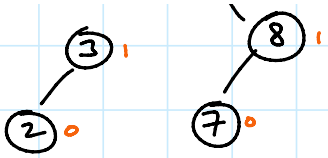


add - 7

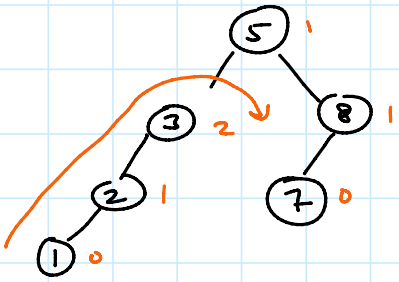


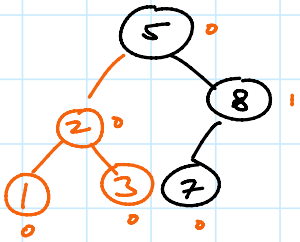
add - 2



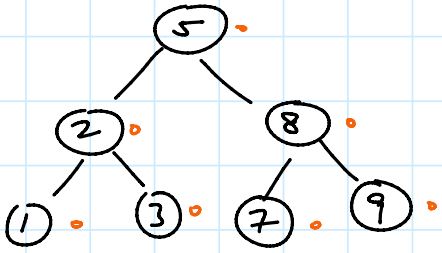


add - 1

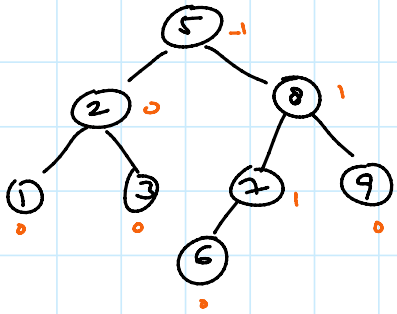




add - 9

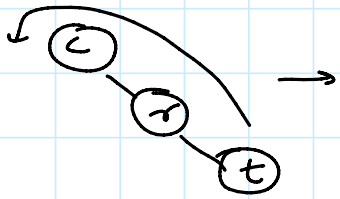


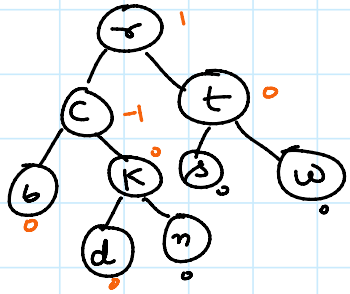
add - 6

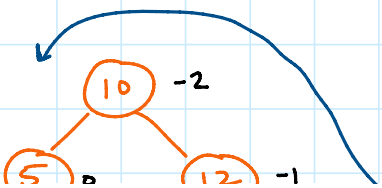


Draw AVL

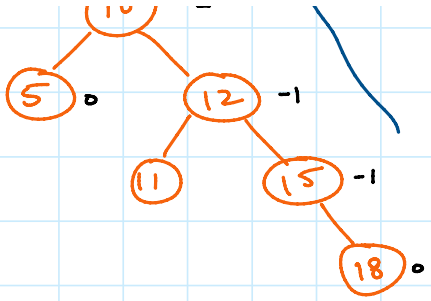
c r t w b k d s n

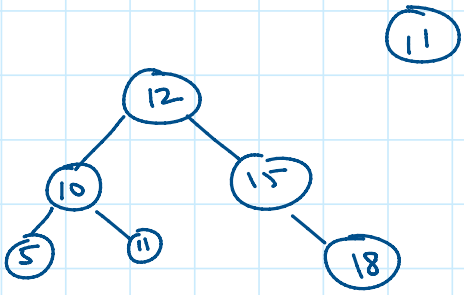






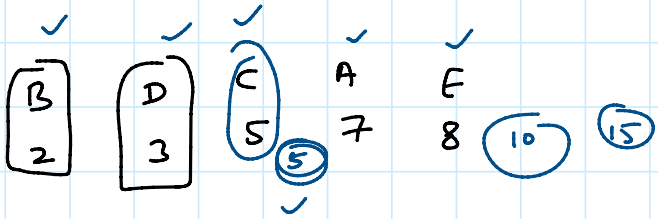
11

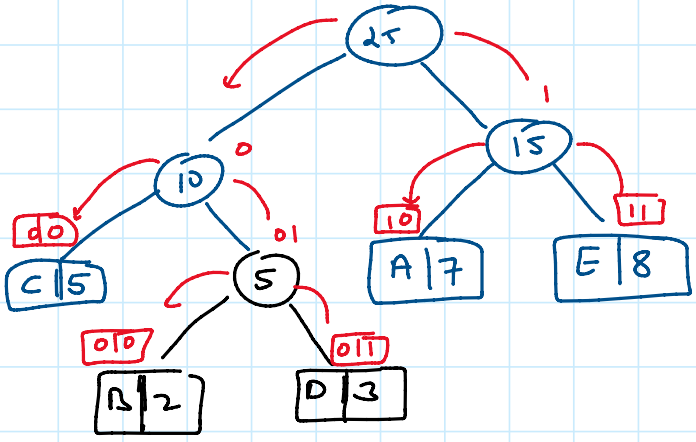




Huffman tree :-

char	A	B	C	D	E
frequency	7	2	5	3	8





char 9:2

00	0
10	1
01	2