Assignment No 1(E)

Title:

      Read the marks obtained by students of second year in an online examination of particular subject. Find out maximum and minimum marks obtained in a that subject. Use heap data structure. Analyze the algorithm.

Objectives:

1. To understand concept of heap

2. To understand concept & features like max heap, min heap.

Learning Objectives:

    ✓ To understand concept of heap

    ✓ To understand concept & features like max heap, min heap.

Learning Outcome:

- Define class for heap using Object Oriented features.

- Analyze working of functions.

Software Required: g++ / gcc compiler- / 64 bit Fedora, eclipse IDE
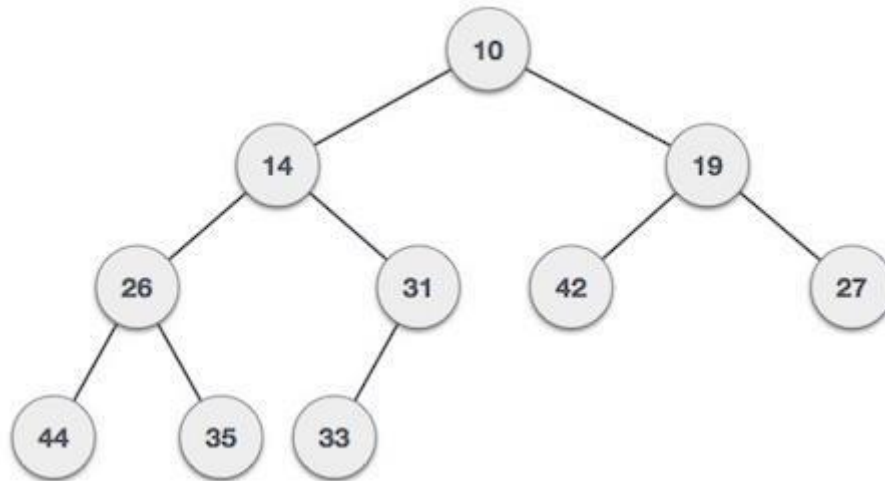
Theory:

Theory:
      Heap is a special case of balanced binary tree data structure where the root-node key is compared with its children and arranged accordingly. If **α** has child node **β** then −
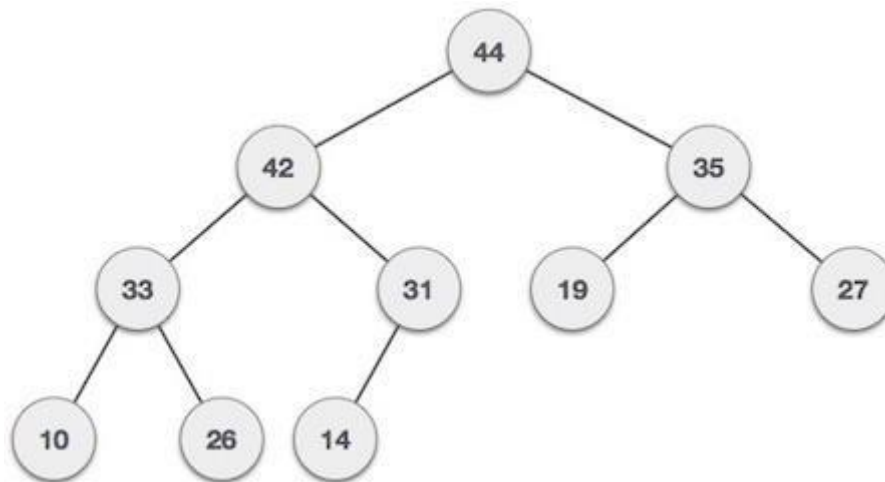$$\text{key}(\alpha) \geq \text{key}(\beta)$$
      As the value of parent is greater than that of child, this property generates **Max Heap**. Based on this criteria, a heap can be of two types −

For Input → 35 33 42 10 14 19 27 44 26 31

Min-Heap − Where the value of the root node is less than or equal to either of its children.

Max-Heap − Where the value of the root node is greater than or equal to either of its children.

Both trees are constructed using the same input and order of arrival.

Max Heap Construction Algorithm

We shall use the same example to demonstrate how a Max Heap is created. The procedure to create Min Heap is similar but we go for min values instead of max values.

We are going to derive an algorithm for max heap by inserting one element at a time. At any point of time, heap must maintain its property. While insertion, we also assume that we are inserting a node in an already heapified tree.

Step 1 − Create a new node at the end of heap.
Step 2 − Assign new value to the node.
Step 3 − Compare the value of this child node with its parent.
Step 4 − If value of parent is less than child, then swap them.

Step 5 − Repeat step 3 & 4 until Heap property holds.

Note − In Min Heap construction algorithm, we expect the value of the parent node to be less than that of the child node.
Let's understand Max Heap construction by an animated illustration. We consider the same input sample that we used earlier.

INPUT:35,33,42,10,14,19,27,44,16,31

Max Heap Deletion Algorithm
Let us derive an algorithm to delete from max heap. Deletion in Max (or Min) Heap always happens at the root to remove the Maximum (or minimum) value.
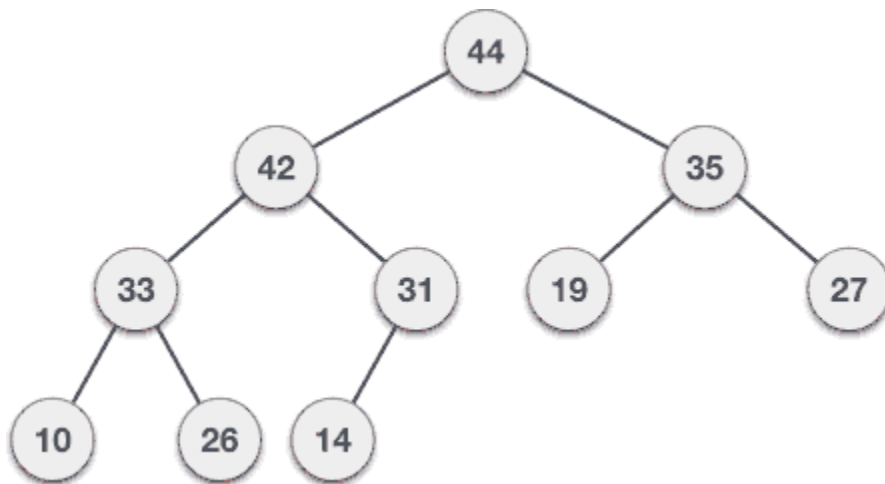**Step 1** − Remove root node.
Step 2 − Move the last element of last level to root.
Step 3 − Compare the value of this child node with its parent.
Step 4 − If value of parent is less than child, then swap them.
Step 5 − Repeat step 3 & 4 until Heap property holds.

Conclusion: This program gives us the knowledge of heap and its types

Program:

```cpp
#include <iostream>
using namespace std;
class heap1{
public:

void MAX_HEAPIFY(int a[], int i, int n)
{
    int l,r,largest,loc;
    l=2*i;
    r=(2*i+1);
    if((l<=n)&&a[l]>a[i])
        largest=l;
    else
        largest=i;

    if((r<=n)&&(a[r]>a[largest]))
        largest=r;
    if(largest!=i)
    {
        loc=a[i];
        a[i]=a[largest];
        a[largest]=loc;
        MAX_HEAPIFY(a, largest,n);
    }
}
void BUILD_MAX_HEAP(int a[], int n)
{
    for(int k = n/2; k >= 1; k--)
    {
        MAX_HEAPIFY(a, k, n);
    }
}
void HEAPSORT(int a[], int n)
{

    BUILD_MAX_HEAP(a,n);
    int i, temp;
    for (i = n; i >= 2; i--)
    {
        temp = a[i];
        a[i] = a[1];
        a[1] = temp;
```

```cpp
        MAX_HEAPIFY(a, 1, i - 1);
    }
}

void accept(){
        int n;
        cout<<"Enter the number of students"<<endl;
        cin>>n;
        int a[n];
        cout<<"Enter the marks of the students "<<endl;
        for (int i = 1; i <= n; i++)
        {
            cin>>a[i];
        }
        HEAPSORT(a, n);
        display(a,n);
}
void display(int a[],int n){


    cout<<":::::::SORTED MARKS::::::"<<endl;

        for (int i = 1; i <= n; i++)
        {
            cout<<a[i]<<endl;
        }

        cout<<"Minimum marks obtained are:"<<a[1];
        cout<<"\nMaximum marks obtained are:"<<a[n];

}
};
int main()
{
        heap1 h;
        h.accept();
        return 0;
}
```