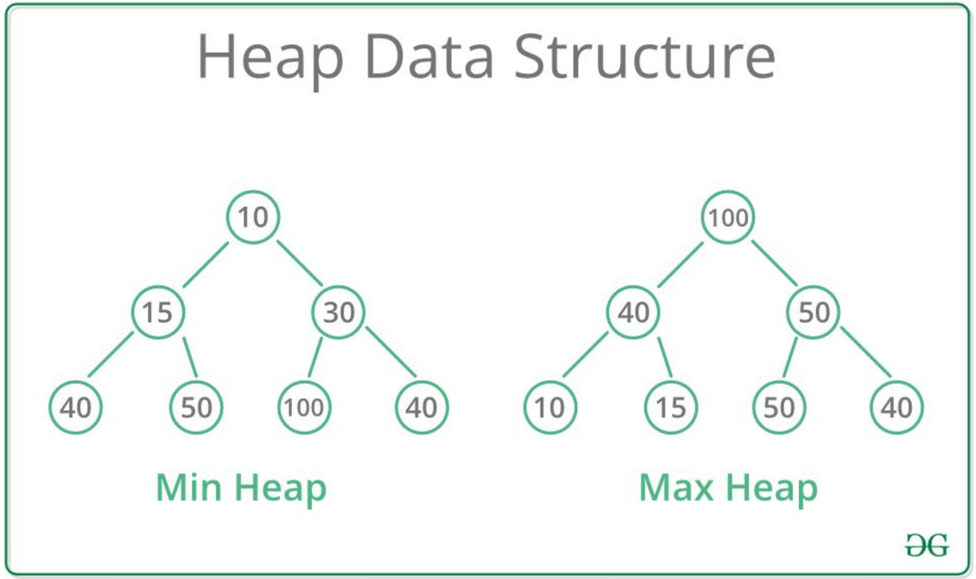
Heaps
Session 27

What is a heap

A Heap is a special Tree-based data structure in which the tree is a complete binary tree. This property of Binary Heap makes them suitable to be stored in an array. (Level Order)



Heaps in Array : Introduction

- Arr[0] : Root element.
- If i th index node
- Arr[(i-1)/2] : Returns the parent node
- Arr[(2*i)+1] :Returns the left child node
- Arr[(2*i)+2] : Returns the right child node

Types of heap

Max heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it’s children. The same property must be recursively true for all sub-trees in that binary tree.

Min heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it’s children. The same property must be recursively true for all sub-trees in that binary tree.

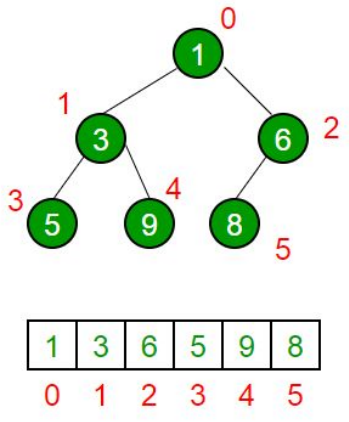
Important operations for heap data structure

- **Heapify :** Process of creating heap from an array
- **Insertion :** Process to insert an element in existing heap
- **Deletion :** Deleting the max / min element in heap and then reorganisation of heap so that next max / min element comes at top

Heapify

- Heaps can be implemented using custom defines class as well as array, since implementation using primary data structure (array) can be done we prefer to implement it that way.
- To represent the binary tree as array we do the following:

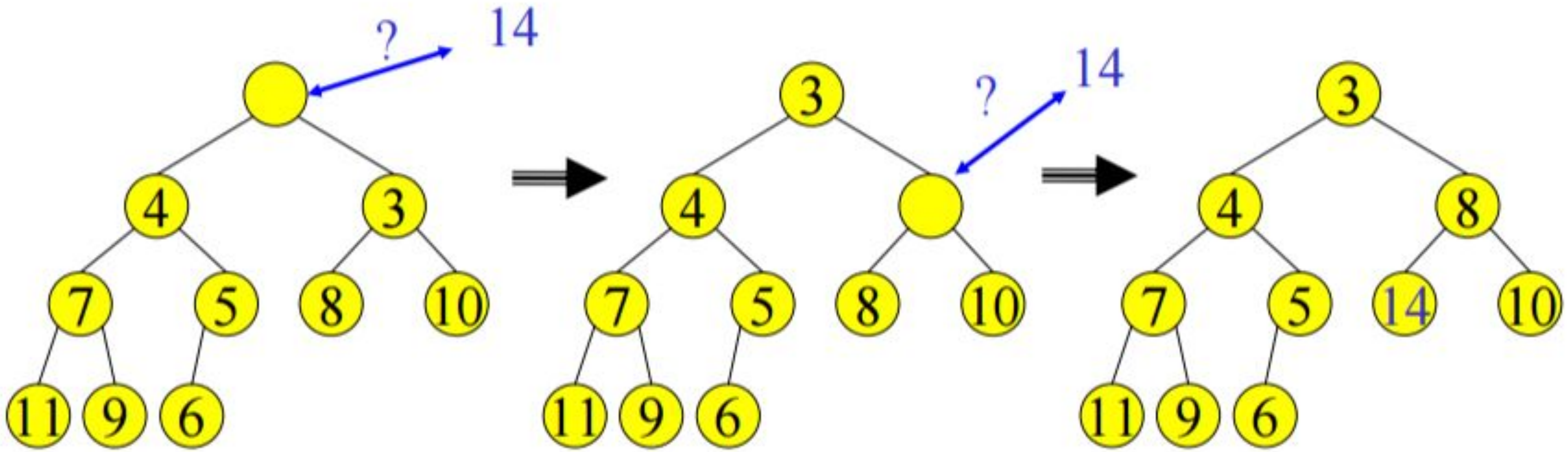
0 based indexing,
child of index i are 2i + 1 and 2i + 2,
parent of i is (i - 1) / 2



Percolate Down

- We do the following step under percolate down algo (if max heap):
    - Pick the max of value of both the child if one child is there pick it’s value
    - Compare this with the value of current node where you are
    - If the value of child is more swap parent with child
    - Continue doing the same

- We do the following step under percolate down algo (if min heap):
    - Pick the min of value of both the child if one child is there pick it’s value
    - Compare this with the value of current node where you are
    - If the value of child is less swap parent with child
    - Continue doing the same.



?
14
4
3
7
5
8
10
11
9
6
3
?
14
4
7
5
8
10
11
9
6
3
4
8
7
5
14
10
11
9
6

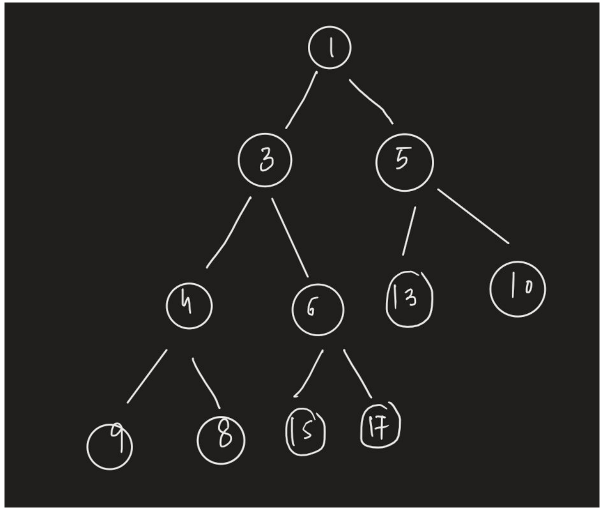
Max heapify an array

Let's say we want to make a max heap

- Start at the last index make sure subtree at that index is max heap
- Move to the previous element
- Recursively apply this step

Example:

11
1 3 5 4 6 13 10 9 8 15 17
https://www.codingninjas.com/studio/problems/build-heap_975375



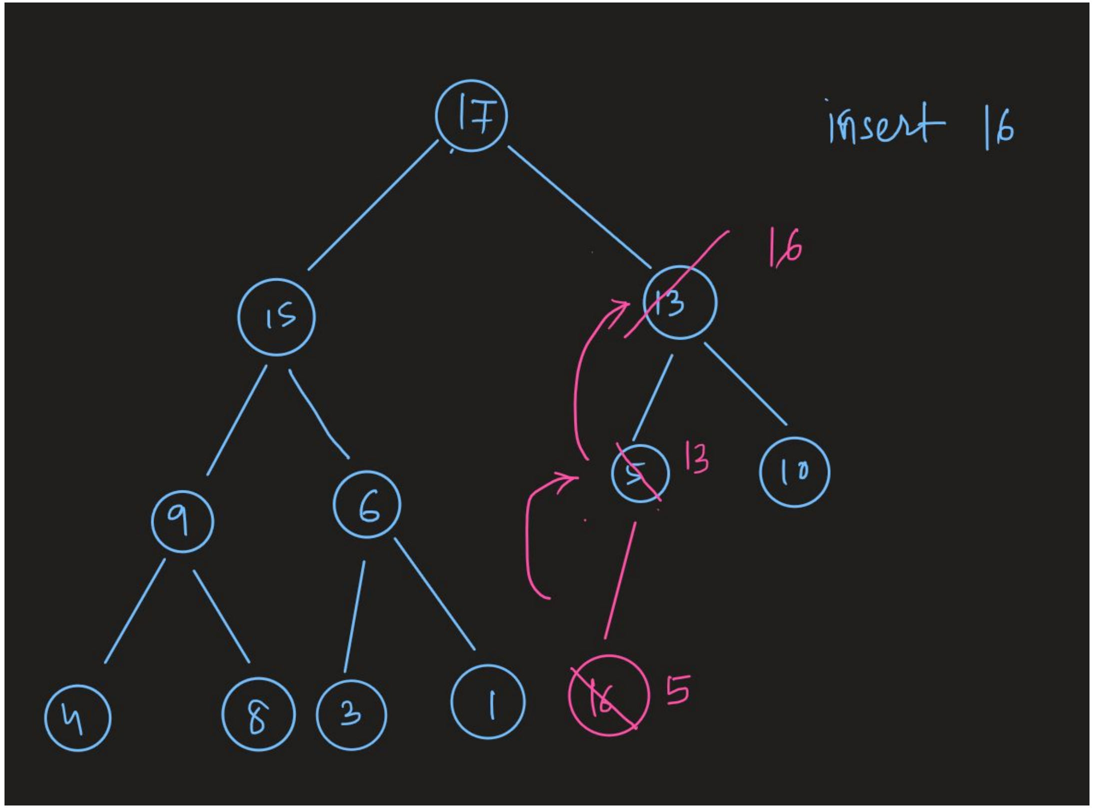
```cpp
#include <bits/stdc++.h>
void help(int i, vector<int> &arr, int n){
    int large = i;
    int left = 2*i+1;
    int right = 2*i+2;
    if(left<n&&arr[left]>arr[large]){
        large = left;
    }
    if(right<n&&arr[right]>arr[large]){
        large = right;
    }
    if(large!=i){
        swap(arr[i],arr[large]);
        help(large,arr,n);
    }
}
vector<int> buildHeap(vector<int> arr, int n)
{
    // Write your code here
    for(int i=n-1;i>=0;i--){
        help(i,arr,n);
    }
    return arr;
}
```

Percolate Up

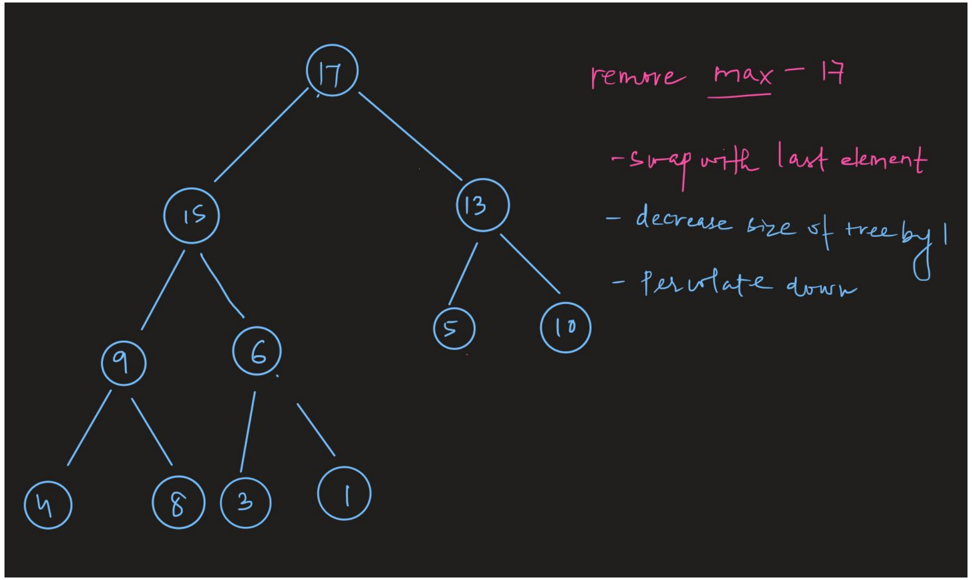
- We do the following step under percolate up algo (max heap):
    - Pick the value of parent
    - Compare this with the value of current node where you are
    - If the value of parent is less swap parent with child
    - Continue doing the same

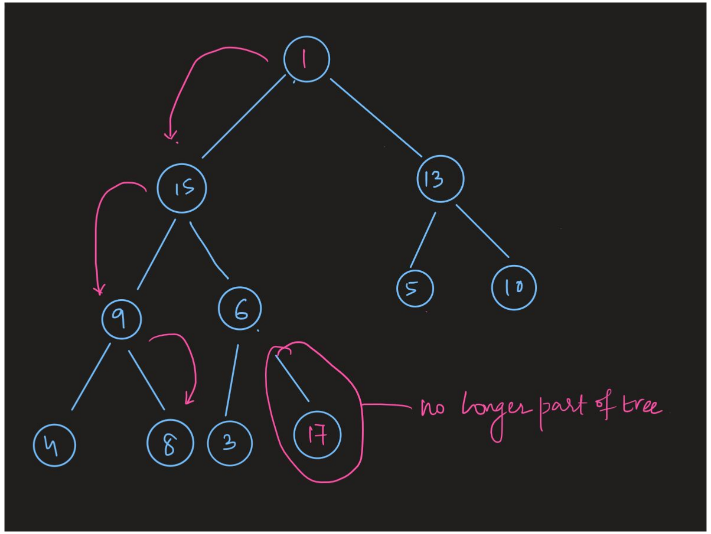
- We do the following step under percolate up algo (min heap):
    - Pick the value of parent
    - Compare this with the value of current node where you are
    - If the value of parent is more swap parent with child
    - Continue doing the same

Addition of a value to heap



Removal of a value from heap





C++ stl equivalent of heap

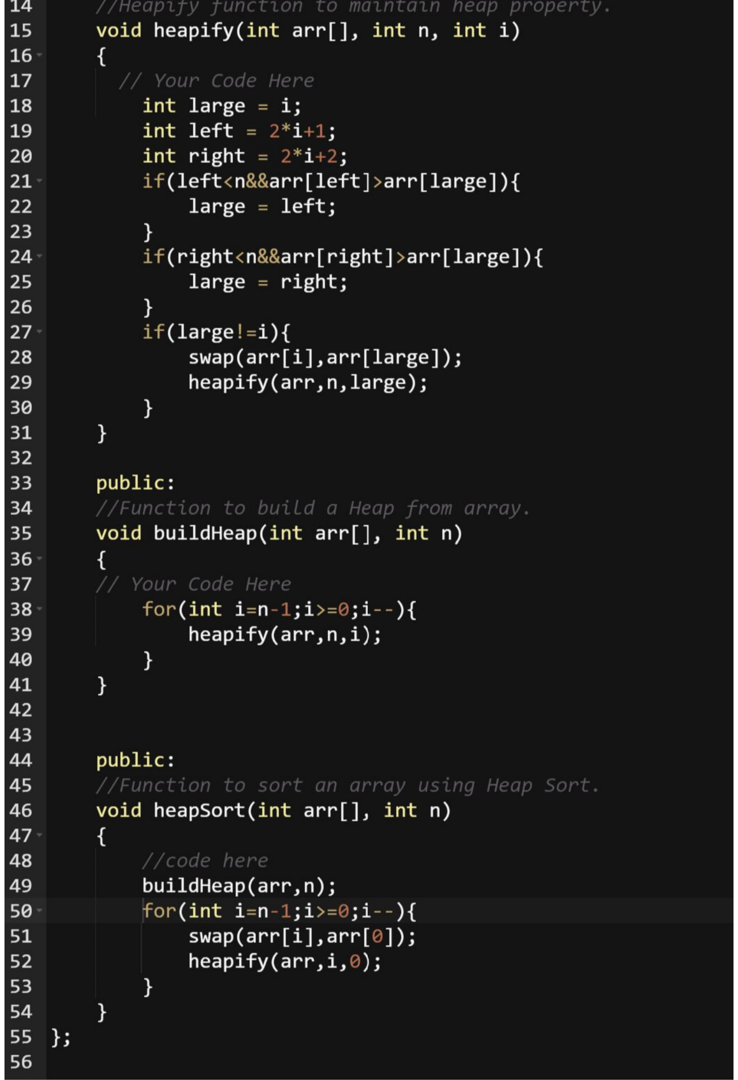
```
std::priority_queue<T, vector<T>, Compare>


class Compare {
    public:
       bool operator()(T a, T b){
           if(cond){
               return true;
           }
           return false;
      }
}
```

Time complexity

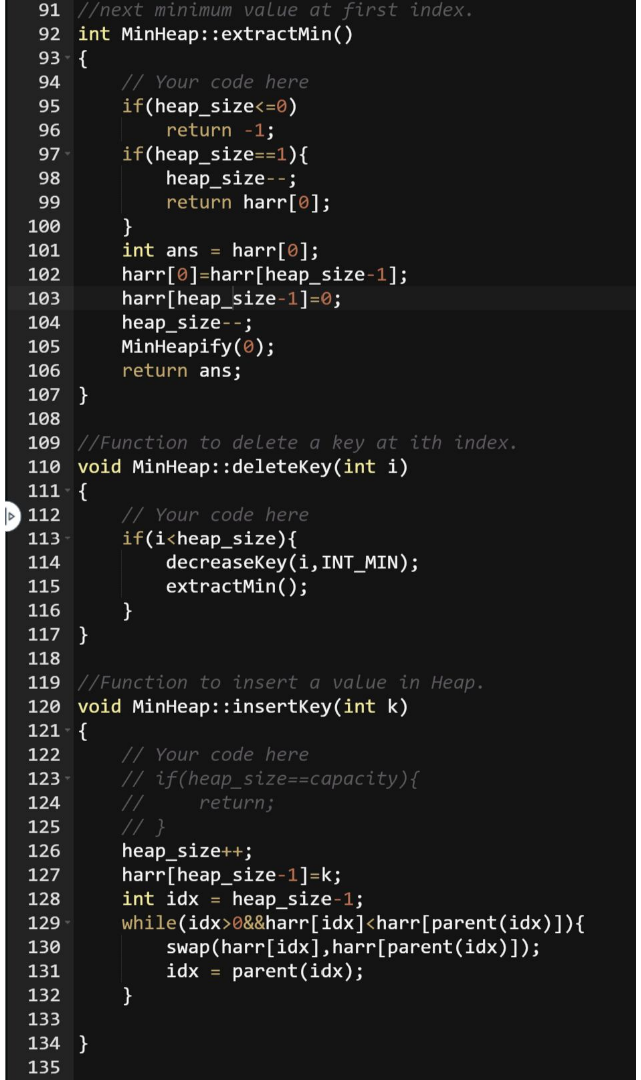
- Heapify : O(N log N)
- Insertion of element : O(log N)
- Deletion of element (min / max) : O(log N)

Basic Operation On Binary Heap

- Heap sort : https://practice.geeksforgeeks.org/problems/heap-sort/1/?page=1&difficulty[]=1&status[]=solved&category[]=Heap&sortBy=submissions
- Insertion and Deletion. https://practice.geeksforgeeks.org/problems/operations-on-binary-min-heap/1



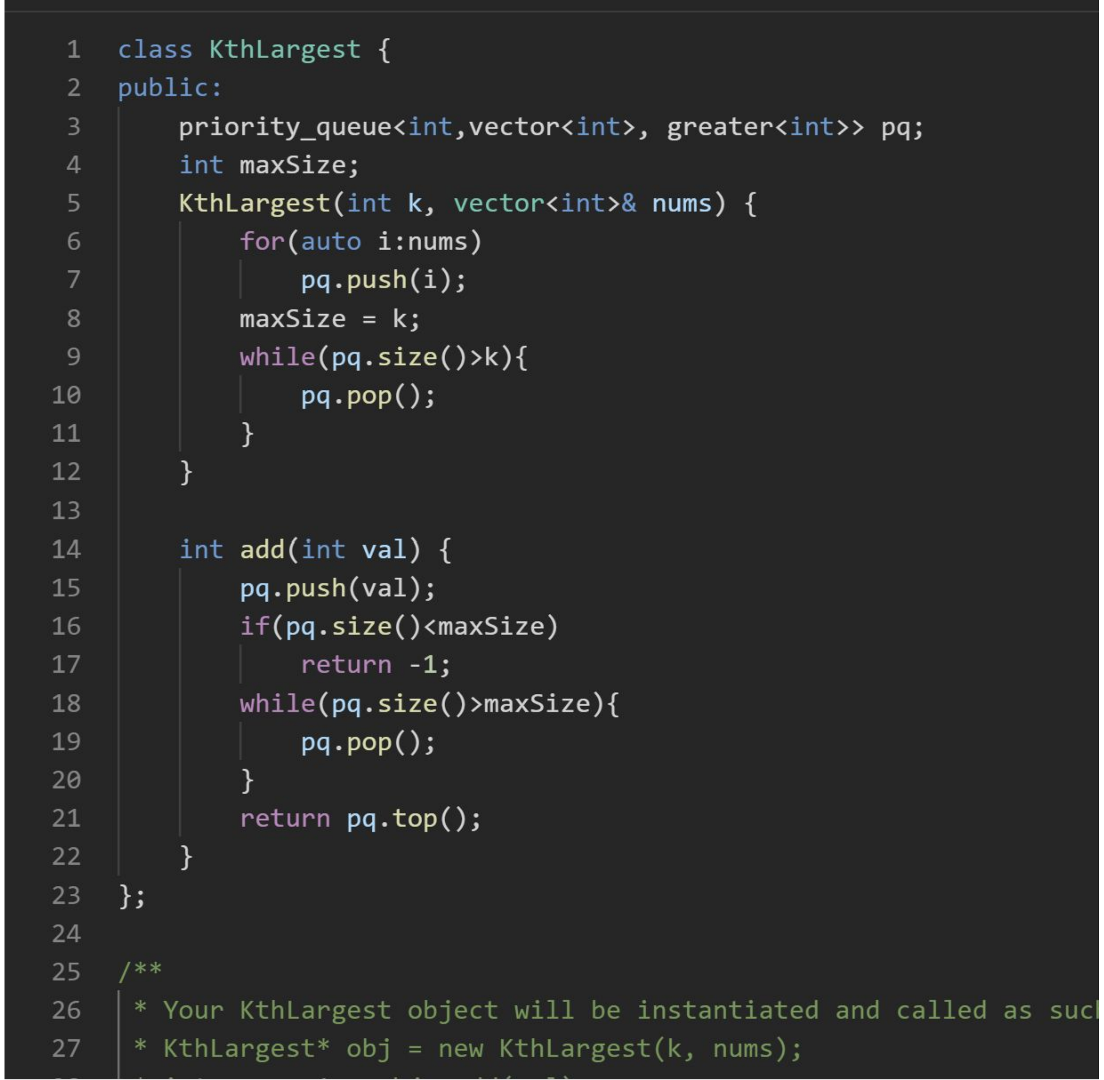
```
    //Heapify function to maintain heap property.
    void heapify(int arr[], int n, int i)
    {
      // Your Code Here
        int large = i;
        int left = 2*i+1;
        int right = 2*i+2;
        if(left<n&&arr[left]>arr[large]){
            large = left;
        }
        if(right<n&&arr[right]>arr[large]){
            large = right;
        }
        if(large!=i){
            swap(arr[i],arr[large]);
            heapify(arr,n,large);
        }
    }

    public:
    //Function to build a Heap from array.
    void buildHeap(int arr[], int n)
    {
    // Your Code Here
        for(int i=n-1;i>=0;i--){
            heapify(arr,n,i);
        }
    }


    public:
    //Function to sort an array using Heap Sort.
    void heapSort(int arr[], int n)
    {
        //code here
        buildHeap(arr,n);
        for(int i=n-1;i>=0;i--){
            swap(arr[i],arr[0]);
            heapify(arr,i,0);
        }
    }
};
```



```cpp
//next minimum value at first index.
int MinHeap::extractMin()
{
    // Your code here
    if(heap_size<=0)
        return -1;
    if(heap_size==1){
        heap_size--;
        return harr[0];
    }
    int ans = harr[0];
    harr[0]=harr[heap_size-1];
    harr[heap_size-1]=0;
    heap_size--;
    MinHeapify(0);
    return ans;
}

//Function to delete a key at ith index.
void MinHeap::deleteKey(int i)
{
    // Your code here
    if(i<heap_size){
        decreaseKey(i,INT_MIN);
        extractMin();
    }
}

//Function to insert a value in Heap.
void MinHeap::insertKey(int k)
{
    // Your code here
    // if(heap_size==capacity){
    //     return;
    // }
    heap_size++;
    harr[heap_size-1]=k;
    int idx = heap_size-1;
    while(idx>0&&harr[idx]<harr[parent(idx)]){
        swap(harr[idx],harr[parent(idx)]);
        idx = parent(idx);
    }

}

```

Question

https://leetcode.com/problems/kth-largest-element-in-a-stream/description/



```cpp
class KthLargest {
public:
    priority_queue<int,vector<int>, greater<int>> pq;
    int maxSize;
    KthLargest(int k, vector<int>& nums) {
        for(auto i:nums)
            pq.push(i);
        maxSize = k;
        while(pq.size()>k){
            pq.pop();
        }
    }

    int add(int val) {
        pq.push(val);
        if(pq.size()<maxSize)
            return -1;
        while(pq.size()>maxSize){
            pq.pop();
        }
        return pq.top();
    }
};

/**
 * Your KthLargest object will be instantiated and called as suc
 * KthLargest* obj = new KthLargest(k, nums);
```