**Min Heap**

#include <bits/stdc++.h>

void minHeapify(int index, vector<int>& heap) {

int leftChild = 2 \* index + 1;

int rightChild = 2 \* index + 2;

int smallest = index;

int size = heap.size();

if (leftChild < size && heap[leftChild] < heap[index]) {

smallest = leftChild;

}

if (rightChild < size && heap[rightChild] < heap[smallest]) {

smallest = rightChild;

}

if (smallest != index) {

swap(heap[smallest], heap[index]);

minHeapify(smallest, heap);

}

}

void insertElement(vector<int>& heap, int element) {

heap.push\_back(element);

int index = heap.size() - 1;

int parent = (index - 1) / 2;

while (index > 0 && heap[parent] > heap[index]) {

minHeapify(parent, heap);

index = parent;

parent = (parent - 1) / 2;

}

}

int removeMin(vector<int>& heap) {

int minElement = heap[0];

swap(heap[0], heap[heap.size() - 1]);

heap.pop\_back();

minHeapify(0, heap);

return minElement;

}

vector<int> minHeap(int n, vector<vector<int>>& q) {

vector<int> heap;

vector<int> result;

for (int i = 0; i < q.size(); i++) {

if (q[i][0] == 0) {

insertElement(heap, q[i][1]);

} else {

result.push\_back(removeMin(heap));

}

}

return result;

}