**ADVANCED DATA STRUCTURE**

**GROUP E**

**ASSIGNMENT 10**

**YEAR: 2017-18**

**COLLEGE: VIIT**

**Date Of Completion : 30/03/2018**

**Title:**

Heap Data Structure

**Problem Statement:**

Read the marks obtained by students of second year in an online examination of particular

subject. Find out maximum and minimum marks obtained in that subject. Use heap data

structure. Analyze the algorithm.

**Objective:**

To find maximum and minimum value of given database using heap data structure

**Software And Hardware Requirement:**

1. 64-bit Open source Linux or its derivative.

2. Open Source C++ Programming tool like G++/GCC.

**Theory:**

A Binary Heap is a Binary Tree with following properties.

1. Its a complete tree (All levels are completely filled except possibly the last level and

the last level has all keys as left as possible). This property of Binary Heap makes

them suitable to be stored in an array.

2. A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at

root must be minimum among all keys present in Binary Heap. The same property

must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to

Min Heap.

1How is Binary Heap represented?

A Binary Heap is a Complete Binary Tree. A binary heap is typically represented as array.

Applications of Heaps:

1. Heap Sort: Heap Sort uses Binary Heap to sort an array in O(nLogn) time.

2. Priority Queue: Priority queues can be efficiently implemented using Binary Heap

because it supports insert(), delete() and extractmax(), decreaseKey() operations in

O(logn) time. Binomoial Heap and Fibonacci Heap are variations of Binary Heap.

These variations perform union also efficiently.

3. Graph Algorithms: The priority queues are especially used in Graph Algorithms like

Dijkstras Shortest Path and Prims Minimum Spanning Tree.

4. Many problems can be efficiently solved using Heaps. See following for example.

(a) Kth Largest Element in an array.

(b) Sort an almost sorted array.

(c) Merge K Sorted Arrays.

Operations on Min Heap:

1. getMini(): It returns the root element of Min Heap. Time Complexity of this operation

is O(1).

2. extractMin(): Removes the minimum element from Min Heap. Time Complexity of

this Operation is O(Logn) as this operation needs to maintain the heap property (by

calling heapify()) after removing root.

3. decreaseKey(): Decreases value of key. Time complexity of this operation is O(Logn).

If the decreases key value of a node is greater than parent of the node, then we dont

need to do anything. Otherwise, we need to traverse up to fix the violated heap

property.

4. insert(): Inserting a new key takes O(Logn) time. We add a new key at the end of

the tree. IF new key is greater than its parent, then we dont need to do anything.

Otherwise, we need to traverse up to fix the violated heap property.

5. delete(): Deleting a key also takes O(Logn) time. We replace the key to be deleted

with minum infinite by calling decreaseKey(). After decreaseKey(), the minus infinite

value must reach root, so we call extractMin() to remove key

**Algorithm:**

Insertion Operation in Max Heap

Step 1: Insert the new node as last leaf from left to right.

Step 2: Compare new node value with its Parent node.

2Step 3: If newNode value is greater than its parent, then swap both of them.

Step 4: Repeat step 2 and step 3 until newNode value is less than its parent nede (or) n

The root of the max heap will give maximum marks

Insertion Operation in Min Heap

Step 1: Insert the newNode as last leaf from left to right.

Step 2: Compare newNode value with its Parent node.

Step 3: If newNode value is less than its parent, then swap both of them.

Step 4: Repeat step 2 and step 3 until newNode value is greater than its parent nede (or

The root of the min heap will give minimum marks

**Code :**

/\*

\* MinMaxHeap.cpp

\*

\* Created on: Mar 23, 2017

\*

Author: student

\*/

#include<iostream>

using namespace std;

class Heap

{

private:

int total,marks[20];

public:

Heap();

void Input();

int Max\_Child(int a,int b);

int Min\_Child(int a,int b);

void Max\_Heap();

void Min\_Heap();

};

Heap::Heap()

{

int i;

for (i = 1; i <20; i++) {

marks[i]=0;

}

}

void Heap::Input()

{

int i;

3cout<<"\nEnter Total No.of Students : ";

cin>>total;

marks[0]=total;

cout<<"\nEnter Marks for Each Student : ";

for (i = 1; i <= total; i++) {

cin>>marks[i];

}

}

int Heap::Max\_Child(int l,int r)

{

if(marks[l] >= marks[r] && r<=total)

return l;

else if(marks[l] <= marks[r] && r<=total)

return r;

else

return l;

}

int Heap::Min\_Child(int l,int r)

{

if(marks[l] <= marks[r] && r<=total)

return l;

else if(marks[l] >= marks[r] && r<=total)

return r;

else

return l;

}

void Heap::Max\_Heap()

{

int temp,P,max\_loc,max;

P=total/2;

while(P>=1)

{

max\_loc=Max\_Child(2\*P,((2\*P)+1));

max=marks[max\_loc];

if(marks[P] < max)

{

temp=marks[P];

marks[P]=max;

marks[max\_loc]=temp;

}

P--;

}

cout<<"\n Max Mark : "<<marks[1];

4}

void Heap::Min\_Heap()

{

int temp,P,min\_loc,min;

P=total/2;

while(P>=1)

{

min\_loc=Min\_Child(2\*P,((2\*P)+1));

min=marks[min\_loc];

if(marks[P] > min)

{

temp=marks[P];

marks[P]=min;

marks[min\_loc]=temp;

}

P--;

}

cout<<"\n Min Mark : "<<marks[1];

}

int main()

{

Heap obj;

obj.Input();

obj.Max\_Heap();

obj.Min\_Heap();

return 0;

}

**Output :**

Enter Total No.of Students : 9

Enter Marks for Each Student : 45

56

21

10

78

89

56

100

99

Max Mark : 100

5Min Mark : 10 Press any key to continue . . .

**Conclusion :**

After completing this assignment we were able to build minheap and maxheap and perform

operations like insertion and deletion on them.