Laboratory

4

BINARY HEAP

A **binary heap** is a complete **binary** tree which satisfies the **heap** ordering property.

Objective

A binary heap is a complete binary tree which satisfies the heap ordering property. The ordering can be one of two types:

***min-heap property*:** the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.

***max-heap property*:** the value of each node is less than or equal to the value of its parent, with the maximum-value element at the root.

Overview

"heap" will always refer to a min-heap.

In a heap the highest (or lowest) priority element is always stored at the root, hence the name "heap". A heap is not a sorted structure and can be regarded as partially ordered. As you see from the picture, there is no particular relationship among nodes on any given level, even among the siblings. Since a heap is a complete binary tree, it has a smallest possible height - a heap with N nodes always has O(log N) height.

A heap is useful data structure when you need to remove the object with the highest (or lowest) priority. A common use of a heap is to implement a priority queue.

Procedure

Aim: To implement operations on binary heap.

PROGRAM:

```
#include <stdio.h>
#include<stdlib.h>
#define max 15
int heap[max],h=0;
void insert();
int deletemin();
void display();
void reheap_up(int heap[],int x);
void reheap_down(int heap[],int);
void main()
{
```

```
int ch,x;
while(1)
{
printf("\n1.insert\n2.deletemin\n3.display\n4.exit\n");
printf("enter ur choice:");
scanf("%d",&ch);
switch(ch)
{
  case 1:
    insert();
    break;
  case 2:
    x=deletemin();
    if(x!=-1)
    printf(" the deleted element is %d",x);
    else
     printf("\ncan not be deleted");
    break;
  case 3:
          display();
          break;
  case 4:
    exit(0);
  default:printf("\nInvalid choice");
    }
 }
}
void insert()
{
   int i,x;
   if(h>max)
     printf("\nheap is full");
   else
   {
      h++;/*inc. size of the heap*/
      printf("\nenter element to be insert:");
      scanf("%d",&x);
    heap[h]=x;/*x is the last element in the heap*/
    reheap_up(heap,h);/*arranging as minheap*/
    }
}

void reheap_up(int heap[],int n)
{
 int x,p;
 p=n/2;
   if(n!=1)/*if n not a root*/
 if(heap[p]>heap[n])/*minheap property does not satisfy*/
```

```
 {
            x=heap[n];
         heap[n]=heap[p]; /*swapping child,i and parent,i/2*/
            heap[p]=x;
            reheap_up(heap,p);/*recursively call until at a parent min-heap property satisfies*/
       }
}

int deletemin()
{
   int last,min;
   if(h==0)
  {
      printf("heap is empty");
   return -1;
  }
   else
   {
     min=heap[1];/*root is the min element*/
     last=heap[h];/*last element in the heap*/
  h--;/*deleting last element*/
  heap[1]=last;/*last element is the root now*/
  reheap_down(heap,1);/*arranging minheap property*/
    return min;
 }
}

void reheap_down(int heap[],int p)
 {
   int lc,rc,small,x;

   lc=2*p;/*left child*/
   rc=2*p+1;/*right child*/
     if(lc<=h)/*if there is atleast one child*/
          {
             small=lc;/*small is left child*/
             if(rc<=h && heap[lc]>heap[rc])
               small=rc;/*small is right child*/

             if(heap[p]>heap[small])/*parent is greater than a child*/
             {
                      x=heap[small];  /*swapping parent and small child*/
                      heap[small]=heap[p];
                      heap[p]=x;
                      reheap_down(heap,small);/*recursively call until leaf encounters*/
                    }
    }
 }
```

```
void display()
{
  int i;
  if(h==0)
    printf("\nheap is empty");
  else
    for(i=1;i<=h;i++)
      printf("%d\t",heap[i]);
}
```

OUTPUT

```
student@teacher-virtual-machine:~$ vi 4.c
student@teacher-virtual-machine:~$ cc 4.c
student@teacher-virtual-machine:~$ ./a.out
1.insert
2.deletemin
3.display
4.exit
enter ur choice:1
enter element to be insert:29
1.insert
2.deletemin
3.display
4.exit
enter ur choice:1
enter element to be insert:7
1.insert
2.deletemin
3.display
4.exit

enter ur choice:1
enter element to be insert:6
1.insert
2.deletemin
3.display
4.exit
enter ur choice:1
enter element to be insert:5
1.insert
2.deletemin
3.display
4.exit
enter ur choice:3
5       6       7       29
```

```
1.insert
2.deletemin
3.display
4.exit
enter ur choice:2
 the deleted element is 5
1.insert
2.deletemin
3.display
4.exit
enter ur choice:3
6          29         7
1.insert
2.deletemin
3.display
4.exit
enter ur choice:4
student@teacher-virtual-machine:~$
```

Viva-Voice Questions

1)what is binary heap?
2)What is min heap?
3)what is max heap?
4)How to insert an element in binary heap?
5)How to delete an element from binary heap?

Lab Report

After successful completion of this lab experiment, the student will be able to know about the implementation of binary heap.

T.SUKANYA
Asst.Professor
CSE Department
Teaching Exp: 5 Yrs
Subjects Taught: CD,DAA,FLAT,DS,ADS