DHRUV GARG 16 BCE 1190

Data Structures and Algorithms

LAB 5 – Heap implementation

MAX HEAP

Code for max heap :

```
#include <stdio.h>
void max_heapify(int *a,int i,int n)
{
   int j, temp;
   temp = a[i];
   j = 2 * i;
   while (j <= n)
   {
      if (j < n && a[j+1] > a[j])
         j = j + 1;
      if (temp > a[j])
         break;
      else if (temp <= a[j])
      {
         a[j/2] = a[j];
         j = 2 * j;
      }
   }
   a[j/2] = temp;
   return;
}
void build_maxheap(int *a, int n)
{
   int i;
   for(i = n/2; i >= 1; i--)
   {
      max_heapify(a,i,n);
   }
}
void print_heap(int *a, int n)
{
   int i;
   for (i = 1; i <= n; i++)
   {
      printf(" %d ", a[i]);
   }
   printf("\n");
}
int main()
{
   int n, i, x;
   printf("\n\tMAX HEAP
IMPLEMENTATION\n");
   printf("\nEnter no of elements of array : ");
   scanf("%d", &n);
   int a[15] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
   for (i = 1; i <= n; i++)
   {
      printf("\nEnter element %d : ", i);
      scanf("%d", &a[i]);
      build_maxheap(a, n);
      print_heap(a, n);
   }
   build_maxheap(a, n);
   printf("\nMax Heap : ");
   print_heap(a, n);
}
```

Screenshot for max heap

C:\Users\Dhruv\Documents\C\DSA_maxHeap.exe

```
        MAX HEAP IMPLEMENTATION

Enter no of elements of array : 10

Enter element 1 : 82
 82  0  0  0  0  0  0  0  0  0

Enter element 2 : 100
 100  82  0  0  0  0  0  0  0  0

Enter element 3 : 44
 100  82  44  0  0  0  0  0  0  0

Enter element 4 : 253
 253  100  44  82  0  0  0  0  0  0

Enter element 5 : 90
 253  100  44  82  90  0  0  0  0  0

Enter element 6 : 314
 314  100  253  82  90  44  0  0  0  0

Enter element 7 : 73
 314  100  253  82  90  44  73  0  0  0

Enter element 8 : 54
 314  100  253  82  90  44  73  54  0  0

Enter element 9 : 583
 583  314  253  100  90  44  73  54  82  0

Enter element 10 : 12
 583  314  253  100  90  44  73  54  82  12

Max Heap :  583  314  253  100  90  44  73  54  82  12
```

MIN HEAP

Code for min heap :

```
#include <stdio.h>
void min_heapify(int *a,int i,int n)
{
   int j, temp;
   temp = a[i];
   j = 2 * i;
   while (j <= n)
   {
      if (j < n && a[j+1] < a[j])
         j = j + 1;
      if (temp < a[j])
         break;
      else if (temp >= a[j])
      {
         a[j/2] = a[j];
         j = 2 * j;
      }
   }
   a[j/2] = temp;
  return;
}
void build_minheap(int *a, int n)
{
   int i;
   for(i = n/2; i >= 1; i--)
   {
      min_heapify(a,i,n);
   }
}
void print_heap(int *a, int n)
{
   int i;
   for (i = 1; i <= n; i++)
   {
      printf(" %d ", a[i]);
   }
   printf("\n\n");
}
int main()
{
   int n, i, x;
   printf("\n\tMIN HEAP
IMPLEMENTATION\n");
   printf("\nEnter no of elements of array : ");
   scanf("%d", &n); printf("\n");
   int a[15] =
{999,999,999,999,999,999,999,999,999,999,9
99,999,999,999,999};
   for (i = 1; i <= n; i++)
   {
      printf("Enter element %d : ", i);
      scanf("%d", &a[i]);
      build_minheap(a, n);
      print_heap(a, n);
   }
   build_minheap(a, n);
   printf("\nMin Heap : ");
   print_heap(a, n);
}
```

Screenshot for min heap

C:\Users\Dhruv\Documents\C\DSA_minHeap.exe

```
        MIN HEAP IMPLEMENTATION

Enter no of elements of array : 10

Enter element 1 : 510
 510  999  999  999  999  999  999  999  999  999

Enter element 2 : 120
 120  510  999  999  999  999  999  999  999  999

Enter element 3 : 682
 120  510  682  999  999  999  999  999  999  999

Enter element 4 : 66
 66  120  682  510  999  999  999  999  999  999

Enter element 5 : 326
 66  120  682  510  326  999  999  999  999  999

Enter element 6 : 121
 66  120  121  510  326  682  999  999  999  999

Enter element 7 : 948
 66  120  121  510  326  682  948  999  999  999

Enter element 8 : 73
 66  73  121  120  326  682  948  510  999  999

Enter element 9 : 250
 66  73  121  120  326  682  948  510  250  999

Enter element 10 : 40
 40  66  121  120  73  682  948  510  250  326


Min Heap :  40  66  121  120  73  682  948  510  250  326
```