ACADEMY OF TECHNOLOGY

Lab Assignment 7

Paper name: Design and Analysis of Algorithms Lab

Code: PCC-CS494 | **Semester: 4^{th}**

Discipline: CSE | **Time: 2 Hours**

Date: April 10, 2023

1. Write a program in C or C++ to implement Heap Sort algorithm.

Algorithm 1: **Heap-Adjust(a, i, n)**

// The complete binary trees with roots $2i$ and $2i+1$ are combined with i to form a heap rooted at i, No node has an address greater than n or less than 1

$j := 2 * i$; $key := a[i]$;
while $j \leq n$ **do**
  // compare left and right child, j points to the larger child
  if $j < n$ and $a[j] < a[j+1]$ **then** $j := j + 1$;
  if $key \geq a[j]$ **then break**; // a position for key is found
  $a[\lfloor \frac{j}{2} \rfloor] := a[j]$; $j := 2 * j$;
  // move the larger child up a level
end
$a[\lfloor \frac{j}{2} \rfloor] := key$;

Algorithm 2: **Make-Heap(a, n)**

// Readjust the elements in $A[1:n]$ to form a heap
for $i := \lfloor \frac{n}{2} \rfloor$ *to* 1 *step* -1 **do**
  Heap-Adjust(a, i, n);
end

Algorithm 3: **Heap-Sort(a, n)**

// $a[1:n]$ contains n elements to be sorted. Heap-Sort rearranges them in-place into non-decreasing order.
Make-Heap(a, n); // first transform the elements into a heap
// interchange the new maximum with the element at the end of the tree
for $i := n$ *to* 2 *step* -1 **do**
  $t := a[i]$; $a[i] := a[1]$; $a[1] := t$;
  Heap-Adjust$(a, 1, i-1)$;
end

2. Write a program in C or C++ to implement minimum priority queue using Heap. And perform the following operation.

a) Get-Minimum() to get the minimum element.

b) Extract-Min()to removes the minimum element from Min Heap.

c) Decrease-Key()to decreases value of key.

d) Insert-Key()to add a new key.

e) Delete-Key()to delete a key.

***Algorithm 4:* Get-Minimum()**

```
return A[1];
```

***Algorithm 5:* Extract-Min()**

```
if heapSize ≤ 0 then return ∞;
if heapSize = 1 then
    heapSize := heapSize − 1;
    return A[1];
end
// Store the minimum value, and remove it from heap
min := A[1];
A[1] := A[heapSize];
heapSize := heapSize − 1;
HEAP-ADJUST(A, 1, heapSize);
return min;
```

***Algorithm 6:* Decrease-Key(i, $newVal$)**

```
A[i] := newVal;
while i ≥ 1 and A[PARENT(i)] > A[i] do
    EXCHANGE(A[i], A[PARENT(i)]);
    i := PARENT(i);
end
```

Algorithm 7: **Insert-Key(k)**

```
if heapSize = heapCapacity then
    Write"Overflow: Could not insert Key";
    return;
end
heapSize := heapSize + 1;
A[heapSize] := k; // First insert the new key at the end
i := heapSize;
// Fix the min heap property if it is violated
while i ≥ 1 and A[Parent(i)] > A[i] do
    Exchange(A[i], A[Parent(i)]);
    i := Parent(i);
end
```

Algorithm 8: **Delete-Key(i)**

```
if i > heapSize then
    Write"Delete key is not possible";
    return;
end
Decrease-Key (i, −∞);
Extract-Min ();
```