**Department of Computing**

**CS250: Data Structures and Algorithms**

**Class: BEE-6AB**

# Lab 10 : Binary Heap

**Date: 26th November, 2015**

**Time: 10am-1pm & 2pm-5pm**

# Instructor: Dr. Faisal Shafait

**Lab 10 : Binary Heap / Heap Sort**

**Introduction**

This lab is based on the Binary Heap (min heap).

**Objectives**

The objective of this lab is to implement binary heap (min heap).

**Tools/Software Requirement**

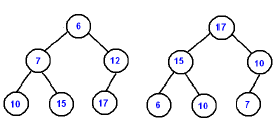
Visual Studio C++

**Description**

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

 the *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.

 the *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.

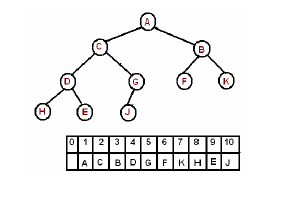


In a heap the highest (or lowest) priority element is always stored at the root, hence the name "heap". A heap is not a sorted structure and can be regarded as partially ordered.

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.

**Array Implementation**

A complete binary tree can be uniquely represented by storing its level order traversal in an array.



The root is the second item in the array. We skip the index zero cell of the array for the convenience of implementation. Consider k-th element of the array,

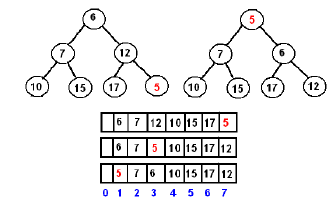
its left child is located at 2\*k index

its right child is located at 2\*k+1. index

its parent is located at k/2 index

**Insert**

The new element is initially appended to the end of the heap (as the last element of the array). The heap property is repaired by comparing the added element with its parent and moving the added element up a level (swapping positions with the parent).



**Delete Min**

The minimum element can be found at the root, which is the first element of the array. We remove the root and replace it with the last element of the heap and then restore the heap property

**Lab Tasks**

Implement a binary heap using an array & Implement the following functions.

**Top** - returns min element without removing it from the heap

**Pop** - Make sure that the tree remains a complete binary tree.

**Push** – Insert a number into the heap and make sure the heap maintains its key property

**isEmpty** – return TRUE if the heap is empty

**size** - returns the number of elements in the heap

**height** - returns the height of the tree

**buildHeap(array)** - converts any array of numbers into a heap.

**print** - prints the tree

**Deliverables**

Students are required to upload the lab on LMS before deadline.

**Note:** Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks. You will submit your workingcodes in **word document** (do **NOT** take screenshot of code, just copy your code and paste it). The name of word document should follow this format. i.e. **YOUR\_NAME\_Lab#**