1 Introduction and Objective

The purpose of this tutorial is to further reinforce the concepts of Binary Heap data structure (Lecture L02, part of Tutorial tut01, PS1, Lab Demo D01) which can be used as ADT Priority Queue. Depending on when you actually read this tutorial 02 question document, you may or may not be able to use the insights on time for PS1 that will due on Saturday, 29 August 2015, 07.59am, before this tutorial is actually conducted.

Then, we will move on to discuss ADT Table (Lecture03, PS2, Lab Demo D02). We will also touch a bit of PS2 Subtask A in this tutorial.

You can use http://visualgo.net/heap.html and http://visualgo.net/bst.html for your reference in this tutorial.
2 Tutorial 02 Questions

More About Binary Heap Data Structure

Q1. What is the minimum and maximum number of comparisons between Binary Heap elements required to construct a Binary (Max) Heap of arbitrary \(n\) elements using the \(O(n)\) \texttt{BuildHeap(array)}? Note that this question has been integrated in VisuAlgo Online Quiz, so it may appear in future Online Quizzes :).

![Figure 1: Now automated :)](image1)

Q2. Give an algorithm to find all vertices bigger than some value \(x\) in a max heap that runs in \(O(k)\) time where \(k\) is the number of vertices in the output. This is a new algorithm analysis type for most of you as the time complexity of the algorithm does not depend on the input size \(n\) but rather the output size \(k\) :O...

Note that this question has also been integrated in VisuAlgo Online Quiz, so it may appear in future Online Quizzes :).

![Figure 2: Also automated :)](image2)
Q3. The second largest element in a max heap with more than two elements (all elements are unique) is always one of the children of the root. Is this true? If yes, show a simple proof. Otherwise, show a counter example.

Note that this kind of (simple) proof will appear in CS2010 written tests, so please refresh your CS1231 knowledge.

**Binary Heap... or Not?**

Q4. We know that Binary (Max) Heap can be used as Priority Queue and can do \texttt{ExtractMax()} in $O(\log n)$ time. What modifications/additions/alterations are required so that both \texttt{ExtractMax()} and \texttt{ExtractMin()} can be done in $O(\log n)$ time for the set of $n$ elements and every other Priority Queue related-operations, especially Insert/Enqueue retains the same $O(\log n)$ running time?

Q5. Follow up from Q4 above: If you can answer Q4 before PS1 is due, will you solve PS1 differently?

**Problem Set 2**

We will end the tutorial with discussion of PS2 subtask A only.