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 $O(n)$ $\text{BuildHeap(array)}$? Note that this question has been integrated in VisuAlgo Online Quiz, so it may appear in future Online Quizzes :).

![Image](image.png)

Figure 1: Now automated :)

Ans: Let’s use an example for $n = 8$

The min and max answers are 7 and 11 respectively.

The case for minimum happens when the array to be converted into a Binary (Max) Heap already satisfies the Max Heap property (e.g. try http://visualgo.net/heap.html?create=8,7,6,5,4,3,2,1). The structure of an 8 node heap is shown below.

![Image](image.png)

Figure 2: Minimum case, $n=8$

In this case except for the last internal node (the red node), which only does 1 comparison, the rest of the internal nodes (green nodes) do exactly 2 comparisons (with it’s left and right child) so the # of comparisons $= 1 + 3 \times 2 = 7$. 


The case for maximum happens where we have to call \texttt{ShiftDown} at each internal node all the way to the deepest leaf (note: contrary to intuition \url{http://visualgo.net/heap.html?create=1,2,3,4,5,6,7,8} does not really produce the maximum number of comparison as 1 will be shifted down to 8 then to 5, try \url{http://visualgo.net/heap.html?create=1,2,3,5,4,6,7,8} where 1 will be shifted down to 8, then 5, then 2.

![Figure 3: Maximum case, n=8](image)

The red node requires 1 comparison
Then, the purple node requires 2 comparison
The green node requires 2+1 (2 comparison at level 1, 1 comparison one level down at level 2)
Finally, the yellow node requires 2+2+1 (trace the longest path)
Total = $1+2+(2+1)+(2+2+1) = 11$.

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 :).

Ans: Perform a pre-order traversal of the max heap starting from the root. At each node, check if node key is \(> x\). If yes, output node and continue traversal. Otherwise terminate traversal on subtree rooted at current node, return to parent and continue traversal.

\textbf{Algorithm 1 findNodesBiggerThanX(node,x)}

\begin{verbatim}
    if (node.key > x) then
        output(node.key)
        findNodesBiggerThanX(node.left,x)
        findNodesBiggerThanX(node.right,x)
    end if
\end{verbatim}
Analysis of time bound required -
The traversal terminates when it encounters that a node’s key $\leq x$. In the worst case, it encounters $k \times 2$ number of such nodes. That is, each of the left and right child of a valid node (node with key $> x$) are invalid. It will not process any invalid node other than those $k \times 2$ nodes. It will process all $k$ valid nodes, since there cannot be any valid nodes in the subtrees not traversed (due to the heap property). Thus the traversal encounters $O(k + 2 \times k) = O(k)$ number of nodes in order to output the $k$ valid nodes.

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.

Yes it is true. This can be proven easily by proof of contradiction. Suppose the second largest element is not one of the children of the root. Then, it could be the root, but this cannot be since root is the largest element by definition, thus contradiction. Or it has a parent that is not the root :O. There will be a violation to max heap property (there is nothing between largest and 2nd largest element). Contradiction. So, the second element must always be one of the children of the root.

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

Q4. We know that Binary (Max) Heap can be used as Priority Queue and can do $\text{ExtractMax}()$ in $O(\log n)$ time. What modifications/additions/alterations are required so that both $\text{ExtractMax}()$ and
**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?

There is a long answer that uses two Binary Heaps... but it is super complex to explain, so I will skip that.

Competitive Programmer answer is thus: Just use a (balanced) BST to model the Priority Queue. The keys in this bBST are the priority values. Then, to find the minimum/maximum element, we just need to call $O(\log n)$ findMin()/findMax(). Insertion/Enqueue to bBST is still $O(\log n)$. Notice that we do NOT have to associate **PriorityQueue** ADT to be always a **Binary Heap** data structure! We will have discussed BST by the time we discuss this tutorial and will complete the discussion of balanced BST on Wednesday of Week 04.

Tutors are expected to use http://visualgo.net/bst.html to quickly review the concept of BST to support this discussion.

Q5. Follow up from Q4 above: If you can answer Q4 before PS1 is due, will you solve PS1 differently? Competitive Programmer answer: Same as above, just use a bBST to model the Priority Queue.

**ArriveAtHospital** = Insert to bBST $O(\log n)$,

**GiveBirth** = Delete from bBST $O(\log n)$,

**UpdateDilation** = Delete the old value and insert the new value to the bBST :$O(2 \log n = \log n)$,

**Query** = FindMax of bBST, $O(\log n)$.

All operations are still in order of $O(\log n)$...

**Problem Set 2**

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

Again, just whack... $N$ (number of baby names suggestions) is small...

Remind them about corner cases involving [ and ] notations in PS2, e.g. if I have \{“ANDY”, “BOB”\} and my query for Subtask A is Query(“A”, “B”, 0), then actually the answer is 1 (only “Andy”) as “Bob” is already greater than “B”.

Tutors are warned not to touch PS2 subtask B first by Monday/Tuesday of Week04 as the easy solution with Java API will be discussed by Lab TAs.