CS21003 Algorithms-1
Tutorial 4
August 24th, 2018

1. Prove or disprove:
    - **(a)** The minimum value of any max-heap must be present in a leaf.
    - **(b)** The second minimum value of any max-heap must be present in a leaf.

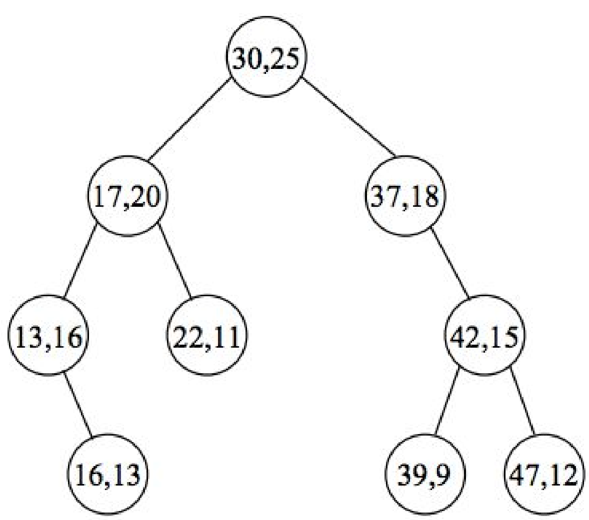
2. Let us add the facility to a priority queue that the priority of an item may change after insertion. Provide an algorithm *changePriority* that, given the index of an element in the supporting array and a new priority value, assigns the new priority value to the element, and reorganizes the array so that heap ordering is restored. Your algorithm should run in O(log n) time for a heap of n elements.

3. Design an O(nlogk)-time algorithm to merge k sorted linked lists having a total of n items.

4. Answer the following questions:
    - (a) How can you find the second and the third maximum elements of a max-heap in constant time?
    - **(b)** Generalize the result for the **k**-th maximum element, where **k** ∈ N is a constant.

5. Propose an algorithm for printing 'k' largest elements in an array of 'n' elements in O (n + klogn) time. Can you now make it more efficient by doing it in O (n+klogk) time?

6. A treap T is a binary search tree with each node storing (in addition to a value) a priority. The priority of any node is not smaller than the priorities of its children. The root is the node with the highest priority. Unlike heaps, a treap is not forced to satisfy the heap-structure property. An example of a treap is given in the adjacent figure, where the pair (x, y) stored in a node indicates that x is the value of the node, and y is the priority of the node. The x values satisfy binary-search-tree ordering, and the y values satisfy heap ordering. An example of a treap is given below.



Design an O(h(T))-time algorithm to insert a value x with priority y in a treap. (Hint: Use rotations.) Insert (18,25).