# CSCI 570 - Fall 2021 - HW 5

## Due date : 30th September 2021

For all divide-and-conquer algorithms follow these steps:

1. Describe the steps of your algorithm in plain English.
2. Write a recurrence equation for the runtime complexity.
3. Solve the equation by the master theorem

1. Solve the following recurrences by giving tight Θ-notation bounds in terms of n for sufficiently large n. Assume that T(·) represents the running time of an algorithm, *i.e.* T(n) is positive and non-decreasing function of n and for small constants c independent of n, T(c) is also a constant independent of n. Note that some of these recurrences might be a little challenging to think about at first.

   a) $T(n) = 4T(n/2) + n^2 \log n$
   b) $T(n) = 8T(n/6) + n \log n$
   c) $T(n) = \sqrt{6000}\, T(n/2) + n^{\sqrt{6000}}$
   d) $T(n) = 10T(n/2) + 2^n$
   e) $T(n) = 2T(\sqrt{n}) + \log_2 n$

2. Consider an array A of n numbers with the assurance that $n > 2$, $A[1] \geq A[2]$ and $A[n] \geq A[n-1]$. An index i is said to be a local minimum of the array A if it satisfies $1<i<n$, $A[i-1] \geq A[i]$ and $A[i+1] \geq A[i]$.

   (a) Prove that there always exists a local minimum for A.
   (b) Design an algorithm to compute a local minimum of A.

   Your algorithm is allowed to make at most O(log n) pairwise comparisons between elements of A.

3. The recurrence $T(n) = 7T(n/2) + n^2$ describes the running time of an algorithm ALG. A competing algorithm ALG′ has a running time of $T'(n) = aT'(n/4) + n^2 \log n$. What is the largest value of a such that ALG′ is asymptotically faster than ALG?

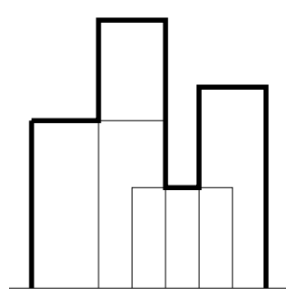
4. Solve Kleinberg and Tardos, Chapter 5, Exercise 3.

5. Given a binary search tree T, its root node *r*, and two random nodes *x* and *y* in the tree. Find the lowest common ancestry of the two nodes *x* and *y*. Note that a parent node *p* has pointers to its children *p.leftChild()* and *p.rightChild()*, a child node does **not** have a pointer to its parent node. The complexity must be $O(\log n)$ or better, where *n* is the number of nodes in the tree.
   Recall that in a binary search tree, for a given node *p*, its right child *r,* and its left child *l*, $r.value() \geq p.value() \geq l.value()$. Hint: use divide and conquer

6. Suppose that you are given the exact locations and shapes of several rectangular buildings in a city, and you wish to draw the skyline (in two dimensions) of these buildings, eliminating hidden lines. Assume that the bottoms of all the buildings lie on the x-axis. Building $B_i$ is represented by a triple $(L_i, H_i, R_i)$, where $L_i$ and $R_i$ denote the left and right x coordinates of the building, respectively, and $H_i$ denotes the building's height. A skyline is a list of x coordinates and the heights connecting them arranged in order from left to right.
   For example, the buildings in the figure below correspond to the following input

   $$(1, 5, 5), (4, 3, 7), (3, 8, 5), (6, 6, 8).$$

   The skyline is represented as follows: (1, **5**, 3, **8**, 5, **3**, 6, **6**, 8). Notice that the x-coordinates in the skyline are in sorted order.



a) Given a skyline of n buildings and another skyline of m buildings in the form ($x_1$, $\mathbf{h_1}$, $x_2$, $\mathbf{h_2}$, ….$x_n$) and ($x'_1$, $\mathbf{h'_1}$, $x'_2$, $\mathbf{h'_2}$, ….$x'_m$) , show how to compute the combined skyline for the m + n buildings in O(m + n) steps.

b) Assuming that we have correctly built a solution to part a, give a divide and conquer algorithm to compute the skyline of a given set of n buildings. Your algorithm should run in O(n log n) steps.