CS466/666, Fall 2009: Assignment 1

Out: September 21, Due: October 7, 5pm

1. **Simulating a queue with two stacks**: You may be familiar with the technique of simulating a queue with two stacks as follows. Keep all elements on one stack. Whenever you are asked to enqueue, add the element at the top of the first stack (= end of the queue). Whenever you are asked to dequeue, temporarily pop everything from the first stack and push it into the second stack, then return the top element of the second stack, and then pop everything off the second stack and push it onto the first.

   This implementation takes $\Theta(n)$ worst-case time for a de-queue, and the amortized time is no better, since n de-queue operations will take $\Theta(n^2)$ time.

   Give a *different* way to simulate a queue with two stacks such that the amortized cost of each Enqueue and Dequeue is O(1). You should only use $O(1)$ memory other than for the two stacks.

2. **Amortized analysis of binary min-heaps**: Recall that in the ordinary binary min-heap structure, Insert is done by inserting the element as the last element of the heap and bubbling up, while ExtractMin is done by removing the root, moving the last element in its place, and bubbling it down. If the binary heap has n elements, then both Insert and ExtractMin take $O(\log n)$ actual time.

   (a) Give a potential function Φ such that the amortized cost of ExtractMin is $O(1)$, but the amortized cost of Insert is still $O(\log n)$. Show that it works.

   (b) Can you find a potential function Φ such that the amortized cost of Insert is $O(1)$, but the amortized cost of ExtractMin is still $O(\log n)$? If yes, give it and show that it works. If no, argue where the difficulty lies.

3. **Euclidean Minimum Spanning Tree:** The Euclidean Minimum Spanning Tree problem is the following: Given n points $p_1, \ldots, p_n$ in the plane, define the complete graph K_n where edge (i, j) has weight equal to the Euclidean distance between p_i and p_j. Then find a minimum spanning tree in the usual sense: find a set T of $n-1$ edges that is connected (hence a tree.)

   Show that any algorithm to solve the Euclidean Minimum Spanning tree must have time complexity $\Omega(n \log n)$, under some reasonable assumptions. (These assumptions will not be given precisely here; figuring out what assumptions you need and stating them precisely are part of this assignment.)

4. **Orthogonally convex hull**: Let Q be a set of n points in the plane. We say that point (x, y) *dominates* point (x', y') if $x \geq x'$ and $y \geq y'$. A point in Q that is dominated by no other points in Q is said to be *maximal*. Note that Q may contain many maximal points. The *orthogonally convex NE-hull* consists of the sequence of maximal points of Q, sorted by increasing x-coordinate.

   (a) Describe an $O(n \log n)$ worst-case time algorithm to compute the orthogonally convex NE-hull.

   (b) Describe an $O(nh)$ worst-case time algorithm to compute the orthogonally convex NE-hull, where h is the size of that hull.

   (c) Can the two algorithms be combined to give an $O(n \log h)$ algorithm for computing the orthogonally convex NE-hull? If yes, give it. If no, argue where the difficulty lies.

Remarks (for all assignment questions in this course):

- You may make assumptions on your input if it simplifies an argument. This may lead to some deduction (e.g., in Question 4, assuming that all x-coordinates are distinct will not give full credit), but usually not very much. Typical assumptions to make are: n is a power of 2, all input numbers are distinct, all edge-weights are distinct, ... Make sure you *state your assumption clearly.*

- If you are asked to show a bound (e.g. on the running time), and you can't do it, then partial credit can often be obtained if you show a weaker bound. For example, in Question 4(a) and (b), any bound less than $O(n^2)$ should get some partial credit.