CS 180, Fall 2015 Homework 6

The following homework is due on Wednesday, November 18th at the beginning of lecture.

When submitting your homework, please include your name at the top of each page. If you submit multiple pages, please staple them together. Late submissions are not accepted.

1. Give an algorithm that takes two permutations of numbers $1, 2, \ldots, n$ and finds a pair of indices i, j such that in one permutation the ith number is bigger than the jth number and in the other permutation the order is reversed. For example, consider two permutations $A = (5, 7, 2, 1, 4, 6, 3)$ and $B = (6, 7, 4, 3, 1, 5, 2)$. In the first list $A[4] = 1 < A[7] = 3$, but in the second list $B[4] = 3 > B[7] = 2$. Thus, the algorithm can return the pair $(4, 7)$ of indices. Alternatively it can return the pair $(3, 5)$. Your algorithm should have the best possible running time. (hint: show that without loss of generality you can assume that the permutations are such that for all i: $A[i] \neq B[i]$).

2. Consider a graph with edge weights that are distinct. We can define an order on the set of spanning trees of the graph as follows. Each tree has $n-1$ edges, by sorting the weights of all $n-1$ tree edges in the increasing order, we obtain a tuple $(w_1, \ldots, w_{n-1})$ where $w_1 < w_2 < \ldots < w_{n-1}$. We can easily define a lexicographic order on tuples by imagining that each tuple is a word with $n-1$ letter. More precisely, we say tuple $(w'_1, \ldots, w'_{n-1})$ is bigger than tuple $(w_1, \ldots, w_{n-1})$ if for the smallest index i at which $w'_i \neq w_i$, we have $w'_i > w_i$. We say tree $T_1 > T_2$ if T_1's tuple is lexicographically bigger than T_2's tuple. We define *lexicographically minimum tree* to be the smallest tree in this ordering of trees.

   (a) prove that every lexicographically minimum tree is a minimum spanning tree.

   (b) Given an undirected graph construct a data structure that allows the following operations: Given any pair of vertices u and v, the data structure can be used to return a path between u and v such that maximum-weight edge on the path has a smaller weight that the maximum-weight edge of any other path between u and v. Prove correctness of your algorithm.

3. (a) Given a graph with *distinct* edge weights, prove that the minimum spanning tree is unique.

   (b) If the edge weights in the graph are not distinct there might be more than one (possibly many) minimum spanning tree(s). Design an algorithm that outputs all minimum spanning trees of a given graph. The running time of your algorithm can depend on the number of MST's which might not be polynomial.

4. Prove that every algorithm for computing the convex hull of a set of n points needs at least $\Omega(n \log n)$ time. To show this, it is enough to prove the following: if you have a $O(n \log n)$ algorithm for computing the convex hull of n points, then you can do sorting in $O(n \log n)$ time. (hint: to prove the latter statement, consider using the points on the curve of $y = x^2$.)