

CS466/666, Fall 2011: Assignment 3

Out: October 19, Due: November 2, 5pm

1. **Minimum spanning trees in planar graphs:** (8 marks) Show that the minimum spanning tree can be found in a planar graph in $O(n)$ time. (Hint: You don't need to know much about planar graphs. All you need to know is that any planar graph without multiple edges or loops has at most $3n$ edges, and that any planar graphs remains planar after contracting or deleting an edge.)
2. **Minimum spanning trees in changing graphs:** (18 marks) Given a graph G and an MST T .
 - (a) Suppose we decrease the weight of one e in T . How quickly can you compute the MST T' of the resulting graph G' ?
 - (b) Suppose we decrease the weight of one edge e not in T . How quickly can you compute the MST T' of the resulting graph G' ?
 - (c) Suppose we add a new vertex v with an arbitrary number of incident edges. How quickly can you compute the MST T' of the resulting graph G' ?

For all three parts, you should give an algorithm and (as usual) argue why it is correct and what its run-time is. You should also justify why it can't be done any faster than what your algorithm achieves.

You may use randomized Las Vegas algorithms if they have a faster expected run-time.

3. **Maintaining lists while splitting:** (7 marks) Show how to maintain a set of doubly-linked lists that can be split. Initially, the set contains just one list with n elements in it. Two operations can be applied, in arbitrary order:
 - Find(x): This returns a unique identifier of the list that currently contains x .
 - Split(ℓ, x): This splits list ℓ into two lists, by splitting after x .

For both operations, x knows where it is in the list. Explain how to implement this data structure such that any Find can be done in $O(1)$ time, and the total time for all Split operations is $O(n \log n)$ time.

4. **Lowest Common Ancestor in binary trees:** (7 marks) Let T be a binary tree with n nodes. Explain how you would preprocess T such that afterwards, you can answer the query for the lowest common ancestor of nodes u and v in $O(d)$ time, where d is the depth of that lowest common ancestor. Your pre-processing should take $O(n)$ time and you should only use $O(1)$ auxiliary space per node.