Honors Analysis of Algorithms

Problem Set 2

Collaboration is allowed, but you must write your own solutions. Proving correctness of algorithms is a must.

Problem 1

Suppose $G(V, E)$ is a connected graph and $h : E \mapsto \mathbf{R}$ is an assignment of costs to its edges. Let $g : E \mapsto \mathbf{R}$ be another cost assignment that satisfies:

$$\forall\ e, e' \in E, \quad h(e) \leq h(e') \iff g(e) \leq g(e').$$

Prove that there exists a spanning tree of G that is a minimum cost spanning tree with respect to costs $h(\cdot)$ as well as a minimum cost spanning tree with respect to costs $g(\cdot)$.

Solve: [Kleinberg Tardos]: Chapter 4, Problem 26, on page 202.

Note: The greedy algorithm for minimum spanning tree (taught in class) works even when costs are allowed to be negative.

Problem 2

Let G be an n-vertex connected graph with costs on the edges. Assume that all the edge costs are distinct.

1. Prove that G has a unique minimum cost spanning tree.
2. Give a polynomial time algorithm to find a spanning tree whose cost is the second smallest.
3. Give a polynomial time algorithm to find a cycle in G such that the maximum cost of edges in the cycle is minimum amongst all possible cycles. Assume that the graph has at least one cycle.

Problem 3

You are given a set of n intervals on a line:

$$(a_1, b_1],\ (a_2, b_2], \ldots, (a_n, b_n].$$

Design a polynomial time greedy algorithm to select minimum number of intervals whose union is the same as the union of all intervals.

Problem 4

Solve: [Kleinberg Tardos]: Chapter 4, Problem 29, on page 203.

Hint: First prove that a non-increasing sequence $(d_1, d_2, \ldots, d_n)$ *is a degree sequence of some* n*-vertex graph if and only if* $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ *is a degree sequence of some* $(n-1)$*-vertex graph.*

Problem 5

Suppose you have an unrestricted supply of pennies, nickels, dimes, and quarters. You wish to give your friend n cents using a minimum number of coins. Describe a greedy strategy to solve this problem and prove its correctness.

Problem 6

Given an array $a[i], 1 \le i \le n$ of integers and an integer b, show how to rearrange the array and find an index k in $O(n)$ time so that (after the rearrangement)

- $a[i] \le b$ for $1 \le i \le k$, and
- $b < a[i]$ for $k < i \le n$.

Your algorithm is not allowed to use any other array (i.e. the rearrangement has to be "in place"). Note that Quick-Sort would use this algorithm as a sub-routine, b being the "pivot"; the nice feature is that no extra storage is needed.