Problem Set 6

Due: Thursday, August 9th at 10:15 a.m.

Problem 1: Renting Skis (20 points)

A ski rental agency has m pairs of skis, where the height of the i^{th} pair of skis is s_i. There are n skiers who wish to rent skis, where the height of the j^{th} skier is h_j. Ideally, each skier should obtain a pair of skis whose height matches his on height as closely as possible. Design an efficient algorithm to assign skis to skiers so that the sum of the absolute differences of the heights of each skier and his skis is minimized.

Problem 2: Building Towers (20 points)

Professor Babylonia wants to construct the tallest tower possible out of building blocks. She has n types of blocks, and an unlimited supply of blocks of each type. Each type-i block is a rectangular solid with linear dimensions $\langle x_i, y_i, z_i \rangle$. A block can be oriented so that any two of its three dimensions determine the dimensions of a base and the other dimension is the height. In building a tower, one block may be placed on top of anything block as long as the two dimensions of the upper block's base are each strictly smaller than the corresponding base dimensions of the lower block. (So, for example, blocks oriented to have equal-sized bases cannot be stacked.) Design an efficient algorithm to determine the tallest tower that the professor can build. (You must account for the fact that the blocks can be reoriented.

Problem 3: Multiple Lecture Halls (15 points, courtesy of CLR, Exercise 17.1-2)

Suppose we have a set of activities to schedule among a large number of lecture halls. We wish to schedule all of the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall. There is an obvious algorithm that runs in $O(n^2)$ time, but there is even better algorithm whose running time is bounded by the time is takes to sort the events.

Problem 4: Topological Sorting Issues (15 points, courtesy of CLR, Exercise 23.4-2)

There are many different orderings of the vertices of the vertices of a directed graph G that are topological sorts of G. `Topological-Sort` (as presented on pp. 485-487 of the text) produces an ordering that is the reverse of the depth-first finishing times. Show that not all topological sorts can be produced in this way—that there exists a graph G such that one of its topological sorts cannot be produced by `Topological-Sort`, no matter what adjacency-list structure is given for G. Show also that there exists a graph for which two distinct adjacency list representations yield the same topological sort.