Computer Science 130B
Spring 2017
Homework #2

Due: **4pm**, April 28, Friday

Problem 1 Consider the coin change problem. You are at a check out register, and have an unlimited supply of quarters, dimes, nickels, and pennies. You have to make change for a customer. Design a greedy algorithm to make out an amount of x cents that uses the smallest number of coins. Prove the optimality of your algorithm.

Problem 2 You are given n intervals on the x-axis, each specified by a starting and an end x-coordinate, or interval i goes from s_i to e_i inclusively and $s_i < e_i$. We want to select a maximum set of mutually non-overlapping intervals. That is, if two intervals i and j are both selected, either $e_i \leq s_j$ or $e_j \leq s_i$. Here, maximum means that the selected set contains the largest number of the input intervals. This problem has many real-world applications in task scheduling. For example, the intervals can correspond to availability of clients at a doctor's office and the scheduling goal is to accommodate as many patients as possible.

There are many greedy strategies that can be used to select intervals from the given set. For example, some reasonable strategies are selecting intervals based on (1) their starting x locations (smaller ones first), (2) their end x locations (smaller ones first), (3) their lengths (shorter ones first), and (4) the number of other intervals that they overlap with (fewer overlaps first). It is not clear if all these different strategies will generate the same results and/or if any of them will always generate the optimal results (the most number of intervals included). For each strategy, give a counter example if you believe that such a strategy will not generate the optimal results. If you believe a strategy will always generate the optimal results (with most intervals), prove it.

Problem 3 Assume n programs of length $l_1, l_2, \cdots, l_n$ are to be stored on a tape. Program i is to be retrieved with frequency f_i. If the program are stored in the order of $i_1, i_2, \cdots, i_n$, the expected retrieval time (ERT) is

$$\frac{\sum_j (f_{i_j} \sum_{k=1}^{j} l_{i_k})}{\sum_j f_{i_j}}$$

a. Show that storing programs in nondecreasing order of l_i does not necessarily minimize ERT.
b. Show that storing programs in non-increasing order of f_i does not necessarily minimize ERT.
c. Show that storing programs in non-increasing order of f_i/l_i does minimize ERT. **Note:** You can assume that the tape is long enough to hold all the programs. So $\sum_j f_{i_j} = 1$.