Assignment 4
COSC 3P03: Design and Analysis of Algorithms
Fall 2019

Due: Dec. 4, Wed, 5:00 PM.
The total of 100 points include a bonus of 20 marks.

1. (25) Consider the activity-selection problem covered in class wehere we discussed a greedy algorithm that always finds an optimal solution. However, not just any greedy approach to the activity-selection problem produces a maximum-sized set of mutually compatible activities. (a) Give an example to show that the approach of selecting the activity of least duration from those that are compatible with previously selected activities does not work. (b) Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and (c) always slecting the compatible remaining activity with the earliest start time.

An easy way to give such examples is to use a horizontal line segment to represent an activity (see the following figure). For example, in the following example, there are three activities. Also, your example should be such that the strategies will fail regardless how a tie (a tie happens when we have two or more activities that satisfy the greedy condition at the same time) is broken.

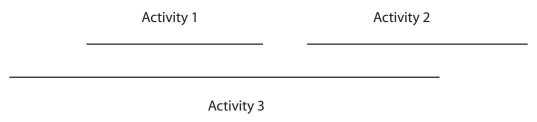

Figure 1: Three Activities.

2. (10) We have unlimited number of bins each of capacity 1, and n objects of sizes s_1, s_2, ..., s_n, where $0 < s_i \leq 1$. Our job is to pack these objects into bins using the fewest bins possible (optimization). The decision version can be stated as follows: given n objects and k, is there a packing using no more than k bins. Show how to solve one version if you know how to solve the other version, i.e., show how to solve the decision version by solving the optimization version and show how to solve the optimization version by solving the decision version.

3. (15) The greedy algorithm to solve the fractional knapsack problem optimally selects objects in decreasing value to weight ratios (discussed in class). For the 0-1 knapsack problem:
(a) show that this greedy algorithm does not work;
(b) show that the greedy method not only does not work, it yields arbitrarily bad solutions, that is, given any $0 < \epsilon < 1$, construct an example where the ratio of the greedy solution to the optimal solution is $< \epsilon$. Note that if you are able to (b), then your answer will also serve as an answer to (a). In this case, there is no need to do (a). On the other hand, if you are unable to do (b), you should try to do (a).

4. (10) A person wishes to cross a desert carrying only a single water bottle. He has a map that marks all the watering holes along the way. Assuming he can walk k miles on one bottle of water, design an efficient algorithm for determining where he should refill his water bottle in order to make as few stops as possible. Argue why your algorithm is correct.

5. (10) Sally is hosting an Internet auction to sell n widgets. She receives m bids, each of the form "I want k_i widgets for d_i dollars," for $i = 1, 2, \cdots, m$. Show how this problem can be formulated as a 0-1 knapsack problem.

6. (20) Consider the job scheduling problem we discussed in class. But this time, each job j_i with time t_i is now associated with a probablity c_i. So now the average waiting time for a schedule $j_{i_1} j_{i_2} \cdots j_{i_n}$, where $i_1 i_2 \cdots i_n$ is a permutation of 1, 2, ..., n, is computed as follows:

$$\textit{Average waiting time} = p_{i_1} t_{i_1} + p_{i_2}(t_{i_1} + t_{i_2}) + \cdots + p_{i_n}(t_{i_1} + t_{i_2} + \cdots + t_{i_n}).$$

In our original version, $p_i = 1/n$, $1, \leq i \leq n$. Suppose that our goal is to minimize the average time for the general case, (1) will our strategy of shortest job first generate an optimal solution? (2) what about arranging jobs in decreasing order of probablities, i.e., placing jobs with the bigger probablities at the beginning?

7. (10) Suppose we are given a black-box that has the following properties: for any sequence of real numbers and an integer k, it will answer yes or no, depending on whether there is a subset of numbers whose sum is k. Show how to use this black box to find the subset whose sum is k for the subset sum problem whose input is $\{x_1, x_2, \cdots, x_n\}$, a set of positive integers, and an integer k.