COMPSCI330 Design and Analysis of Algorithms
Assignment 3

Due Date: Wednesday, February 12, 2020

Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Typesetting and Submission** Please submit the problems to GradeScope. You will be asked to **label your solution for individual problems**. Failing to label your solution can cost you 5% of the total points (3 points out of 60 for this homework).
- LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you need more time for your homework please use this form and submit a STINF.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (Counter-Examples)**.** (16 points) Give a counter-example to each greedy strategy. When you give a counter-example, you need to specify the input, the output of the greedy strategy and the optimal solution.
(a) (8 points) Consider the interval scheduling problem. Your input should be n, the number of meetings, followed by n pairs of numbers $(s_i, t_i)(0 \leq s_i < t_i)$, which are the starting time and ending time of the meetings. The goal is to schedule as many meetings as possible. Consider the strategy where the algorithm schedules the shortest meeting first (the meeting with smallest $t_i - s_i$). Give a counter-example for this strategy (your example should have $n \leq 4$).
(b) (8 points) Consider the Longest Increasing Subsequence problem (the problem we solved using Dynamic Programming). Your input should be n, the length of the sequence, followed by n numbers $a[1..n]$. The goal is to find the longest increasing subsequence of $a[1..n]$. Consider the strategy where we enumerate the starting location $a[i]$, then scan the array from $a[i]$ to the right, and whenever a number can be added to the subsequence (meaning it's larger than all numbers added before) then the algorithm adds it to the sequence. The strategy picks the longest sequence (among all starting locations) generated this way. Give a counter-example for this strategy (your example should have $n \leq 5$).

Problem 2 (Diet Options II)**.** (20 points) Remember Rong decided to go on a diet and he wants to have between A and B calories at every meal ($0 < A < B$). Now he is at a buffet which has n items. He can take these items in any fractions. One unit of item i has $c_i > 0$ calories, and a (very subjective) rating of tastiness t_i (which need not be positive). Precisely, Rong can take a fraction $0 \leq p_i \leq 1$ for every item, and receive $p_i c_i$ calories and $p_i t_i$ tastiness (the buffet does not have a constraint of $p_i \leq 1$, however Rong does not like to eat more than 1 unit of the same item). Rong is looking for some items to take, such that the meal will have at least A calories and at most B calories, while the sum of tastiness is as large as possible. Please design an algorithm to help Rong. More precisely, given n, A, B, c_i, t_i, output the maximum sum of tastiness.

(a) (10 points) Design an algorithm to help Rong and analyze the running time of your algorithm.

(b) (10 points) Prove the correctness for the algorithm you designed in (a).

Problem 3 (Getting Coffee)**.** (24 points) Rong is sleep deprived and needs coffee to stay awake. During the day he has n meetings, meeting i has starting time s_i and ending time t_i. The meetings are scheduled to have no conflict (that is, for every $1 \leq i < n$, $t_i \leq s_{i+1}$. We also know that $s_1 \geq 0$ and $t_n \leq T$.

Rong needs to stay awake for the entire duration of the meetings. He can do so by getting coffee in between the meetings (that is, between time t_i and s_{i+1} for $1 \leq i < n$, or between 0 and s_1). Drinking coffee takes 0 time (even if $t_i = s_{i+1}$ Rong can get a coffee at time t_i). Every cup of coffee allows Rong to stay awake for a length L (which is longer than length of every meeting). Your goal is to design an algorithm to minimize the number of cups Rong will take. Note: the times to drink two cups of coffees do not need to be L apart.

(a) (6 points) Design an algorithm to compute the minimum number of coffees Rong will take. Analyze its running time.

(b) (10 points) Prove the correctness for the algorithm you designed in (a).

(c) (8 points) Rong decides that he only wants to drink K cups of coffee per day. He can get different amount of coffee so the length L can be changed. Design an algorithm to compute

what is the smallest length L that allows Rong to drink K cups of coffee while still being able to stay awake for all the meetings. Analyze its running time. You don't need to prove correctness for this problem. (Note that in this subproblem K is an input and L is what you need to output. For this subproblem you can assume every number in the problem is an integer, and the smallest length L is also an integer. Hint: You can use algorithm you designed in (a) as a subroutine.)