CSCI 3100, Fall 2018

Homework 3

For this homework, you may work in groups and discuss problem solutions with your classmates. While you can work in groups to figure out the solutions to problems, your need to write up your solutions on your own. On your write-up, **include names of people with whom you discussed the solutions**. Please type up your solutions.

Problem 1

Consider a greedy algorithm for determining the minimum number of bills to use to pay for an item that has price P. The algorithm is this:

1. Initialize an empty result list
2. Find the largest denomination bill that is smaller than P. Suppose the value of that denomination is d.
3. Add that denomination to the result list.
4. Subtract d from P: P = P-d
5. Continue to step 2, until P becomes 0.

For example, consider bill denominations: 1, 10, 20, and 100. Suppose price P is 120. Then the greedy algorithm will come up with 2 bills: 100 and 20 and that solution is optimal.

a. [5 points] Given arbitrary bill denominations (so not necessarily denominations used for US Dollars), this algorithm does not produce the smallest number of bills. Find an example set of bill denominations and a price P, for which this algorithm does not find an optimal solution.

b. [20 points] Provide an alternative algorithm for determining smallest number of bills that can be used to pay for an item with price P. Your algorithm should produce an optimal solution regardless of the available bill denominations, assuming 1 is always included in your denominations. For example, your algorithm should produce an optimal solution for the example you provided in part (a) of this problem. (Hint: consider how rod cutting algorithm can be adjusted to work for this problem.)

b. [20 points] Prove that, given bill denominations of the form 1, 4, 8, …, 2^k (for any value of k > 0), the greedy algorithm produces the smallest number of bills needed to pay for an item with price P.

Problem 2

Consider the following scheduling problem. You are given a set of activities A_1, A_2, …, A_N. Each activity *i* is broken into two independent parts (S_i, F_i). Each part has a duration. Let d(S_i) be the duration of the first part of activity *i*, and d(F_i) is the duration of the second part of activity *i*. The

first part of each activity must be completed prior to the second part. Additionally, to do the first part, all activities require a common resource so the first parts of all activities must be scheduled sequentially. The second part of all activities can be done in parallel.

a. [5 points] Come up with a greedy strategy for scheduling activities such that latest activity completion time is minimized.

b. [20 points] Prove that your greedy strategy leads to an optimal solution.

Problem 3

[10 points] Recall that a sequence S' is a subsequence of S if there is a way to delete certain elements from S so that the remaining elements, in order, are equal to the sequence S' (we covered this during the maximum longest subsequence lecture). Let S' and S be two sequences, with |S'|=m and |S|=n. Give a O(n+m) algorithm to determine if S' is a subsequence of S.

Problem 4

A small photocopying service business with a single large machine has the following scheduling problem. Each day they get a set of jobs from customers. They want to do the jobs on a single machine in an order that keeps customers happiest. Customer *i*'s job will take t_i time to complete. Given a schedule (an order of the jobs), let C_i denote the finishing time of job *i*. For example, if job *k* is the first to be done, we have $C_k=t_k$; and if job *k* is done right after job *i*, we have $C_k = C_i + t_k$. Each customer *i* also has a given weight w_i that represents his or her importance to the business. The happiness of customer *i* is expected to be dependent on the finishing time of job *i*. The company decides that they want to order the jobs to minimize the weighted sum of the completion times $\sum_{i=1}^{n} w_i C_i$.

[20 points] Design an efficient algorithm to solve this problem.