Computer Science C73 October 27, 2021
Scarborough Campus University of Toronto

Homework Assignment #6
(worth 6% of the course grade)
Due: November ~~10~~ 14, 2021, by 11:59 pm

• ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
• *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.*
• *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Recall that a dynamic programming algorithm to solve a given problem P involves the following elements:
(a) *A definition of a polynomial number of subproblems that will be solved (and from whose solution we will compute the solution to P — see (c) below).*
(b) *A recursive formula to compute the solution to each subproblem from the solutions to smaller subproblems. This induces a partial order on the subproblems defined in (a).*
(c) *A way to compute the solution to P from the solutions to the subproblems computed in (b).*

Proving the correctness of a dynamic programming algorithm amounts to justifying (i) why the recursive formula in step (b) correctly computes the subproblems defined in step (a), and (ii) why the computation in (c) yields a solution to the given problem. Part (ii) is often immediate from the definition of the subproblems.

Question 1. (15 marks) Business is booming for the catering service described in Question 1 of Assignment 2. The decision is therefore made to expand the variety of orders so that each may take any number

of time units, not necessarily just one as in the original setting. To ensure the high quality of the product, however, the same single cook still does all the cooking. The goal is to find a most profitable schedule for filling orders in this new and more general setting.

To be more precise, the catering service receives a set of n orders. Order i, $1 \leq i \leq n$, has a ***deadline*** d_i by which it must be completed, a ***length*** $\ell_i \leq d_i$ that it requires to be completed, and a ***profit*** $p_i > 0$ that the catering service receives if the order is completed by the deadline; if the order is completed after the deadline there is no profit made and therefore it is not filled at all. Deadlines and lengths are expressed as ***positive integers***, to be thought of as multiples of an appropriate time unit.

The cook starts work at time 0. The catering service must decide on a schedule in which to fill a subset of the orders, each completed by its deadline. A schedule now is a function S that maps each order i to the finish time $S(i)$ of preparing that order, where $S(i) \geq \ell_i$ is a positive integer, meaning that order i will be prepared during the interval $\big(S(i) - \ell_i, S(i)\big)$; or to the finish "time" $+\infty$, meaning that order i will not be filled. We require that, for any distinct orders i and j, if $S(i) \neq +\infty$ and $S(j) \neq +\infty$, then the intervals $\big(S(i) - \ell(i), S(i)\big)$ and $\big(S(j) - \ell_j, S(j)\big)$ are disjoint: the single cook works on one order at a time and brings it to completion before starting another one. The problem is to design an algorithm that, given as input the n triples (d_i, ℓ_i, p_i), returns as output a schedule S that maximizes the total profit: for every schedule S', $\sum_{i:\, S(i) \neq +\infty} p_i \geq \sum_{i:\, S'(i) \neq +\infty} p_i$.

a. Give a counterexample proving that the following greedy strategy does not work: Sort and consider the orders by non-increasing profit. For the current order i, if there is a time t such that $\ell_i \leq t \leq d_i$ and the interval $(t - \ell_i, t)$ is disjoint from all intervals of jobs scheduled so far, let t^* be the maximum such time and schedule order i to finish at time t^* — i.e., set $S(i) := t^*$); otherwise, don't fill order i — i.e., set $S(i) := +\infty$.

b. Describe and justify the correctness of a dynamic programming algorithm that finds the ***maximum profit*** achievable by scheduling a subset of the given orders. Analyze the running time of your algorithm. Is your algorithm a polynomial-time one? Explain why or why not.

Hint: Sort the orders by non-~~in~~decreasing deadline, and take the subproblems to be the maximum profit $P(i, t)$ achievable by scheduling a subset of orders $1..i$ so that they all finish by time t.

c. Retrofit your algorithm for part (b) to find an optimal schedule (as opposed to its profit). It suffices to show only the changes to your algorithm for part (b).

Question 2. (10 marks) We are given an $n \times n$ array $B[1..n, 1..n]$ of bits (0s or 1s). We want to find a largest square of 1s within this array. (For example, the array could represent a map, where a 0 denotes a square parcel of private land and a 1 denotes a square parcel of public land. We want to find the largest possible square piece of public land, say to build a new school.)

More precisely, a ***square of 1s*** is a triple of integers (i, j, ℓ) such that $i, j \geq 1$, $\ell \geq 0$, $i + \ell \leq n$, $j + \ell \leq n$, and $B[i', j'] = 1$ for all i', j' such that $i \leq i' \leq i + \ell$ and $j \leq j' \leq j + \ell$. We want to find a square of 1s such that ℓ is as large as possible.

Describe and justify the correctness of a dynamic programming algorithm to solve this problem, and analyze its running time. For full credit, your algorithm should run in $O(n^2)$ time.

Hint: For each i, j, consider the size of a largest square of 1s whose lower right corner is (i, j).

Question 3. (10 marks) Let $G = (V, E)$ be an undirected graph, $\mathbf{wt} : E \to \mathbb{R}$ be an edge weight function such that $\mathbf{wt}(e) > 0$ for every $e \in E$, $V' \subseteq V$, and $d \in \mathbb{R}$ be a positive real number. The graph represents a roap map of a rural area. Each node corresponds to a town, each edge corresponds to a road that connects two towns, and the weight of an edge indicates the length of that road. The subset V' is the subset of

towns that have a gas station, and d is the maximum distance that your car can travel on a full tank of gas.

Given the above information and two nodes $s, t \in V'$, you want to find the shortest route so you can drive from s to t without running out of gas; that is, any portion of the route that is longer than d should pass through a town with a gas station. Note that s and t are both in V' — so you can fill up before the trip starts and when it ends.

Using one or more algorithms described in class, describe a polynomial time algorithm that takes as input G, $\mathbf{wt}$, V', d, and $s, t \in V'$, and returns the desired route (as a sequence of towns). Analyze the running time of your algorithm. For full credit your algorithm should run in $O(n^3)$ time.

You should describe your algorithm in clear, point-form English; do not write pseudocode. You do not need to justify the correctness of your algorithm, as it it should be immediate from the algorithm(s) it uses. Algorithms designed "from scratch", which do not take advantage of algorithms discussed in this course, will receive no credit. (This is to help you avoid designing unnecessarily complicated, and probably wrong, solutions.)