Computer Science CSC373H
St. George Campus

January 20, 2021
University of Toronto

Homework Assignment #2

Due: February 3, 2021, by 11:00 am

- **You must submit your assignment through the Crowdmark system.** You will receive by email an invitation through which you can submit your work. If you haven't used Crowdmark before, give yourself plenty of time to figure it out!
- You must submit a **separate** PDF document with for **each** question of the assignment.
- To work with one or two partners, you and your partner(s) must form a **group** on Crowdmark (one submission only per group). We allow groups of up to three students, submissions by groups of more than three students will not be graded.
- The PDF file that you submit for each question must be typeset (**not** handwritten) and clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.
- If this assignment is submitted by a group of two or three students, for each assignment question the PDF file that you submit should contain:
  1. The name(s) of the student(s) who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.
- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy for this course, as stated in: `http://www.cs.toronto.edu/~sam/teaching/373/#HomeworkCollaboration`.
- 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 (in lectures or tutorials) by referring to it.
- 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.
- The total length of your pdf submission should be no more than 4 pages long in a 11pt font.

Question 1. (10 marks) Consider the following variation on the *Interval Scheduling Problem* that we saw in lecture. You have a processor that operates 24 hours a day, every day. People submit requests to run *daily* jobs on the processor; there are a total of n such job requests. Each job request comes with a start time and an end time. If the job is accepted to run on the processor, it must run continuously *every day*, for the period between its start and end times. Note that jobs intervals can straddle any time of the day (including midnight). Moreover, the accepted jobs are executed repeatedly (forever) in 24-hour cycles; intuitively, here the job intervals are arcs of a circle rather than segments of a straight timeline. This makes for a type of situation different from what we saw in the Interval Scheduling Problem.

Given a list of n such job requests, your goal is to accept as many jobs as possible subject to the constraint that the processor can run at most one job at any given time. Describe an algorithm to do this with an asymptotic running time that is smaller than $\Theta(n^3)$. Your can use any algorithm or result that we covered in class without reproving it. For simplicity, you may assume that no two jobs have the same start or end times.

EXAMPLE: Consider the following four jobs specified by (start-time, end-time) pairs:

(6pm, 6am), (9pm, 4am), (3am, 2pm), (1pm, 7pm)

The optimal solution here is to pick the two jobs (9pm, 4am) and (1pm, 7pm), which can be scheduled without overlapping.

Describe your algorithm ***clearly and concisely*** in English (you don't need to give the pseudo-code), and give its asymptotic running time. Explain why your algorithm works, and briefly justify the running time.

Question 2. (10 marks) Consider n lakes, labelled $1, 2, \ldots, n$, that are navigable by canoe and are connected by portages (i.e., footpaths that can be used to go from one lake to another by carrying a canoe). For each lake i, we are given a list $L[i]$ containing pairs of the form (j, d_{ij}) where j is a lake $j \neq i$ that is reachable from i by a portage, and d_{ij} is a non-negative integer indicating the ***degree of difficulty*** of carrying a canoe from lake i to lake j on that portage. Note that because lakes i and j may be on different heights, the difficulty d_{ij} of carrying a canoe from i to j may be different from the difficulty d_{ji} of doing it from j to i (if a portage from j to i exists).

A ***trip*** from a lake s to a lake t is a path from s to t that involves traversing a sequence of lakes and portages. The ***toughness*** of a trip is the greatest degree of difficulty among all the portages on that trip. An ***easiest*** trip from a lake s to a lake t is a trip from s to t with minimum toughness among all possible trips from s to t.

a. (5 marks) Your task is to modify Dijkstra's algorithm so that, given any starting lake s, the algorithm: (1) finds an easiest trip from s to each other lake t, and (2) computes the toughness of this trip. If no trip from s to a lake t exists, the algorithm should report that the toughness of the easiest trip from s to t is ∞.

Describe the algorithm's modification clearly and concisely in English, and then give the algorithm's pseudocode. Give, and briefly justify, the complexity of your modified algorithm.

b. (5 marks) A real-estate promotion company is now offering a free chauffeured limousine service (with a trailer to carry canoes) from some lakes to some other lakes. Since they also offer free breakfast, lunch, and dinner with pink champagne and fancy gifts, the difficulty degree d_{ij} of going from a lake i to a lake j that has such a limousine service can be ***negative***. Does your modified Dijkstra's algorithm still work correctly? (Recall that Dijkstra's algorithm for shortest-paths does ***not*** work with negative weights).

• If it does not work correctly, explain why by giving a specific counterexample. Your counterexample should be as small as possible, i.e., one with the fewest number of lakes and portages.

• If it works correctly, explain why. To do so: (1) point out where the proof of correctness of Dijkstra's original algorithm for the shortest-path problem relies on the assumption of non-negative weights, and

(2) why this assumption is not necessary to the correctness proof of your modified Dijkstra's algorithm (for finding the easiest paths).

Question 3. (10 marks) Consider the following *Halloween Costume Fitting Problem*. There are n actors and n Halloween costumes (each one representing a different monster). Each actor must be given one of the costumes. We are given the height of each actor and the length of each costume (both are non-negative integers). If an actor of height h is given a costume of length ℓ, then the fitting cost is $|h - l|$. We want an efficient algorithm to find an optimal assignment of costumes to actors, more precisely, one that minimizes the *total* fitting costs (i.e., the sum of the fitting costs for all n actors).

a. (5 marks) Bombazzino Californiano says that the following greedy algorithm is optimal: Find an (actor,costume) pair with minimum fitting cost (i.e., that has the smallest $|h - l|$ over all (actor,costume) pairs), and match these two. Continue doing so until you run out of actors and costumes. Is Bombazzino's algorithm optimal? If so give a proof, otherwise give a small counterexample.

b. (5 marks) Rizzuto Siciliano proposes a more efficient and simpler greedy algorithm: sort the actors and costumes by increasing height and length, respectively, and assign costume i to actor i, for every i $(1 \leq n)$. Is Rizzuto's algorithm optimal? If so give a proof, otherwise give a small counterexample.