St. Francis Xavier University
Department of Computer Science

CSCI 355: Algorithm Design and Analysis
Assignment 2
Due February 8, 2024 at 1:30pm

Assignment Regulations.

- This assignment must be completed individually.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible.

[8 marks] 1. The university registrar is creating the timetable for next year, and they need to schedule each of the following courses in such a way that no two courses are assigned to the same classroom at the same time:

- CSCI 355, Algorithm Design and Analysis: 11am–12pm
- CSCI 356, Theory of Computing: 10am–11:30am
- MATH 110, Binary Arithmetic: 12:30pm–1:30pm
- BSKT 201, Advanced Basketweaving: 12:30pm–2pm
- NISH 101, How to Pronounce Antigonish: 9:30am–12:30pm
- MEAL 247, Cuisine of Morrison Hall: 9:30am–10:30am
- STFX 499, Hagiography of St. Francis Xavier: 9am–10:30am
- JOHN 316, Popular Bible Verses: 11:30am–12:30pm
- HTML 404, Website Troubleshooting: 10:30am–12pm
- SPIE 007, Espionage Fundamentals: 12pm–1pm

(a) Treat this situation as an instance of the interval partitioning problem, and find a solution to the instance requiring as few classrooms as possible.

(If you like, you may implement an algorithm to solve the interval partitioning problem in the programming language of your choice, but your answer must include a copy of your source code and a verbose listing of each step of your implementation's output.)

(b) Instances of interval partitioning problems can be represented by *interval graphs*, where vertices correspond to lectures and an edge between two vertices indicates that those respective lectures are mutually incompatible. Draw the interval graph for the interval partitioning instance in part (a).

(c) We can alternatively solve instances of interval partitioning problems by assigning colours to the vertices of an interval graph in such a way that two adjacent vertices do not share the same colour. Find a vertex colouring for your interval graph from part (b) that uses as few colours as possible. How does this number of colours relate to the number of classrooms you found in part (a)?

[7 marks] 2. Consider a variant of the interval scheduling problem where we are scheduling *daily recurring* jobs on a single machine that runs 24 hours a day; that is, each scheduled job starts and ends at the same time every day. Note the major difference between the standard problem and this variant: jobs may start before midnight and end after midnight, so the timeline is not discretized into a single 24-hour interval.

Given a list of n jobs and the start time s_j and finish time f_j for each job j, devise an algorithm that schedules as many jobs as possible such that no two jobs run at the same time. You may assume that no two jobs share the same start times or finish times. Your algorithm must have a runtime polynomial in n, but you do not need to prove optimality.

[6 marks] 3. As Canadian drivers well know, weather has a big impact on the time required to drive from one place to another. Travelling on a clear, dry road takes much less time than travelling on a wet, snowy road. For this reason, a group of enterprising computer science students has created an app that predicts how much time will be needed to drive between two locations based on the current season.

This app handles input in the form of a road $r = (a, b)$ between two locations a and b together with a departure time t, and returns a value $f_r(t)$ indicating the predicted travel time between departing a at time t and arriving at b. The value $f_r(t)$ is arbitrary, aside from two conditions: $f_r(t) \geq t$ for all roads r and times t (you can't travel back in time) and $f_r(t)$ is a monotone increasing function (you can't arrive earlier by departing later).

Construct an algorithm that uses these values $f_r(t)$ to find the fastest route between any two locations a and b. You do not need to prove the correctness of your algorithm, but it should run in polynomial time.

Hint. Can we modify an existing graph algorithm to work for this problem?

[4 marks] 4. When we give the same input graph G to Kruskal's algorithm multiple times, the algorithm can return different minimum spanning trees depending on how it "breaks ties" between edges having the same weight. However, we can modify Kruskal's algorithm in such a way that we can force it to give us a specific minimum spanning tree for G.

Show that, for every minimum spanning tree T of a graph G, there is a way to sort the edges of G so that Kruskal's algorithm is guaranteed to return T.