This assignment is **due on May 15** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the assignment.

Problem 1. (20 points)
We are given a simple connected undirected graph $G = (V, E)$ with edge costs $c : E \to R_+$. We would like to find a spanning binary tree T rooted a given node $r \in T$ such that T has minimum weight.

Consider the following MODIFIEDPRIM algorithm that works similar to Prim's MST algorithm: We maintain a tree T (initially set to be r by itself) and in each iteration of the algorithm, we grow T by attaching a new node T in the cheapest possible way such that we do not violate the binary constraint; if it is not possible to grow the tree, we declare the instance to be infeasible.

```
1: function MODIFIEDPRIM(G=(V, E), r)
2:     $T \leftarrow \{r\}$
3:     while $|T| < |V|$ do
4:         $S \leftarrow \{u \in V : u \in T$ and $|children(u)| < 2\}$
5:         $R \leftarrow \{u \in V : u \notin T\}$
6:         if $\exists\, (u, v) \in E$ with $u \in S$ and $v \in R$ then
7:             let $(u, v)$ be the minimum cost such edge
8:             Add $(u, v)$ to $T$
9:         else
10:            return infeasible
11:    return $T$
```

Your task is to either prove the correctness of MODIFIEDPRIM or provide a counter example where it fails to return the correct answer.

Problem 2. (40 points)
My colleague, Dr. Strange tells me that there is an alternate universe where Sydney buses are never cancelled and they always run on time; in fact, bus timetables specify to the second the exact arrival and departure time of each bus on each stop. Sadly, in this alternate reality there are no free transfers (i.e., you need to pay for the full fare of every bus you ride) and different bus lines charge different fees (but the good news is that the fee is flat and does not depend on distance travelled).

A travel plan is a sequence of *stop-time* pairs where *stop* is a location of a bus stop and *time* is when we arrive at that stop. The plan is feasible if for any two consecutive pairs (a, t) and (b, t') in the plan there exists a bus that departs a after t and arrives at b at exactly t'. That is, a travel plan does not allow us to walk between stops. Assuming that no two buses arrive at the same time at the same stop, a feasible plan uniquely identifies the bus lines that we need to take to realize the plan. The cost of the plan is the sum of the fares we need to pay.

Your task is to design an efficient algorithm that given a departure time t, an arrival time t', an origin stop a and a destination stop b, finds the cheapest travel plan from a to b that obeys the departure and arrival time constraints (we start traveling at or after t and we arrive to our destination at or before t'). Remember to

a) Describe your algorithm in plain English.

b) Prove the correctness of your algorithm.

c) Analyze the time complexity of your algorithm.

Suppose that in this alternative universe, there are m bus lines, each having k instances, and each having exactly ℓ stops. The timetable is therefore made up of mk lists (one for each instance of each bus line) each having triplets of the form $(stop, arrival, departure)$, so the size of the instance is $\Theta(mk\ell)$. To get full marks your algorithm must run in time that is polynomial on m, k, and ℓ.

Problem 3. (40 points)
Given a collection of n shapes on the plane, a valid traversal from shape A to shape B is a sequence of shapes starting at A and ending at B such that every two consecutive shapes overlap. The cost of a traversal is the sum of the areas of the individual shapes in the traversal. Suppose we only have two types of shapes: squares and circles. Your task is to design an algorithm to find the minimum cost traversal from A to B subject to the constraint that we are not allowed to use consecutive squares in the traversal. Remember to:

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

You can assume that there is routine that for a given shape computes its area in $O(1)$ time and another routine that tests if two shapes overlap in $O(1)$ time. To get full marks your algorithm needs to run in $O(n^2)$ time.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When asked to give a plain English description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, when designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you marks for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.

- Finally, to make marking run more smoothly, please specify in with your Gradescope submission, which pages in your submission cover which problem.