CSCI 3104 Fall 2021
Instructors: Profs. Grochow and Waggoner

Midterm 1- Standard 4

Due Date .... October / 11th / 2021
Name .... **Michael Ghattas**
Student ID **109200649**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You **may not collaborate with other students**. **Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1.

- My submission is in my own words and reflects my understanding of the material.
- I have not collaborated with any other person.
- I have not posted to external services including, but not limited to Chegg, Discord, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

I agree to the above, Michael Ghattas. □

3 Standard 4- Examples Where Greedy Algorithms Fail

3.1 Problem 2

Problem 2. Suppose you want to drive from Town A to Town B along some fixed route of distance d, and you have a gas tank whose capacity will take you at most m miles along the route. Let $0 < d_1 < d_2 < ... < d_n < d$ be the distances of each gas station along the route from Town A. (So the distance from A to gas station 2 is d_2; the distance between the first two gas stations is $d_2 - d_1$.) Your goal is to get from A to B (equivalently, distance 0 to distance d) (a) without running out of gas, and (b) stopping as few times as possible.

The natural strategy most people use is a greedy one: go as far as you can, but refuel at gas station i if you don't have enough to get you to gas station $i+1$. This strategy indeed minimizes the number of stops you need to make.

Consider a different greedy strategy, in which you stop at the nearest available gas station. Give an example (specify d, m, and the distances between the gas stations) showing that this strategy is not optimal. Show what this greedy algorithm does on your example, which subset of gas stations it outputs, and exhibit a strictly smaller set of gas stations that would still allow you to successfully complete the trip.

Answer:

Lets take a closer look at the two algorithms mentioned in the problem. If we let Q be the set of visited gas stations, let D be the set of all gas stations, and assume there exists more than one solution (i.e. we do not have to visit every gas station to complete our journey), **then an optimal solution should give us $|Q| < |D|$, while a sub-optimal solution would give us $|Q| = |D|$. Below is our proof of both algorithms and their respective outputs:**

- **Optimal algorithm will give us $|Q| < |D|$**

```
func roadTrip{D[$d_1 \rightarrow d_n$], m(max fuel)}
  location = null
  Q = [0]
  m = int(max fuel)
  while[(m != 0) && (location != $d_n$)]:
    for i in length(D):
      for j in length(D - 1):
        $d_j = d_i + 1$
        if [m < ($d_j$ - $d_i$)]:
          location = $d_i$
          m = int(max fuel)
          add $d_i \rightarrow$ Q
        else:
          location = $d_j$
          m = int(max fuel)
          add $d_j \rightarrow$ Q
return(Q);
```

- **Sub-optimal algorithm will give us $|Q| = |D|$**

```
func roadTrip{D[$d_1 \rightarrow d_n$], m(max fuel)}
  location = null
  Q = [0]
  m = int(max fuel)
  while[(m != 0) && (location != $d_n$)]:
    for i in length(D):
      location = $d_i$
      m = int(max fuel)
      add $d_i \rightarrow$ Q
return(Q);
```

□