CSCI 332, Fall 2024
Homework 5

Due before class on Tuesday, October 8, 2024–that is, due at 9:30am Mountain Time

---

Submission Requirements

- Type or clearly hand-write your solutions into a PDF format so that they are legible and professional. Submit your PDF on Gradescope.
- Do not submit your first draft. Type or clearly re-write your solutions for your final submission.
- Use Gradescope to assign problems to the correct page(s) in your solution. If you do not do this correctly, we will ask you to resubmit.
- You may work with a group of up to three students and submit ***one single document*** for the group. Just be sure to list all group members at the top of the document. When submitting a group assignment to Gradescope, only one student needs to upload the document; just be sure to select your groupmates when you do so.

Academic Integrity

Remember, you may access ***any*** resource in preparing your solution to the homework. However, you ***must***

- write your solutions in your own words, and
- credit every resource you use (for example: "Bob Smith helped me on this problem. He took this course at UM in Fall 2020"; "I found a solution to a problem similar to this one in the lecture notes for a different course, found at this link: www.profzeno.com/agreatclass/lecture10"; "I asked ChatGPT how to solve part (c)"; "I put my solution for part (c) into ChatGPT to check that it was correct and it caught a missing case.") If you use the provided LaTeX template, you can use the `sources` environment for this. Ask if you need help!

Grading

Remember, submitted homeworks are graded for completeness, not correctness. Correctness is evaluated using homework quizzes.

Each submitted problem will be graded out of six points according to the following rubric:

- Does the solution address the correct problem?
- Does the solution make a reasonable attempt at solving the problem, even if not fully correct?
- Is the presentation neat?
- Is the explanation clear?

- Does the solution list collaborators or sources, or state that the student did not use any collaborators or outside resources?
- Is the solution written in the student's own voice (not copied directly from an outside resource)?

1. The pseudocode for Dijkstra's algorithm is given below. The input to the algorithm is a directed graph G with edge weights denoted by ℓ_e for edge e.

> Let S be the set of explored nodes
> For each $u \in S$, we store a distance $d(u)$
> Initially $S = \{s\}$ and $d(s) = 0$
> While $S \neq V$:
> Select a node $v \notin S$ with at least one edge from S for which $d'(v) = \min_{e=(u,v):u\in S} d(u) + \ell_e$ is as small as possible
> Add v to S and define $d(v) = d'(v)$

Answer the following questions about Dijkstra's algorithm. For your answers, use n to denote the number of nodes and m to denote the number of edges in G.

(a) How many times *exactly* does the while loop run? (This should be a function of m and/or n.)

(b) Suppose that we implement the while loop naively by examining every $v \notin S$, computing $\min_{e=(u,v):u\in S} d(u) + \ell_e$, and then using those values to find the next node to add to S. Fill in the blank: such an implementation of the while loop takes $O($_____$)$ steps each time it is called. (What you fill in should be a function of m and/or n. Also, notice that there are many functions that would work to make the statement true—it's most useful to you for the rest of the problem to choose the smallest one.)

(c) Using your answers to (a) and (b), give an upper bound on the runtime of a naive implementation of Dijkstra's algorithm. That is, fill in the blank in this statement: A naive implementation of Dijkstra's algorithm takes $O($_____$)$. (What you fill in should be a function of m and/or n.)

(d) Notice that we did not specify whether this was a worst case, best case, average case, or any specific case runtime. Are the following true, false, or unknown? Note: you may want to check your work on (b) and (c) before answering these.

- A naive implementation of Dijkstra's algorithm takes $O(m^2n^2)$ steps.
- A naive implementation of Dijkstra's algorithm takes $O(m^2n^2)$ steps in the worst case.
- A naive implementation of Dijkstra's algorithm takes $O(m^2n^2)$ steps in the best case.
- A naive implementation of Dijkstra's algorithm takes $O(m^2n^2)$ steps in the average case.
- A naive implementation of Dijkstra's algorithm takes $O(mn^2)$ steps.
- A naive implementation of Dijkstra's algorithm takes $O(mn^2)$ steps in the worst case.
- A naive implementation of Dijkstra's algorithm takes $O(mn^2)$ steps in the best case.
- A naive implementation of Dijkstra's algorithm takes $O(mn^2)$ steps in the average case.

2. (This is similar to problem 2 from Chapter 4, part a)

For each of the following statements, decide whether it is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

(a) Suppose we are given an instance of the Shortest s-t Path Problem on a directed graph G. We assume that all edge costs are positive and distinct. Let P be a minimum-cost s-t path for this instance. Now suppose we replace each edge cost c_e by its square, c_e^2, thereby creating a new instance of the problem with the same graph but different costs.

True or false? P must still be a minimum-cost s-t path for this new instance.

(b) Suppose we are again given an instance of the Shortest s-t Path Problem on a directed graph G. We assume that all edge costs are positive and distinct. Let P be a minimum-cost s-t path for this instance. Now suppose we replace each edge cost c_e by $c_e + 21$, thereby creating a new instance of the problem with the same graph but different costs.

True or false? P must still be a minimum-cost s-t path for this new instance.

3. Dijkstra's algorithm may fail to find the minimum-cost s-u distances to all nodes u for a graph with edges with negative weights. For each of the following, given an example graph and show how $d(u)$ would be computed for each node u according to Dijkstra's algorithm:

(a) A graph with at least one negative-weight edge where Dijkstra's algorithm finds the correct minimum-cost distances to all nodes.

(b) A graph with at least one negative-weight edge where Dijkstra's algorithm fails to find the correct minimum-cost distances to all nodes.