Northeastern University
CS5800 – Algorithms
Fall 2023 Semester, Jonathan Mwaura

Problem Set #7

This Problem Set is due at **11:30PM on Friday, Nov** 10^{th}, and will be submitted on GRADESCOPE.

The problem set will be marked out of 50.

Problem #3, part 1 can be carried out as a group assignment. Make sure that you write the name of all your colloborators on top of the page for this problem if you work with others. Problem #1 and Problem #2 (part has to be carried out individually).

Please type (or neatly handwrite) your solutions on standard 8.5×11 paper, with your name at the top of each solution. Ensure that you submit you solutions in one file PDF file on Gradescope. **each problem sets solution should be on in its own individual page, Gradescope will help ensure you submit each solution under its correct problem number**

While a solution must be absolutely perfect to receive full marks, I will be generous in awarding partial marks for incomplete solutions that demonstrate progress.

So that there is no ambiguity, there are two non-negotiable rules. A violation of either rule constitutes plagiarism and will result in you receiving an F for this course.

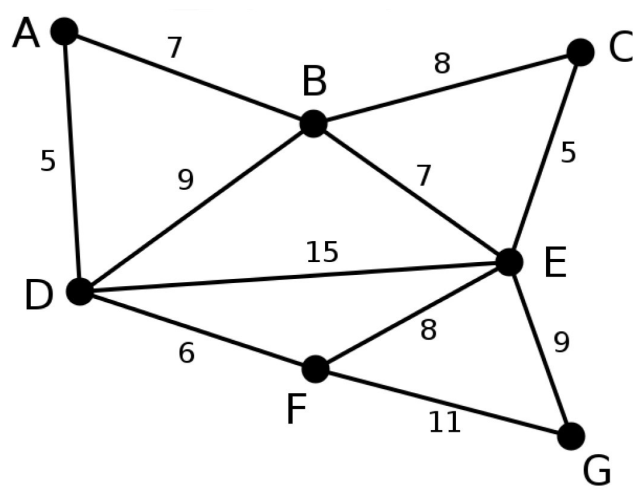
(a) If you meet with a classmate to discuss any of the Individual Problems, your submission must be an individual activity, done in your own words, away from others. The process of finding a solution might take 3 - 5 iterations or even more BUT you learn from all these attempts and your confidence grows with each iteration.

(b) These problem sets might seem hard on a first look. They are designed to be so. We learn by attempting problems, struggling through them and coming on top. I encourage you to make this learning exercise worth your while. What do I mean? Open the problem sets as early as you get them, then do not look at hints or answers any where (including on the internet and consulting other students for direct answers), give it the best shot you can. If you get stuck come to Professor or TA's office hour and we shall be glad to listen to your rationale and work with you till you are able to tackle the problem sets.

Problem #1 (15 points)

In this question you will explore Dijkstra's Single Source Shortest Path algorithm

(a) Consider the following weighted undirected graph with 7 vertices and 11 edges.



Apply Dijkstra's Algorithm on the graph above, to determine the shortest distance from vertex G to each of the other six vertices (A, B, C, D, E, F). Clearly show all of your steps.

(b) Now suppose we change the weight of edge EF from +8 to −8. What happens?

Using this example, explain why Dijkstra's Algorithm can produce incorrect outputs when one or more edges is negative.

(c) Determine a precise Loop Invariant for the Dijkstra's Algorithm, clearly stating your Initialization, Maintenance, and Termination statements. Prove that your loop invariant holds, clearly and carefully justifying each step in your proof.

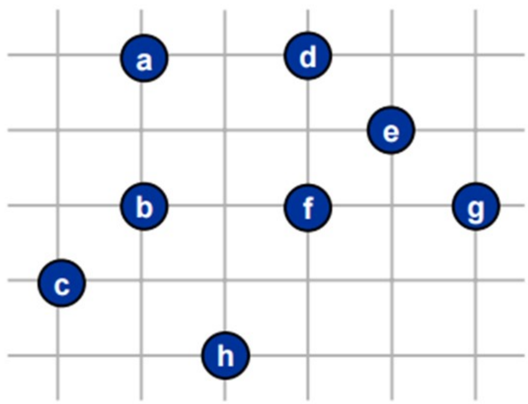
Problem #2 (20 points)

In this question you will explore algorithms that generate Minimum-Weight Spanning Trees.

(a) Let G be a graph with V vertices and E edges. One can implement Kruskal's Algorithm to run in $O(E \log V)$ time, and Prim's Algorithm to run in $O(E + V \log V)$ time.

If G is a *dense* graph with an extremely large number of vertices, determine which algorithm would output the minimum-weight spanning tree more quickly. Clearly justify your answer.

(b) Consider eight points on the Cartesian two-dimensional x-y plane.



For each pair of vertices u and v, the weight of edge uv is the Euclidean (Pythagorean) distance between those two points. For example, $dist(a, h) = \sqrt{4^2 + 1^2} = \sqrt{17}$ and $dist(a, b) = \sqrt{2^2 + 0^2} = 2$.

Using the algorithm of your choice, determine one possible minimum-weight spanning tree and compute its total distance, rounding your answer to one decimal place. Clearly show your steps.

(c) Because many pairs of points have identical distances (e.g. $dist(h, c) = dist(h, b) = dist(h, f) = \sqrt{5}$), the above diagram has more than one minimum-weight spanning tree.

Determine the total number of minimum-weight spanning trees that exist in the above diagram. Clearly justify your answer.

(d) Suppose the n points are situated so that each of the $\binom{n}{2} = \frac{n(n-1)}{2}$ distances are *distinct* positive numbers.

Prove that graph G has only one minimum-weight spanning tree. Clearly explain each step in your proof.

Problem #3 (15 points)

There are over 200 LeetCode problems on Greedy Algorithms.

In this question, you will create a **mini-portfolio** consisting of LeetCode problems on Greedy Algorithms, chosen from the following website.

`https://leetcode.com/list/50f6p33i/`

As always, you may code your algorithms in the programming language of your choice.

Here is how your mini-portfolio will be graded.

(i) There will be a total of 10 points for any of the combination of problems in your mini-portfolio: For each of these, provide the problem number, problem title, difficulty level, and the screenshot of you getting your solution accepted by LeetCode (10 points).

Note that you are allowed to work with Teammates on this part of the problem. Make sure you write all names of the collaborators.

To get the total points for this question, you could submit any of the following options:

- 4 Easy problems
- 2 Easy problems and 1 either hard or medium
- 2 either hard or medium problems or 1 hard and 1 Medium

You will get full credit for *any* correct solution accepted by LeetCode, regardless of how well your runtime and memory usage compares with other LeetCode participants.

(ii) (5 points) For **one** of the problems you are including in your mini-portfolio, explain the various ways you tried to solve this problem, telling us what worked and what did not work. Describe what insights you had as you eventually found a correct solution. Reflect on what you learned from struggling on this problem, and describe how the struggle itself was valuable for you.

The choice of problems is yours, though you may only include problems that took you a minimum of 30 minutes to solve.

I ask you to only include new problems that you will solve in the next seven days. However, I will make an exception if you previously solved a problem in an inefficient way (but still got the solution accepted by LeetCode) and then found a new way to solve the same problem using the methods uncovered in this module on Greedy Algorithms.