CSCI 3104 Spring 2023
Instructors: Ryan Layer and Chandra Kanth Nagesh

Homework 7

Due Date ..... February 16, 2023
Name ..... **Blake Raphael**
Student ID **109752312**
Collaborators ..... **Alex Barry and Brody Cyphers**

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 Gradescope page** only (linked from Canvas). 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 are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. 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, 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)

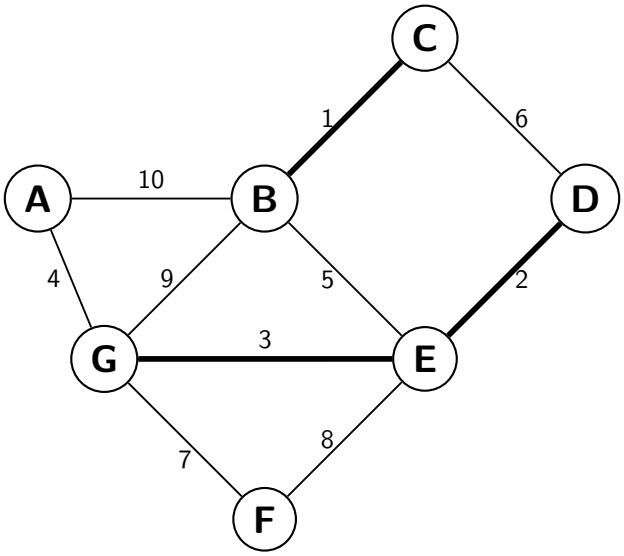
- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (I agree to the above, Blake Raphael). □

3 Standard 7: Kruskal's MST Algorithm

3.2 Problem 1 (2 points)

Problem 1. Consider the weighted graph $G(V, E, w)$ below. Clearly list the order in which Kruskal's algorithm adds edges to a minimum-weight spanning tree for G. Additionally, clearly articulate the steps that Kruskal's algorithm takes as it selects the first **three** edges.



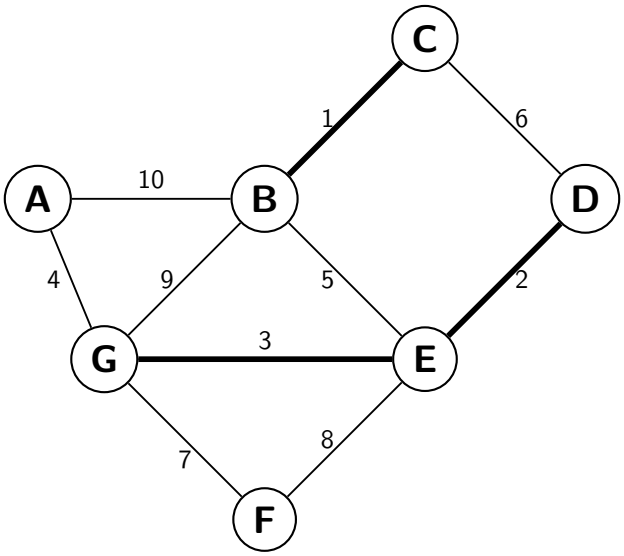
Proof. Kruskal's algorithm adds the edges in the following order:
1: $\{B, C\}$
2: $\{D, E\}$
3: $\{G, E\}$
4: $\{A, G\}$
5: $\{B, E\}$
6: $\{F, E\}$

First we add all the edges to the priority queue ordered in ascending order based on edge weights. Next, we check if the edge creates a cycle or there is already a path connecting those vertices. If not, we add the edge and pop it from the queue. For the first 3 edges, we check if $\{B, C\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{B, C\}$ from the queue. Next we check if $\{D, E\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{D, E\}$ from the queue. Next we check if $\{G, E\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{G, E\}$ from the queue. □

4 Standard 7: Prim's MST Algorithm

4.1 Problem 2 (2 points)

Problem 2. Consider the weighted graph $G(V, E, w)$ below. Clearly list the order in which Kruskal's algorithm adds edges to a minimum-weight spanning tree for G. Additionally, clearly articulate the steps that Kruskal's algorithm takes as it selects the first **three** edges. Consider the weighted graph $G(V, E, w)$ below. Clearly list the order in which Prim's algorithm, **using the source vertex** A, adds edges to a minimum-weight spanning tree for G. Additionally, clearly articulate the steps that Prim's algorithm takes as it selects the first **three** edges.



Proof. Prim's algorithm adds the edges in the following order:
1: $\{A, G\}$
2: $\{G, E\}$
3: $\{E, D\}$
4: $\{E, B\}$
5: $\{B, C\}$
6: $\{G, F\}$

First we add all the edges to the priority queue ordered in ascending order based on edge weights starting from our source node. Next, we check if the edge creates a cycle or there is already a path connecting those vertices. We add edges to the priority queue if they are child nodes from the source node or children nodes of the source node. If not, we add the edge and pop it from the queue. For the first 3 edges, we check if $\{A, G\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{A, G\}$ from the queue. Next we check if $\{G, E\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{G, E\}$ from the queue. Next we check if $\{E, D\}$ creates a cycle or there is a path between the two. Since it does neither, we add that edge and pop $\{E, D\}$ from the queue. □