CSCI 3104 Spring 2023
Instructors: Prof. Layer and Chandra Kanth Nagesh

Homework 2

Due Date ..... February 2, 2023
Name ..... **Blake Raphael**
Student ID **109752312**
Collaborators ..... **Brody Cyphers and Alex Barry**

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 2- BFS and DFS

3.1 Problem 1 (1.5 points)

Problem 1. Consider the Connectivity problem:

- Instance: Let $G(V, E)$ be a simple, undirected graph. Let $s, t \in V(G)$.
- Decision: Is there a path from s to t in G?

Do the following. [**Note:** There are parts (a) and (b). Part (b) is on the next page.]

(a) Design an algorithm to solve the Connectivity problem. Your solution should provide enough detail that a CSCI 2270 student could reasonably be expected to implement your solution.

Answer for Part (a). Connectivity Problem

```
1: Start at node 's'
2: Push node 's' into the queue
3: Mark 's' as visited
4: Pop from queue and visit the neighbors of 's'
5: Push the neighbors of 's' into the queue if they have not been marked as visited yet
6: Mark the neighbors of 's' as visited
7: Continue procesing the neighbors of the neighbors of 's'
8: Repeat until all nodes are marked as visited or we mark 't' as visited
9: If 't' is visited, then we know that there is a path from 's' to 't'
```

Since we marked both nodes s and t as visited in our search, we know there is a path from s to t so we have solved the connectivity problem. □

(b) We say that the graph G is *connected* if for every pair of vertices $s, t \in V(G)$, there exists a path from s to t. Design an algorithm to determine whether G is connected. Your algorithm should only traverse the graph once- this means that you should **not** apply BFS or DFS more than once. Your solution should provide enough detail that a CSCI 2270 student could reasonably be expected to implement your solution.

Answer for Part (b). Is Graph G connected?

```
1: Begin at any node N
2: Put node N into the queue
3: Mark N as visited and iterate a node counting variable
4: Pop node N from the queue
5: Visit the neighbors of N
6: If the neighbors have not been visited, add them to the queue
7: Mark the neighbors in the queue as visited
8: Add to the counting variable the number of new nodes visited
9: Pop the neighbors from the queue
10: Continue visiting the neighbors of the neighbors until all nodes are marked as visited
11: Compare the counting variable to the total number of nodes in our graph
12: If these numbers are equal then we have determined that the graph is connected
```

If the counting variable and number of nodes are equal then we have determined the graph is connected since all nodes were visited. This implies there is at least one path that can get us from one node to any other node. □

3.2 Problem 2 (1 point)

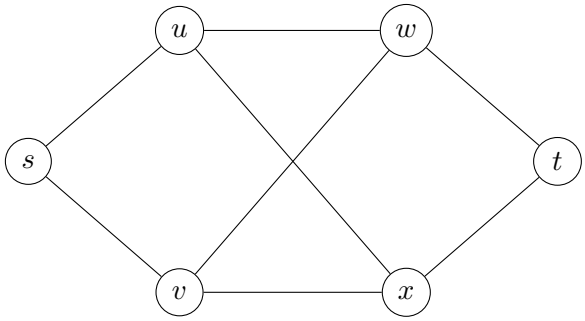
Problem 2. Give an example of a simple, undirected, and unweighted graph $G(V, E)$ that has a single source shortest path tree which a **depth-first traversal** will not return for any ordering of its vertices. Your answer must

(a) Provide a drawing of the graph G. [**Note:** We have provided TikZ code below if you wish to use LaTeX to draw the graph. Alternatively, you may hand-draw G and embed it as an image below, provided that (i) your drawing is legible and (ii) we do not have to rotate our screens to grade your work.]

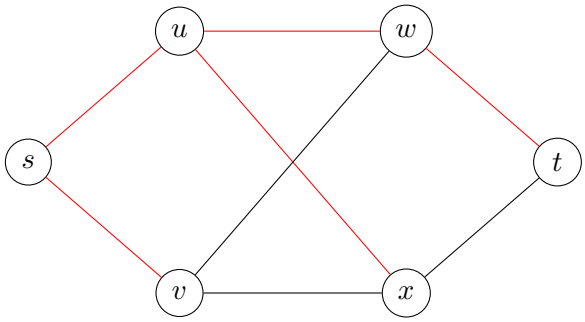
(b) Specify the single source shortest path tree $T = (V, E_T)$ by specifying E_T and also specifying the root $s \in V$. [**Note:** You may again hand-draw this tree. If you wish, you may clearly mark the edges of T on your drawing of G. Please make it easy on the graders to identify the edges of T.]

(c) Include a clear explanation of why the depth-first search algorithm we discussed in class will never produce T for any orderings of the vertices.

Proof. (a) Graph G



(b) The single shortest path tree T on graph G will be the following:



(c) DFS will not produce T because it will never process both neighbor nodes of root s first, instead opting to exploit one of the neighbor nodes until it reaches a visited node or a node that does not have any more neighbors.

□

3.3 Problem 3 (1.5 points)

Problem 3. Give an example of a simple, undirected, weighted graph such that a breadth-first traversal outputs a search-tree that is not a single source shortest path tree. (That is, BFS is not sufficiently powerful to solve the shortest-path problem on weighted graphs. This motivates Dijkstra's algorithm, which will be discussed in the near future.) Your answer must

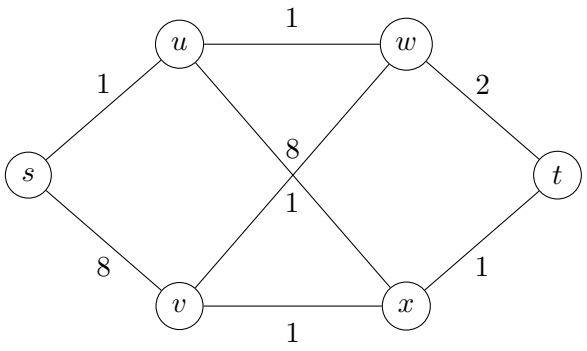
(a) Draw the graph $G = (V, E, w)$ by specifying V and E, clearly labeling the edge weights. [**Note:** We have provided TikZ code below if you wish to use LaTeX to draw the graph. Alternatively, you may hand-draw G and embed it as an image below, provided that (i) your drawing is legible and (ii) we do not have to rotate our screens to grade your work.]

(b) Specify a spanning tree $T(V, E_T)$ that is returned by BFS, but is not a single-source shortest path tree. [**Note:** You may again hand-draw this tree. If you wish, you may clearly mark the edges of T on your drawing of G. Please make it easy on the graders to identify the edges of T.]

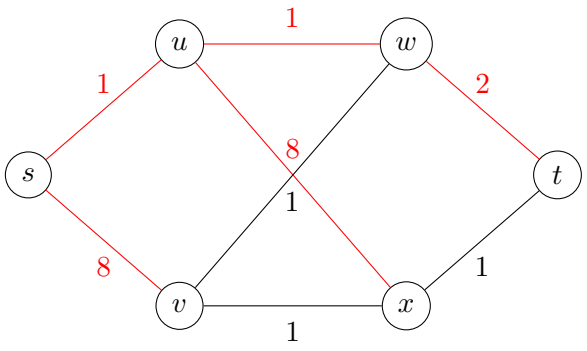
(c) Specify a valid single-source shortest path tree $T' = (V, E_{T'})$. [**Note:** You may again hand-draw this tree. If you wish, you may clearly mark the edges of T on your drawing of G. Please make it easy on the graders to identify the edges of T.]

(d) Include a clear explanation of why the search-tree output by breadth-first search is not a valid single-source shortest path tree of G.

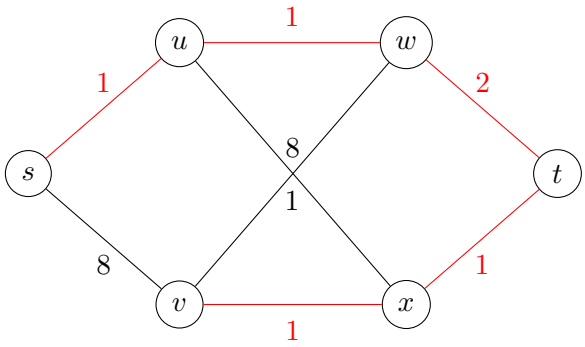
Proof. (a) Graph G



(b) Spanning tree T returned by BFS:



(c) Valid single-source shortest path tree T':



(d) The search-tree provided by BFS is not a valid single-source path tree because, while providing us with the shortest amount of "hops" from start node to end, by adding the weights, we see this is not our shortest path. We get a path with weight 20 where the valid single-shortest path tree provides us a path with weight 6.

□