CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 2

Due Date ..... February 1, 2022
Name ..... **Your Name**
Student ID ..... **Your Student ID**
Collaborators ..... **List Your Collaborators Here**

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 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)

Problem 1.

- 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 (signature here). I agree to the above. Ethan Richman. □

3 Standard 5- BFS and DFS

3.1 Problem 2

Problem 2. Consider a Modified Connectivity problem:

- Instance: Let $G(V, E)$ be a simple, undirected graph. Let $s, t \in V(G)$.
- Decision: Given an integer $k \geq 1$, is there a shortest path from s to t in G that consists of k edges? Here the length/weight of a path is defined as the number of edges of the path.

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

(a) Design an algorithm to solve the Modified 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). First run breadth first search on graph G. Then extract the distance value from the vertices s,t that came from bfs. Then compare it to k. □

(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). Create a Boolean array that is the size of the number of vectors in graph G. Start depth first search from any vertex and mark the visited vertices as true in your array. Once dfs has finished iterate through your array and total the number of true values. If the value is equal to the number of vertices then the graph is connected if it is less then it is not connected. □

3.2 Problem 3

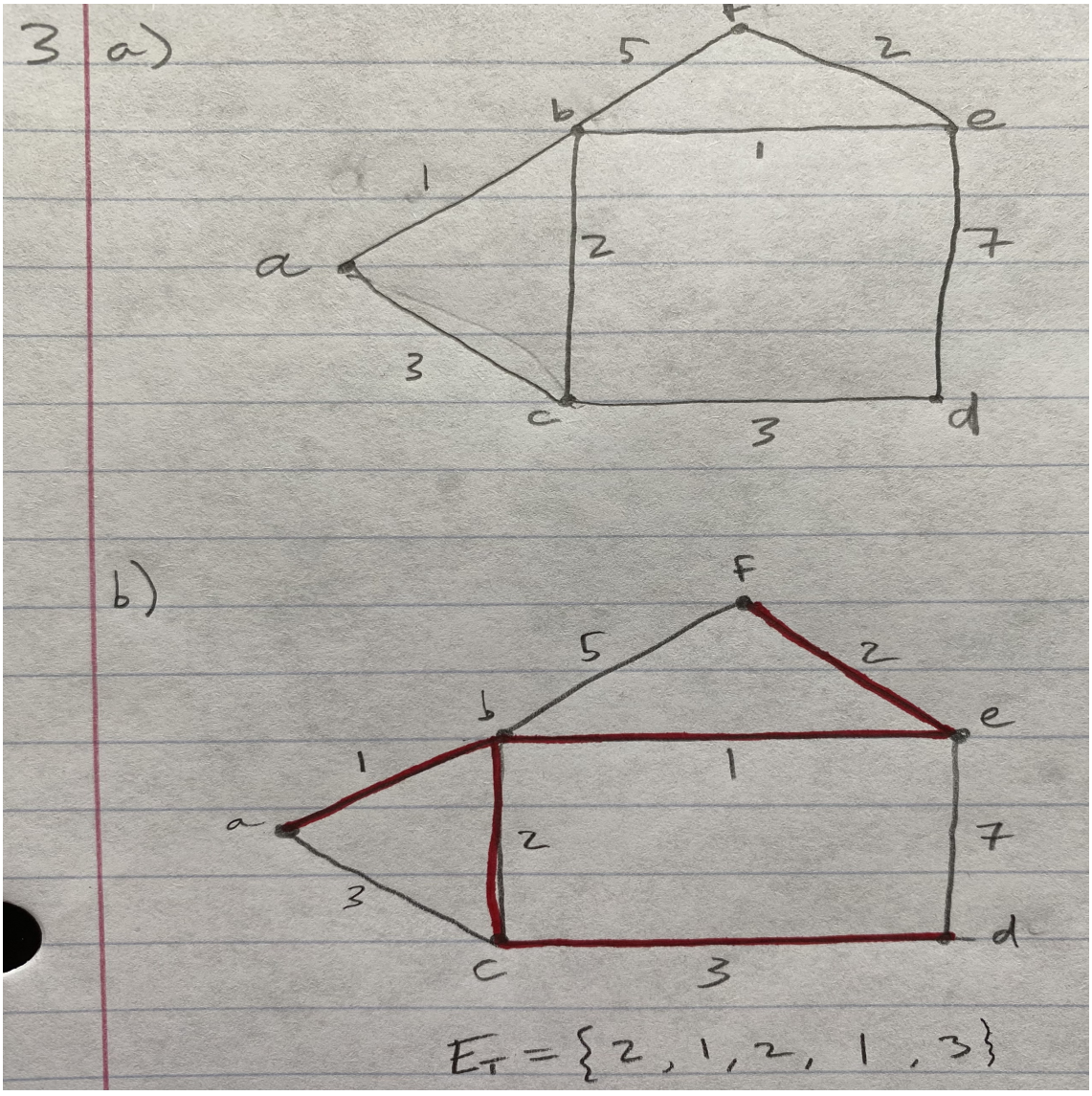
Problem 3. Give an example of a simple, undirected, and unweighted graph $G(V, E)$ that has a single source shortest path tree which a **depth-first traversal** will not return for any ordering of its vertices. Your answer must

(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.

Answer.



c: Depth-first search selects a path and continues on that path not accounting for the neighboring nodes in each iteration. Unless every vertex has a degree of either 1 or 2 dfs will hardly ever give the shortest path. □

3.3 Problem 4

Problem 4. 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.) Your answer must

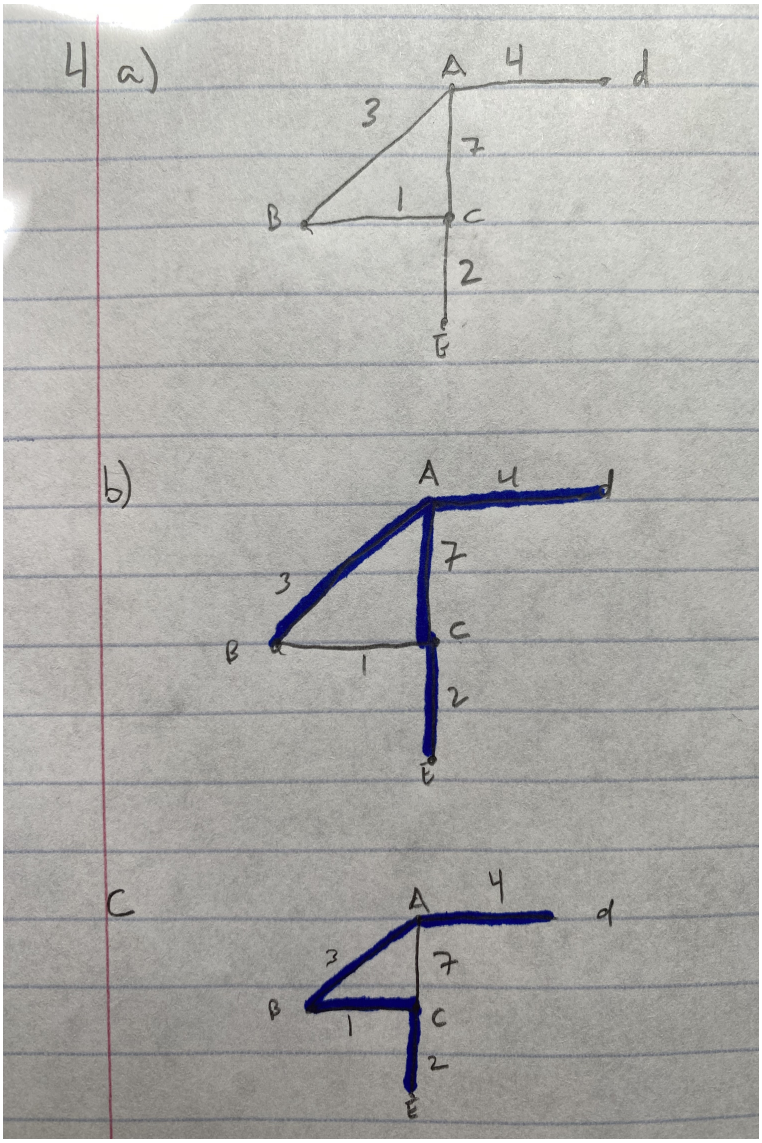
(a) Draw the graph $G = (V, E, w)$ by specifying V and E, clearly labeling the edge weights. [**Note:** We have provided TikZ code below if you wish to use LaTeX to draw the graph. Alternatively, you may hand-draw G and embed it as an image below, provided that (i) your drawing is legible and (ii) we 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.

Answer. □



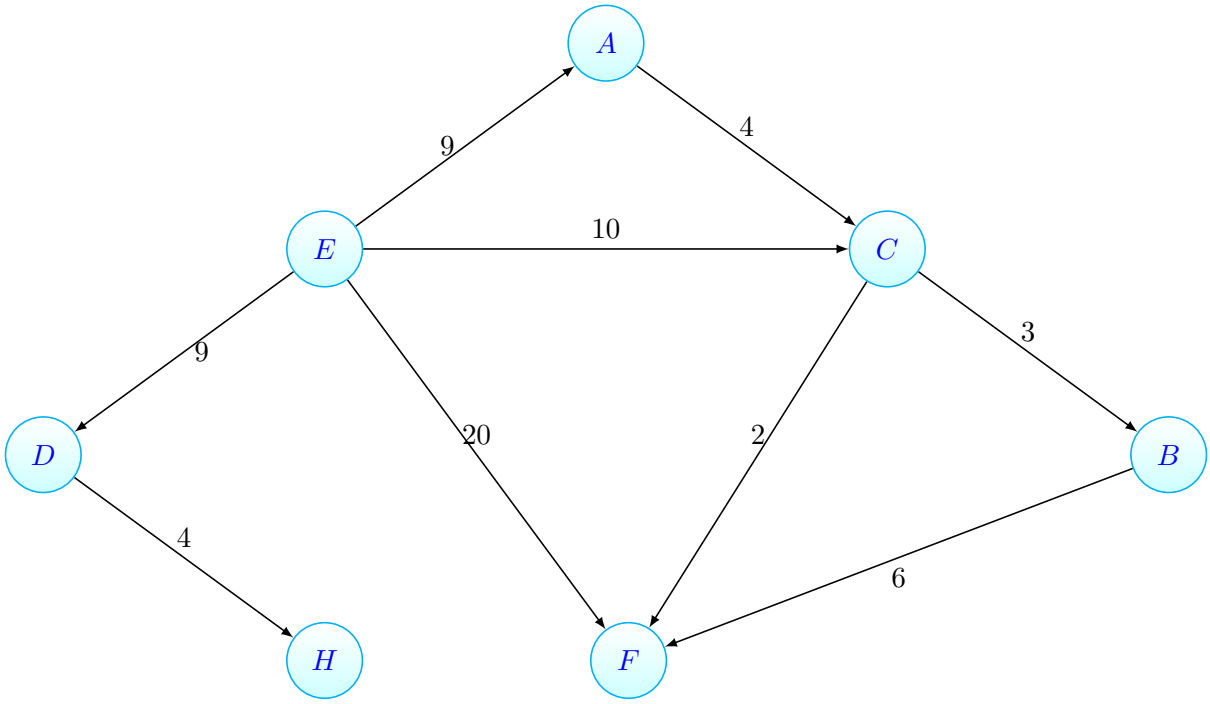
d: The search-tree output by bfs is not a valid single shortest path because the distance between vertices is calculated by the number of edges between the vertices and the source but not by the edge weight. So in the example above when running bfs it goes from A directly to C because there is one edge in between the two. But the shortest path from A to C is actually going from A to B to C because of the respective edge weights.

4 Standard 6- Dijkstra's Algorithm

4.1 Problem 5

Problem 5. Consider the weighted graph $G(V, E, w)$ pictured below. Work through Dijkstra's algorithm on the following graph, using the source vertex E.

- Clearly include the contents of the priority queue, as well as the distance from E to each vertex at each iteration.
- If you use a table to store the distances, clearly label the keys according to the vertex names rather than numeric indices (i.e., `dist['B']` is more descriptive than `dist['1']`).
- You do **not** need to draw the graph at each iteration, though you are welcome to do so. [This may be helpful scratch work, which you do not need to include.]



Answer. **Priority Queue:**
Q=[(E,0)]
Q=[(A,9),(D,9),(C,10),(F,20)]
Q=[(D,9),(C,10),(F,20)]
Q=[(C,10),(H,13),(F,20)]
Q=[(F,12),(H,13),(B,13)]
Q=[(H,13),(B,13)]
Q=[(B,13)]

Shortest Path:
E to A: The shortest path found was E-A which had weight 9
E to B: The shortest path found was E-C-B which had weight 13
E to C: The shortest path found was E-C which had weight 10
E to D: The shortest path found was E-D which had weight 9
E to F: The shortest path found was E-C-F which had weight 12
E to H: The shortest path found was E-D-H which had weight 13 □

4.2 Problem 6

Problem 6. You have three batteries, with capacities of 40, 25, and 16 Ah (Amp-hours), respectively. The 25 and 16-Ah batteries are fully charged (containing 25 Ah and 16 Ah, respectively), while the 40-Ah battery is empty, with 0 Ah. You have a battery transfer device which has a "source" battery position and a "target" battery position. When you place two batteries in the device, it instantaneously transfers as many Ah from the source battery to the target battery as possible. Thus, this device stops the transfer either when the source battery has no Ah remaining or when the destination battery is fully charged (whichever comes first).

But battery transfers aren't free! The battery device is also hooked up to your phone by bluetooth, and automatically charges you a number of dollars equal to however many Ah it just transfered.

The goal in this problem is to determine whether there exists a sequence of transfers that leaves exactly 10 Ah either in the 25-Ah battery or the 16-Ah battery, and if so, how little money you can spend to get this result.

Do the following.

4.2.1 Problem 6(a)

(a) Rephrase this is as a graph problem. Give a precise definition of how to model this problem as a graph, and state the specific question about this graph that must be answered. [**Note:** While you are welcome to draw the graph, it is enough to provide 1-2 sentences clearly describing what the vertices are and when two vertices are adjacent. If the graph is weighted, clearly specify what the edge weights are.]

Answer. In this graph I would have the vertices be a tuple that represents the state of each battery. The edges would represent the transfer of power from one battery to another. □

4.2.2 Problem 6(b)

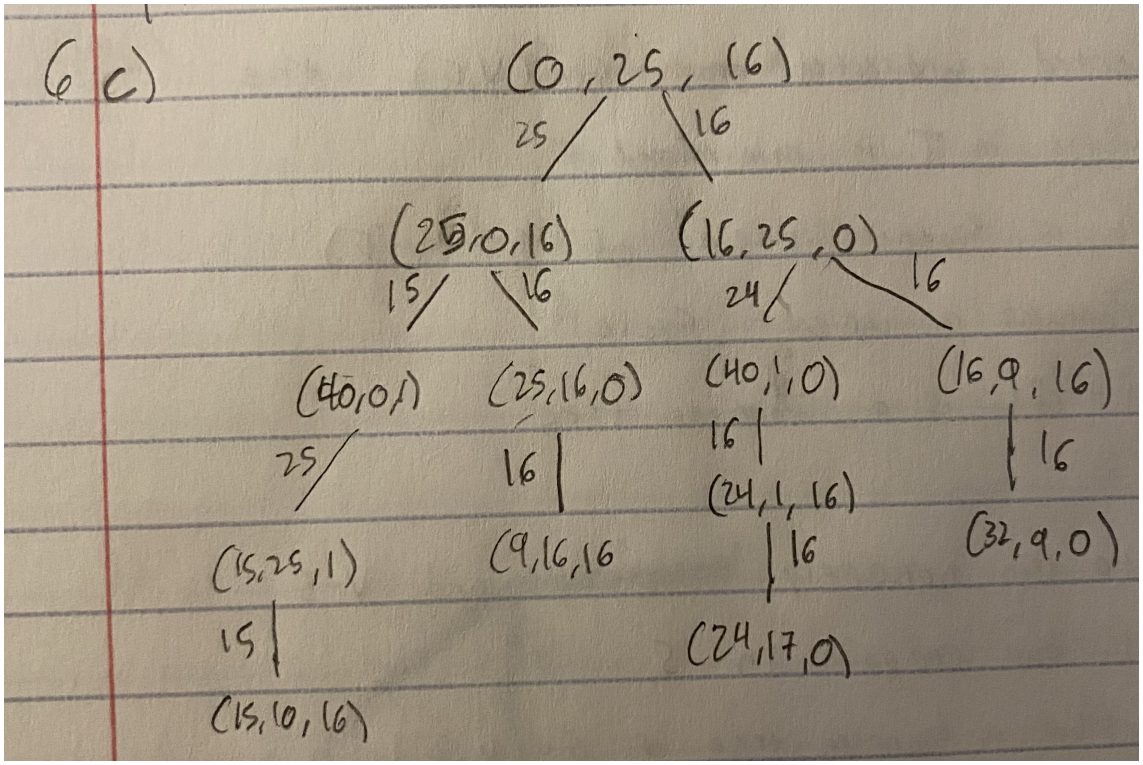
(b) Clearly describe an algorithm to solve this problem. If you use an algorithm covered in class, it is enough to state that. If you modify an algorithm from class, clearly outline any modifications. Make sure to explicitly specify any parameters that need to be passed to the initial function call.

Answer. To solve this problem we would run Dijkstra's algorithm on the tree to find the shortest path. Which in this tree would be the most optimal way to solve the question. □

4.2.3 Problem 6(c)

(c) Apply that algorithm to the question. Report and justify your answer. Here, justification includes the sequences of vertices visited and the total cost.

Answer.



This is the tree that results from the algorithim described in part b. On the bottom left we found the vertex (15,10,16) we can now back track along the edges and add their weights. 25+15+25+15=80 total cost. □