Problem Set 2
data structures and algorithms 2 - spring 2024

mst3k
due february 7, 2024 at 11:59p

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list your collaborators

Sources: list your sources

problem 1 *Negative weights*

Dijkstra's algorithm works for weighted connected graphs in which the weights are non-negative values. However, it does not always work when weights may have negative values. To help understand why this is so, create a counter-example: a weighted connected graph that has both positive and negative weight values such that Dijkstra's algorithm does not find a shortest path between two nodes. Show your graph, then explain why this algorithm does not always work with negative weights.

Solution:

problem 2 *Distance from Destination*

You are part of logistics planning for a major event at UVA. There are several locations that you could host your event, each near a UTS bus stop. Your primary concern is minimizing the amount of time that it takes for a bus passenger to get to the event. You have access to the UTS bus schedule that contains all bus routes and the time to get from one stop to the next. You read this information as a directed graph, with each bus stop as a node, directed edges between stops indicating the bus directions, and edge weights representing the time it takes a bus to get from one stop to the next stop.

Create an algorithm that can identify the most central bus stop; that is, the stop such that the longest path to reach it—in terms of time on the bus—from all other stops is minimized. You may ignore time spent waiting to transfer busses or waiting to catch the bus. Describe your algorithm and analyze its runtime.

Solution:

problem 3 *BFS and DFS Trees*

Consider the BFS tree T_B and the DFS tree T_D for the same graph G and same starting vertex s. In a few sentences, clearly explain why for every vertex v in G, the depth of v in the BFS tree cannot be greater than its depth in the DFS tree. That is:

$$\forall v \in G.V,\ depth(T_B, v) \leq depth(T_D, v)$$

(Here the depth of a node is the number of edges from the node to the tree's root node. Also, you can use properties of BFS and DFS that you've been taught in class. We're not asking you to prove those properties.)

Solution:

PROBLEM 4 *Orange and Blue*

A graph is two-colorable (i.e., bipartite) if each vertex can be assigned one of two colors, say orange or blue, such that for every edge in the graph (v_i, v_j) the vertices v_i and v_j do not have the same color.

Write an algorithm that takes a connected undirected graph $G = (V, E)$ and returns a list of edges E' than can be removed from G to make it two-colorable, i.e., the graph $G = (V, E - E')$ is two-colorable. If G is two-colorable to begin with, the list you return will be empty.

Solution: