Programming Question-5

Help Center

Warning: The hard deadline has passed. You can attempt it, but **you will not get credit for it**. You are welcome to try it as a learning exercise.

☐ **In accordance with the Coursera Honor Code, I (Alisa Zhou) certify that the answers here are my own work.**

Question 1

In this programming problem you'll code up Dijkstra's shortest-path algorithm.

Download the text file here. (Right click and save link as).

The file contains an adjacency list representation of an undirected weighted graph with 200 vertices labeled 1 to 200. Each row consists of the node tuples that are adjacent to that particular vertex along with the length of that edge. For example, the 6th row has 6 as the first entry indicating that this row corresponds to the vertex labeled 6. The next entry of this row "141,8200" indicates that there is an edge between vertex 6 and vertex 141 that has length 8200. The rest of the pairs of this row indicate the other vertices adjacent to vertex 6 and the lengths of the corresponding edges.

Your task is to run Dijkstra's shortest-path algorithm on this graph, using 1 (the first vertex) as the source vertex, and to compute the shortest-path distances between 1 and every other vertex of the graph. If there is no path between a vertex v and vertex 1, we'll define the shortest-path distance between 1 and v to be 1000000.

You should report the shortest-path distances to the following ten vertices, in order: 7,37,59,82,99,115,133,165,188,197. You should encode the distances as a comma-separated string of integers. So if you find that all ten of these vertices except 115 are at distance 1000 away from vertex 1 and 115 is 2000 distance away, then your answer should be 1000,1000,1000,1000,1000,2000,1000,1000,1000,1000. Remember the order of reporting

DOES MATTER, and the string should be in the same order in which the above ten vertices are given. Please type your answer in the space provided.

IMPLEMENTATION NOTES: This graph is small enough that the straightforward $O(mn)$ time implementation of Dijkstra's algorithm should work fine. OPTIONAL: For those of you seeking an additional challenge, try implementing the heap-based version. Note this requires a heap that supports deletions, and you'll probably need to maintain some kind of mapping between vertices and their positions in the heap.

☐ **In accordance with the Coursera Honor Code, I (Alisa Zhou) certify that the answers here are my own work.**

Submit Answers Save Answers

You cannot submit your work until you agree to the Honor Code. Thanks!