DSC 40B - "Super Homework"

Due: Thursday, March 18

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer.

⚠ Because of grading deadlines, we cannot accept slip days for the super homework.

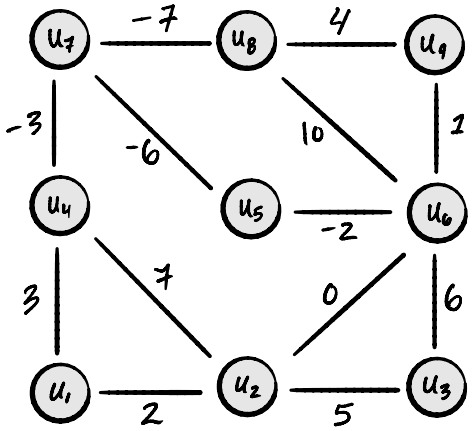
Problem 1. (2 Points Extra Credit)

Please fill out your CAPE for this class! The deadline for submitting CAPEs is Monday, March 15 at 8am PST (before this assignment is due). Submitting a CAPE is optional, but if do submit it **and say that you did so in your answer to this question**, we'll give you two points of extra credit towards this assignment.

Problem 2.

In this problem you will be asked to list the edges in the minimum spanning tree of the graph below in the order that they are added by either Prim's algorithm or Kruskal's algorithm.

In order to simplify grading, please write an edge with the *smaller* node first. For example: (u_3, u_7) instead of (u_7, u_3). Also, when writing an edge, make sure to write the edge as a pair of nodes, not the weight of the edge. Thanks!



a) Suppose Prim's algorithm is run on the above graph, using node u_1 as the starting node. List the edges of the resulting minimum spanning tree computed in the order that they are added by the algorithm.

b) Suppose Kruskal's algorithm is run on the graph above. List the edges of the resulting minimum spanning tree in the order that they are added by the algorithm.

Problem 3.

We have seen that Dijkstra's algorithm can be used to find shortest paths in weighted graphs with positive edge weights. It works by keeping a min-priority queue, and at every step visits the node with the next smallest priority and updates the estimated shortest paths to its neighbors.

Suppose that Dijkstra's algorithm is modified with the goal of computing the **longest** simple paths from the source to every other node. To do so, the min-priority queue is replaced with a max-priority queue, and the `update` procedure is changed so that it updates a node's estimated distance only when a *longer* path is found.

Does this algorithm work? That is, is it guaranteed to find the correct longest paths in general? You do not need to prove your answer, but you should give a short justification of why you believe the modification will work or not. You may assume that the graph has positive edge weights.

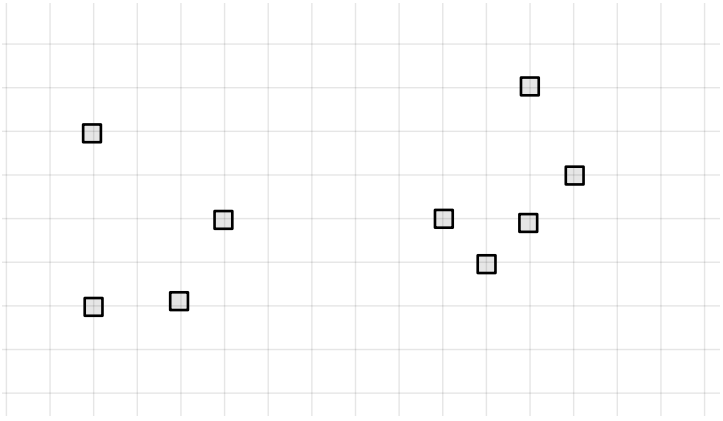
Hint: use ideas from the last lecture.

Problem 4.

A *maximum* spanning tree of a weighted graph G is a spanning tree of G with greatest total edge weight. Describe a simple algorithm for computing a maximum spanning tree.

Problem 5.

The picture below shows a set of points in 2-dimensional space. A grid is provided so that you can compute the distance between points; each grid cell is 1 unit wide and 1 unit tall. You may assume that each data point is placed on a grid intersection.



Suppose a weighted distance graph G is constructed from this data set. Then suppose that a minimum spanning tree is computed for G. What will be the weight of the largest edge in this minimum spanning tree?

Programming Problem 1.

In a file named `has_eulerian.py`, create a function `has_eulerian(graph)` which accepts as input an undirected, unweighted graph and returns `True` if the graph contains an Eulerian cycle, and `False` otherwise. Your code should take polynomial time.

Note: you may assume that the graph does not contain multiple edges between the same pair of nodes. That is, it is not a multigraph. You may assume that the graph has at least one node, and that there are no *isolated* nodes (that is, nodes with degree of zero).