Final project Design

① I am using a graph structure to document friendships between friends. Each node represents a person while each edge represents the "friendship level" between them (1-100). For the purpose of the shortest path algorithm, smaller the "level" the closer the friendship.

Thus, we can use the shortest path algorithm to see how close a person is to other people in the friendship circle/graph.

And we can use the minimum spanning tree to see the closest friendships that connects everyone together.

② create some tests for each functionality

- Add new vertex

① Don't add vertex yet. check that vertex does not exist.

② Add vertex to graph. check that vertex does exist.

- Add new edge

① Don't add edge yet check that edge does not exist.

② Add edge to graph check that edge exists.

- shortest path

① test on empty graph

② test on simple graph
- compare to expected

③ test on little more complex graph → compare to expected

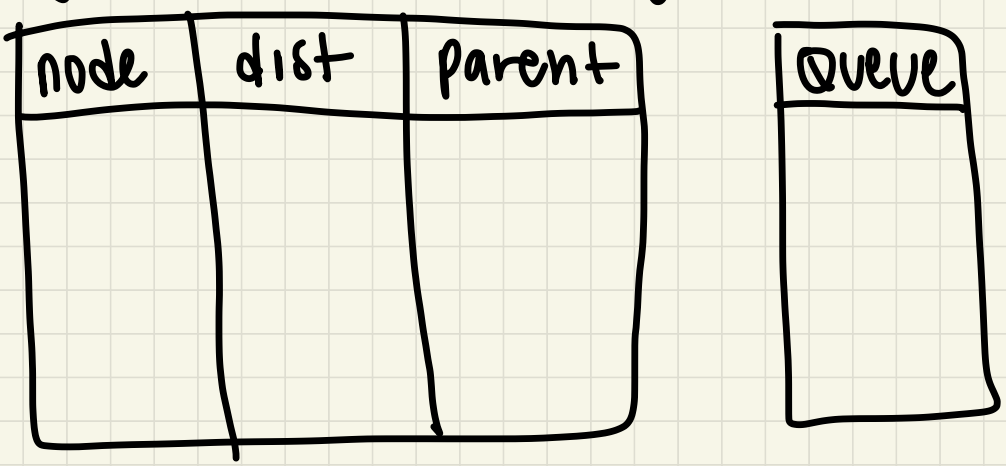
- minimum spanning tree

① test on empty graph

② test on simple graph
- compare to expected

③ test on little more complex graph → compare to expected

My shortest path Algorithm



- start with 4 vectors
  node, distance, parent, Queue
- size of distance & parent is the size of the nodes list.
  - default dist is 100000 (INF.)
  and default parent is nullptr.

- we check for any isolated nodes (keeping count of them.
- while true, we start from the current node and put its children in the queue.
  - if the distance from current node to first item in queue plus the current's distance is smaller than the distance for the item in queue, update dist and change parent to current. Add the queue node to node that has been checked
  - repeat for all items in queue
  - clear queue
  - new current is the next item in checked nodes. Repeat above process until all connected nodes are checked.
  - store info in a string & return

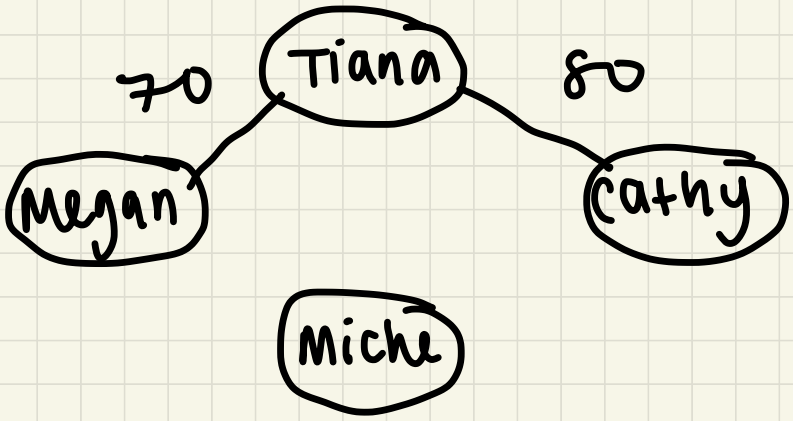
My Minimum spanning Tree Algorithm

- take all the edges of graph and sort them from smallest to largest.
- take smallest edge, add to MST edges vector
- keep taking the next smallest edge, only add to MST if the new edge doesn't create a cycle.

Test Graphs from Driver

For shortest path:

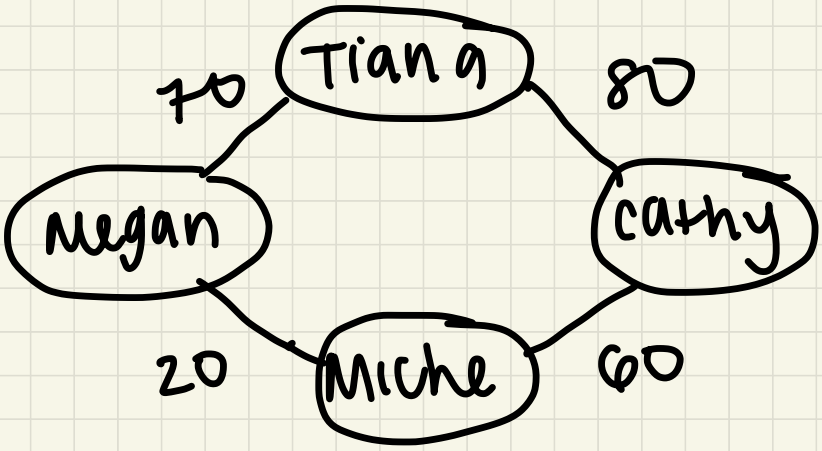
① simple case



Expected output for shortest path from Tiana

node	dist	parent
Tiana	0	—
Megan	70	Tiana
Cathy	80	Tiana

(2) (a little) complex case

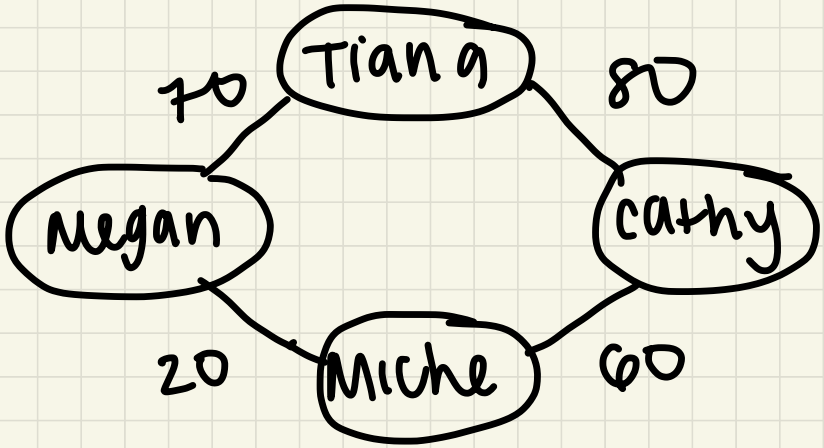


Expected output for shortest path from Tiana

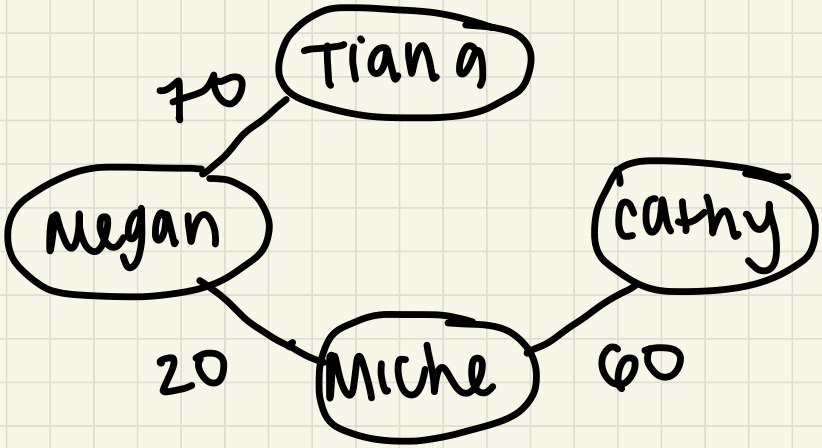
node	dist	parent
Tiana	0	—
Megan	70	Tiana
cathy	80	Tiana
Miche	90	Megan

For MST

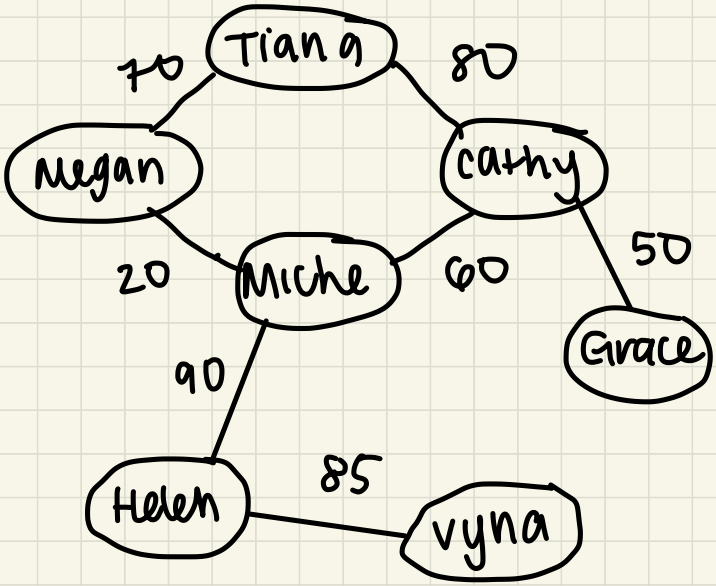
① simple case:



Expected output:



(2) (a little) complex case:



Expected output