Problem type 1:

Answer the problem below:

(See variants below)

P.S. You may **not** use lab solutions as a black box.

a. **BYA**

For what types of graphs does the Bellman-Ford algorithm have roughly the same runtime as the Floyd-Warshall Algorithm.

Solution: When the graph is fully connected ($m = n^2$). ■

b. **BYE**

You are given a graph with all *positive* edge weights and want to find the shortest distance between all pairs of nodes. Give an algorithm (as fast as possible) that would give you this information.

Solution: Optimal solution: Run Djikstra's algorithm n time.
Supoptimal solution (1 point): Floyd-Warshall algorithm.
Very suboptimal solution (0.5 points): Bellman-Ford algorithm n times. ■

c. **BYB**

The Floyd-Warshall algorithm is described by the recurrence:

$$dist(i,j,k) = \min \begin{cases} dist(i,j,k-1) \\ dist(i,k,k-1) + dist(k,j,k-1) \end{cases}$$

In a english sentence, state what the value $dist(i,j,k)$ represents.

Solution: $dist(i,j,k)$ represents the minimum distance between vertices i and j using only vertices $1 \dots k$. ■

d. **BYH**

The Bellman-Ford algorithm is described by the recurrence:

$$d(v,k) = \min \begin{cases} \min_{u \in V}(d(u,k-1) + \ell(u,v)). \\ d(v,k-1) \end{cases}$$

In a english sentence, state what the value $d(v,k)$ represents.

Solution: $d(v,k)$ represents the minimum distance from the start vertex s to v using at most k edges. ■

e. **BYG**

Prof. Nani had a great idea to solve the longest path problem (given a graph G, find the longest path between any two vertices). He would simply make all the edges negative and run the Floyd-Warshall algorithm. Why does this not work?

Solution: It would introduce a whole bunch of negative cycles into the graph. ■

f. **BYC**

Prof. Nani had a great idea to solve the shortest path problem using Djikstra's algorithm. He would simply modify the graph by finding the smallest (most-negative) edge and adding the magnitude of that edge to all the edges weights in the graph. This ensure that all the edge weights are positive which is all that's required for Djikstra's algorithm right? Why does this not work?

Solution: Doing this penalizes paths with more edges than paths with fewer edges and will likely cause incorrect results. ■

g. **BYF**

Suppose you are given a directed, weighted graph with positive and negative edge weights and you want to see if there is a negative edge cycle present in the graph. Describe a efficient algorithm that returns true if there is a negative cycle.

Solution: Multiple solutions:

- Run Bellman-Ford for n rounds. This only works if the cycle is reachable from s so you need to specify that you will augmen tthe graph to include a extra node that can reach all the other nodes and then run BF from there.
- Run the Floyd-Warshall algorihm for n hops. If the value change the last round, there is a negative cycle.

■

h. **BYD**

You are given a directed graph with unweighted edges. Some of the edges are marked as risky and **you can't take more than one** risky edge. Provide algorihtm (as fast a possible) for the shortst path between s and t.

Solution:
Construct a new graph G' where $V' = V \times \{0,1\}$ and edges

$$E' = \left\{(u^{0/1}, v^{0/1}) \text{ for non-risky } (u,v) \in V\right\} + \left\{(u^0, v^1) \text{ for risky } (u,v) \in V\right\}$$

. Then run $BFS(G', s^0)$ with layering and return $layer(min(t^0, t^1))$. ■