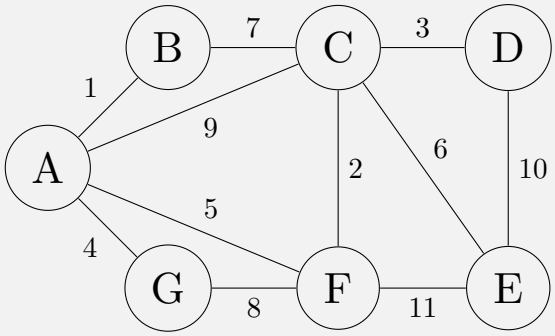
Problem 0 (Course Evaluation, 0 points)

If you haven't done so already, please fill out the course evaluation at `https://go.blueja.io/c74Z6Qe6T0G4GxKD_gfpgA` or by scanning the QR code below.

Problem 1 (MST Algorithms, 20 pts)

Consider the following undirected graph $G = (V, E)$.



(a) Illustrate a run of Kruskal's algorithm on this graph by filling in the table below. State at each step which edge is added to the tree. We have filled the first step in for you. Here, **we only want the edge added to the tree and not every edge that is considered.**

Step	1	2	3	4	5	6
Edge Added	AB					

(b) Illustrate a run of Prim's algorithm on this graph starting from vertex A by filling in the table below. State at each step which edge is added to the tree. We have filled the first step in for you. Here, **we only want the edge added to the tree and not every edge that is considered.**

Step	1	2	3	4	5	6
Edge Added	AB					

Problem 2 (Pizza Delivery, 30 points)

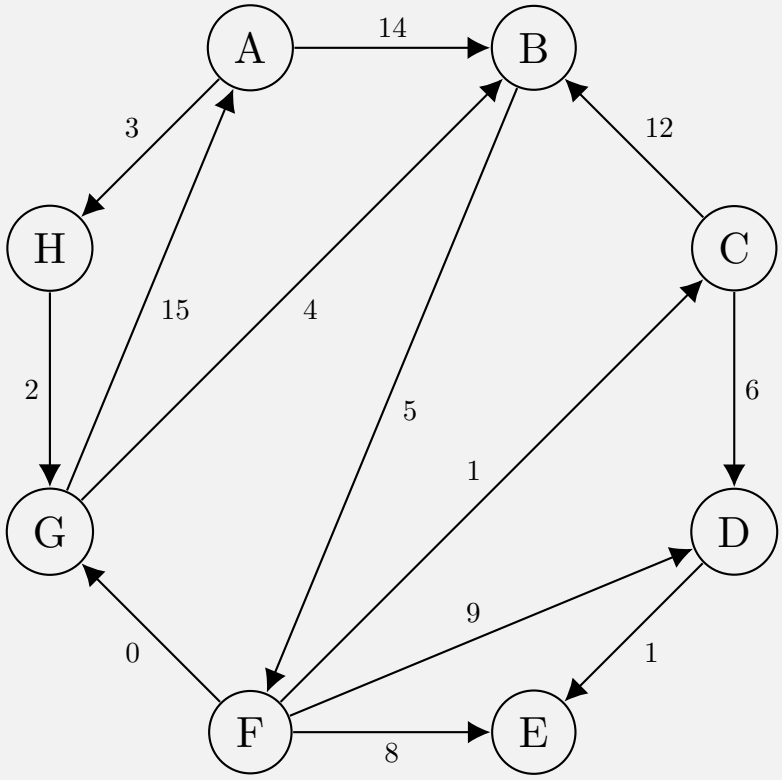
Alice, who lives at a vertex s of a directed, weighted graph $G = (V, E)$ (with non-negative weights), is going to her friend's house at vertex h for dinner. Naturally, Alice wants to get from s to h as soon as possible, but it appears that her friend is terrible at cooking, so along the way she wants to get a pizza from a pizza store just in case her friend cooks something inedible. Let's say the pizza stores form a subset of the vertices $B \subset V$. Thus, starting at s, Alice must go to some vertex $b \in B$ of her choice, and then head from b to h using the shortest overall route possible (assuming she wastes no time at the pizza store). We can help Alice reach h as soon as possible, by solving the following sub-problems.

1. Compute the shortest distance from s to all pizza stores $b \in B$.
2. Compute the shortest distance from every pizza store $b \in B$ to h. Note that this is the dual of the single-source shortest path where we are now asking for the shortest path from every node to a particular destination.
3. Combine part 1 and 2 to solve the full problem.

A straightforward solution is to run Dijkstra's algorithm twice (once in part 1 and once in part 2). In this problem, you will improve this solution by running Dijkstra's algorithm only **once**. Specifically, you should define a new graph G' on $2|V|$ vertices and at most $2|E| + |V|$ edges (and appropriate weights for these edges), so that the original problem can be solved using a single Dijkstra call on G'. Briefly argue correctness of your proposed solution. (*Hint: This is another problem on reductions. You might want to refresh your memories on Problem 2 in Homework 8, which is also a reduction problem.*)

Problem 3 (Dijkstra Practice, 25 points)

Show the running of Dijkstra's algorithm for finding the distance from A to all other vertices in the following graph. You should show the updated distances to all the vertices (i.e., the key of the vertex in the priority queue) after each step in the algorithm.



For the ease of representing your solution, simply complete the following table. The first row (Step 0) represents the initial values.

	Vertex extracted from PQ	dist(A)	dist(B)	dist(C)	dist(D)	dist(E)	dist(F)	dist(G)	dist(H)
0	-	0	∞	∞	∞	∞	∞	∞	∞
1	A								
2									
3									
4									
5									
6									
7									
8									

Problem 4 (Detecting Negative-Weight Cycles, 25 points)

In lecture we mentioned why it does not make sense to consider shortest paths on a graph with negative-weight cycles, but how do we detect if a graph contains such a cycle? Recall that a negative-weight cycle is defined as a cycle where the sum of the weights of the edges in the cycle is negative. Show how one can use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle in a directed graph. Briefly argue the correctness of your algorithm.