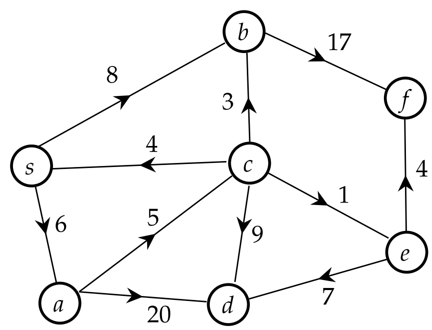
INSTRUCTIONS:

1. Include the name and PSU access ID of every member in your group in your solution.
2. Submit your solution to Gradescope. Make sure only one of your group members submits. After submitting, make sure to add your group members.
3. Your always need to explain the running time of your algorithm.

Problem 1 (10 points).

Run Dijkstra's algorithm (refer to Lecture 18) on the instance given below.

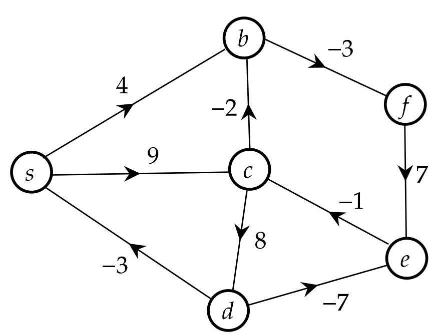
1. For each iteration: (1) show which vertex is removed from the priority queue; (2) show the content of *dist* array at the end of this iteration; (3) show the content of *prev* array at the end of this iteration.
2. Draw the shortest-path tree using the final status of the *prev* array.



Problem 2 (10 points).

Run the dynamic programming algorithm for the shortest path problem (refer to Lecture 20) on the instance given below.

1. Give the dynamic programming table, including the *prev* value for each entry.
2. Draw the shortest-path tree (using *prev* in the final row).



Problem 3 (14 points).

Suppose that you have a directed graph $G = (V, E)$ where each vertex is labeled with a digit in $\{0, 1, \cdots, 9\}$. Given two vertices $s, t \in V$, decide if there is a path from s to t in which the vertex labels follow the pattern 465465465... (It does not necessarily need to end at 5.) For example, if in a graph there exists a path $s \to v_0 \to v_1 \to v_2 \to v_3 \to t$, and the labels for s is 4, v_0 is 6, v_1 is 5, v_2 is 4, v_3 is 6, t is 5, then this path follows the desired pattern.

1. Design an algorithm that can solve this problem. Your algorithm should run in $O(|V| + |E|)$ time.

2. Suppose now that the labels are on the edges instead of vertices. Design an efficient algorithm that decides whether there is a *walk* (i.e., a path which allows same vertex or edge being used multiple times) from s to t in which the edges labels follow the pattern 465465465... For example, consider graph $G = (V, E)$ where $V = (s, v_0, t)$, $E = ((s, v_0, 4), (v_0, s, 6), (s, t, 5))$, then there is a walk: $s \to v_0 \to s \to t$ follows the pattern. Your algorithm should run in $O(|V| + |E|)$ time.

Problem 4 (10 points).

Let $G = (V, E)$ be a directed graph with possibly negative edge length, but without negative cycle. Let $a \in V$. Define $distance_a(u, v)$ as the length of the shortest path from u to v that goes through vertex a. Design an algorithm to calculate $distance_a(u, v)$ for all pairs of vertices in G. Your algorithm should run in $O((|V| + |E|) \cdot |V|)$ time.

Problem 5 (10 points).

You are given a directed graph (V, E) with positive edge length $l(e)$ for any $e \in E$, and a positive weight $w(v)$ for any $v \in V$. Now we define the length of a path p from u to v as the sum of the lengths of all the edges in p plus the sum of the weights of all the vertices in p. You are also given a source $s \in V$ and it happens that $w(s) = 0$. Design an algorithm to find the length of the shortest path (in this new definition) from s to all vertices in V. Your algorithm should run in $O((|V| + |E|) \log |V|)$ time.

Problem 6 (10 points).

You are given a directed graph $G = (V, E)$ with possibly negative edge length but without negative cycle, and $s \in V$. You may assume that s can reach all vertices in V. Describe how to modify the dynamic programming algorithm introduced in lecture so that it also sets a binary array *multiple* of size $|V|$ indexed by $v \in V$, where $multiple[v] = 1$ if there are two or more different shortest paths from s to v while $multiple[v] = 0$ if the shortest path from s to v is unique. Your algorithm should run in $O(|V| \cdot |E|)$ time.

Problem 7 (10 points).

You are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 < r(u, v) < 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and we assume that these probabilities are independent over all edges. Given two vertices $s, t \in V$, design an algorithm in $O((|E| + |V|) \log |V|)$ time to find the most reliable path from s to t.