Problem 1*. Let $G = (V, E)$ be a weighted directed acyclic graph. Given a source vertex $s \in V$, design an algorithm to find the shortest path distances from $s$ to the vertices in $V$. Your algorithm should terminate in $O(|V| + |E|)$ time.

Solution. First run DFS on $G$ to obtain a topological order of $V$. For each $v \in V$, initialize a value $\text{dist}(v)$ which equals 0 if $v = s$, and $\infty$ otherwise. Now, process the vertices of $V$ according to the topological order. Specifically, processing a vertex $u$ means relaxing all the out-going edges $(u,v)$ of $u$. After every vertex has been processed, the final $\text{dist}(v)$ is the shortest path distance from $s$ to $v$, for every $v \in V$.

To prove this is correct, recall that (as discussed earlier in the lecture) the shortest-path distances $\text{spdist}(s,v)$ from $s$ to $v \in V$ satisfy:

$$\text{spdist}(s,v) = \min_{u \in \text{IN}(v)} \text{spdist}(s,u) + w(u,v)$$

where $w(u,v)$ denotes the weight of the edge $(u,v)$, and $\text{IN}(v)$ is the set of in-neighbors of $v$. The correctness of our algorithm thus follows from:

**Claim:** At the moment right before $v$ is processed, $\text{spdist}(u)$ has already been computed for every $u \in \text{IN}(v)$.

The above claim can be easily established by induction on the number of edges in a shortest path.

Problem 2. Let $G = (V, E)$ be a weighted directed graph where the weight of an edge $(u,v)$ is $w(u,v)$. It is guaranteed that $G$ has no negative cycles. Prove: the following is a correct implementation of Bellman-Ford’s algorithm:

**Algorithm Bellman-Ford**

1. Pick an arbitrary vertex $s \in V$
2. Set $\lambda$ to the sum of all the positive edge weights in $G$
3. Initialize $\text{dist}(s) = 0$ and $\text{dist}(v) = \lambda$ for every other vertex $v \in V$
4. For $i = 1$ to $|V| - 1$
5. Relax all the edges in $E$
6. Return $\text{dist}(v)$ for all $v \in V$

Remark: Compared to the description in our lecture notes, the key difference here is that, at Line 3, we initialize $\text{dist}(v)$ as $\lambda$, instead of $\infty$.

Solution. Follows directly from the fact that, to every vertex $v \in V$, $s$ has a shortest path that is a simple path. Notice that every simple path has a length at most $\lambda$.

Problem 3*. Let $G = (V, E)$ be a weighted directed graph where the weight of an edge $(u,v)$ is $w(u,v)$. Prove: the following algorithm correctly decides whether $G$ has a negative cycle:

**Algorithm negative-cycle-detection**

1. Pick an arbitrary vertex $s \in V$
2. Set $\lambda$ to the sum of all the positive edge weights in $G$
3. initialize $dist(s) = 0$ and $dist(v) = \lambda$ for every other vertex $v \in V$
4. for $i = 1$ to $|V| - 1$
5. relax all the edges in $E$
6. for each edge $(u, v) \in E$
7. if $dist(v) > dist(u) + w(u, v)$ then
   return “there is a negative cycle”
8. return “no negative cycles”

Solution. We will prove two directions.

Direction 1: If the inequality of Line 6 holds for any edge $(u, v)$, then there must be a negative cycle. In the lecture we proved that, in the absence of negative cycles, Bellman-Ford’s algorithm correctly finds all shortest path distances (from $s$) after $|V| - 1$ rounds of edge relaxations. This (together with the result of Problem 2) indicates that, if there are no cycles, when we come to Line 5 the value $dist(v)$ must be the final shortest path distance for every $v \in V$. If Line 6 holds for some edge $(u, v)$, however, it means that an even shorter path from $s$ to $v$ has just been discovered. Therefore, in such a case, $G$ must contain a negative cycle.

Direction 2: If there is a negative cycle, then the inequality of Line 6 must hold for at least one edge $(u, v)$. Suppose that the negative cycle is $v_1 \to v_2 \to \ldots \to v_{\ell} \to v_1$. Hence:

$$w(v_{\ell}, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) < 0. \quad (1)$$

Assume that Line 6 does not hold on any edge in $E$. This indicates:

- for every $i \in [1, n]$, $dist(v_{i+1}) \leq dist(v_i) + w(v_i, v_{i+1})$;
- $dist(v_1) \leq dist(v_n) + w(v_n, v_1)$.

These two bullets lead to:

$$\sum_{i=1}^{\ell} dist(v_i) \leq \left( \sum_{i=1}^{\ell} dist(v_i) \right) + w(v_{\ell}, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1})$$
$$\Rightarrow 0 \leq w(v_{\ell}, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1})$$

which contradicts (1).

Problem 4. In our lecture about the Floyd-Warshall algorithm, we have given the following recursive function:

$$spdist(i, j | \leq k) = \min \left\{ spdist(i, j | \leq k - 1), spdist(i, k | \leq k - 1) + spdist(k, j | \leq k - 1) \right\}$$

Give the details of computing $spdist(i, j)$ for all $i, j \in [1, n]$ in $O(n^3)$ time.

Solution.

algorithm Floyd-Warshall
1. for all $i, j \in [1, n]$
2. set \( spdist(i, j | \leq 0) = 0 \) if \( i = j \) or \( \infty \) otherwise
3. for \( k = 1 \) to \( n \)
4. for all \( i, j \in [1, n] \)
5. set \( spdist(i, j | \leq k) \) according to the recursive function

**Problem 5.** Augment your algorithm for the previous problem to compute the shortest path between vertex \( i \) and vertex \( j \), for all \( i, j \in [1, n] \).

**Solution.**

algorithm Floyd-Warshall
1. for all \( i, j \in [1, n] \)
2. set \( spdist(i, j | \leq 0) = 0 \) if \( i = j \) or \( \infty \) otherwise
3. set \( bestchoice(i, j) = \text{nil} \)
4. for \( k = 1 \) to \( n \)
5. for all \( i, j \in [1, n] \)
6. if \( spdist(i, j | \leq k - 1) \leq spdist(i, k - 1 | \leq k - 1) + spdist(k - 1, j | \leq k - 1) \) then
7. \( spdist(i, j | \leq k) = spdist(i, j | \leq k - 1) \)
8. else
9. \( spdist(i, j | \leq k) = spdist(i, k - 1 | \leq k - 1) + spdist(k - 1, j | \leq k - 1) \)
10. \( bestchoice(i, j) = k \)

The function \( bestchoice(.,.) \) computed by the above algorithm encodes all the shortest paths. Specifically, for any \( i, j \in [1, n] \) such that \( i \neq j \):

- if \( bestchoice(i, j) = \text{nil} \), the shortest path from \( i \) to \( j \) consists of just the edge \((i, j)\);
- if \( bestchoice(i, j) = k \), the shortest path concatenates the shortest path from \( i \) to \( k \) and the shortest path from \( k \) to \( j \) — note that the latter two shortest paths can be obtained recursively in the same manner.