Problem 1*. Prove the correctness of Dijkstra’s algorithm (when the edges have non-negative weights).

Solution. We argue that, every time a vertex $v$ is removed from $S$, we must have $\text{dist}(v) = \text{spdist}(v)$. We will do so by induction on the order that the vertices are removed. The base step, which corresponds to removing the source vertex $s$, is obviously correct. Next, assuming correctness on all the vertices already removed, we will prove the statement on the vertex $v$ removed next.

Let $\pi$ be an arbitrary shortest path from $s$ to $v$. Identify the last vertex $u$ on $\pi$ such that $\text{spdist}(u) = \text{spdist}(v)$. In other words, all the edges on $\pi$ between $u$ and $v$ have weight 0. Let $\pi'$ be the prefix of $\pi$ that ends at $u$. Note that $\pi'$ must be a shortest path from $s$ to $u$.

Claim 1: When $v$ is to be removed from $S$, all the vertices on $\pi'$ — except possibly $u$ — must have been removed from $S$.

Proof of Claim 1: Suppose that the claim is not true. Define $v_{\text{bad}}$ as the first vertex on $\pi'$ that is still in $S$ when $v$ is to be removed from $S$. Let $v_{\text{good}}$ be the vertex right before $v_{\text{bad}}$ on $\pi$; note that $v_{\text{good}}$ definitely exists because $v_{\text{bad}}$ cannot be $s$. By how $u$ is defined, we must have $\text{spdist}(v_{\text{bad}}) < \text{spdist}(u) = \text{spdist}(v)$.

By our inductive assumption, when $v_{\text{good}}$ was removed from $S$, we had $\text{dist}(v_{\text{good}}) = \text{spdist}(v_{\text{good}})$. We must have relaxed the edge $(v_{\text{good}}, v_{\text{bad}})$, after which we must have

\[ \text{dist}(v_{\text{bad}}) = \text{dist}(v_{\text{good}}) + w(v_{\text{good}}, v_{\text{bad}}) = \text{spdist}(v_{\text{good}}) + w(v_{\text{good}}, v_{\text{bad}}) = \text{spdist}(v_{\text{bad}}). \]

The value $\text{dist}(v_{\text{bad}})$ never increases during the algorithm. Hence, when $v$ is to be removed from $S$, we must have $\text{dist}(v_{\text{bad}}) = \text{spdist}(v_{\text{bad}}) < \text{spdist}(u) = \text{spdist}(v) \leq \text{dist}(v)$. But this contradicts the fact that $v$ has the smallest $\text{dist}$-value among all the vertices still in $S$. \hfill \Box

Consider the moment when $v$ is to be removed from $S$; define $z$ as the first vertex on $\pi$ that has not been removed from $S$. Note that $z$ is well defined because $v$ itself is still in $S$ at this moment.
Claim 2: When \( v \) is to be removed from \( S \), \( \text{dist}(z) = \text{spdist}(z) \).

Proof of Claim 2: Let \( z' \) be the vertex right before \( z \) on \( \pi \). Note that \( z' \) is well defined because \( z \) cannot be earlier than \( u \) on \( \pi \) (Claim 1) and \( z \) cannot be \( s \).

By our inductive assumption, when \( z' \) was removed from \( S \), we had \( \text{dist}(z') = \text{spdist}(z') \). We must have relaxed the edge \((z', z)\), after which we must have
\[
\text{dist}(z) = \text{dist}(z') + w(z', z) = \text{spdist}(z') = \text{spdist}(z).
\]

It now follows that, when \( v \) is to be removed from \( S \), we have \( \text{dist}(v) \leq \text{dist}(z) = \text{spdist}(z) \). As \( \text{dist}(v) \) cannot be larger than \( \text{spdist}(v) \), we must have \( \text{dist}(v) = \text{spdist}(v) \).

Problem 2. Consider again your proof for Problem 1. Point out the place that requires edge weights to be non-negative.

Solution. We used this assumption in the proof of Claim 1: look for the sentence: “By how \( u \) is defined, we must have \( \text{spdist}(v_{\text{bad}}) < \text{spdist}(u) = \text{spdist}(v) \).”

Problem 3* (SSSP in a DAG). Consider a simple acyclic directed graph \( G = (V, E) \) where each edge \( e \in E \) has an arbitrary weight \( w(e) \) (which can be negative). Solve the SSSP problem on \( G \) in \( O(|V| + |E|) \) time.

Solution. Let \( s \) be the source vertex. For each vertex \( v \in V \), define \( \text{spdist}(v) \) as the shortest path length from \( s \) to \( v \). Also, define \( \text{IN}(v) \) as the set of in-neighbors of \( v \). Observe that:
\[
\text{spdist}(v) = \begin{cases} 
0 & \text{if } v = s \\
\infty & \text{if } \text{IN}(v) = \emptyset \\
\min_{u \in \text{IN}(v)}(\text{spdist}(u) + w(u, v)) & \text{if } v \neq s \text{ and } \text{IN}(v) \neq \emptyset
\end{cases}
\]

We can compute \( \text{spdist}(v) \) in \( O(|V| + |E|) \) time based on a topological order of \( V \), which can also be obtained in \( O(|V| + |E|) \) time (see Prof. Tao’s CSCI2100 homepage). The shortest path tree of \( s \) can then be obtained using the piggyback technique without increasing the time complexity.

Problem 4. Let \( G = (V, E) \) be a simple directed graph where each edge \( e \in E \) carries a weight \( w(e) \), which can be negative. It is guaranteed that \( G \) has no negative cycles. Prove: given any vertices \( s, t \in V \), at least one shortest path from \( s \) to \( t \) is a simple path (i.e., no vertex appears twice on the path).

Solution. Consider a shortest path \( \pi \) from \( s \) to \( t \) that has the least number of edges. We argue that \( \pi \) must be simple. Otherwise, at least one vertex \( v \) appears twice on \( \pi \). Identify any two consecutive occurrences of \( v \) — call the first occurrence \( v_1 \) and the second \( v_2 \). Thus, the subpath of \( \pi \) from \( v_1 \) to \( v_2 \) is a cycle. As \( G \) does not have any negative cycle, that subpath must have a non-negative
length. We can now remove the subpath from $\pi$ to obtain another shortest path from $s$ to $t$ that has fewer edges than $\pi$.

**Problem 5**. Let $G = (V, E)$ be a simple 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 the absolute weights of all edges in $G$
3. initialize $\text{dist}(s) = 0$ and $\text{dist}(v) = 2\lambda$ 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 $\text{dist}(v) > \text{dist}(u) + w(u, v)$ then
8. return "there is a negative cycle"
9. return "no negative cycles"
```

**Solution.** We will prove two directions.

**Direction 1:** If the inequality of Line 7 holds for any edge $(u, v)$, then there must be a negative cycle. The lecture 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 means that, if there are no cycles, when we come to Line 6, the value $\text{dist}(v)$ must be the shortest path distance from $s$ to $v$, for every $v \in V$ (think: for each $v \in V$, we initialized $\text{dist}(v)$ to $2\lambda$, rather than $\infty$; how does it affect the shortest path distances?). If Line 7 holds for some edge $(u, v)$, however, it means that an even shorter path from $s$ to $v$ has just been discovered. Therefore, $G$ must contain a negative cycle.

**Direction 2:** If there is a negative cycle, then the inequality of Line 7 must hold for at least one edge $(u, v)$. Suppose that the negative cycle is $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_\ell \rightarrow v_1$. 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 $i \in [1, \ell]$, $\text{dist}(v_{i+1}) \leq \text{dist}(v_i) + w(v_i, v_{i+1})$;
- $\text{dist}(v_1) \leq \text{dist}(v_\ell) + w(v_\ell, v_1)$.

These two bullets lead to:

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

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

which contradicts (1).