Suppose that $G = (V, E)$ is a simple directed graph where each edge $(u, v) \in E$ has a weight $w(u, v)$, which can be negative. It is known that $G$ is strongly connected and contains at least one negative cycle. In the tutorial, we learned the following algorithm for finding a negative cycle:

algorithm negative-cycle-detection
input: strongly connected $G = (V, E)$ and weight function $w$

1. $s \leftarrow$ arbitrary vertex in $V$
2. $\text{dist}(s) \leftarrow 0$ and $\text{dist}(v) \leftarrow \infty$ for every vertex $v \in V \setminus \{s\}$
3. $\text{parent}(v) \leftarrow \text{nil}$ for all $v \in V$
4. for $i \leftarrow 1$ to $|V| - 1$ do
5. for each edge $(u, v) \in E$ do
6. if $\text{dist}(v) > \text{dist}(u) + w(u, v)$ then
7. $\text{dist}(v) \leftarrow \text{dist}(u) + w(u, v)$; $\text{parent}(v) \leftarrow u$
8. for each edge $(u, v) \in E$ do
9. if $\text{dist}(v) > \text{dist}(u) + w(u, v)$ then
10. $\text{parent}(v) \leftarrow u$
   /* start tracing back the parent pointers until seeing a vertex twice */
11. initialize a vertex sequence $S$ that contains only $v$
12. while $\text{parent}(v) \notin S$ do
13. append $\text{parent}(v)$ to $S$; $v \leftarrow \text{parent}(v)$
14. report a negative cycle: output the appendix of $S$ starting from $v$ and add $v$ in the end

Next, we prove that the algorithm is correct.

**Lemma 1.** During the algorithm, if $u$ is a vertex in $V$ with $\text{parent}(u) \neq \text{nil}$, then $\text{dist}(\text{parent}(u)) + w(\text{parent}(u), u) \leq \text{dist}(u)$.

*Proof.* Let $z = \text{parent}(u)$. When $z$ just becomes $\text{parent}(u)$, $\text{dist}(z) + w(z, u) = \text{dist}(u)$. After that, $\text{dist}(z)$ can only decrease, while $\text{dist}(u)$ stays the same until $\text{parent}(u)$ is updated. $\Box$

**Lemma 2.** Suppose that there is a sequence of $x \geq 2$ vertices $u_1, u_2, \ldots, u_x$ such that $\text{parent}(u_i) = u_{i+1}$ for every $i \in [1, x - 1]$ and $\text{parent}(u_x) = u_1$. Then, $(u_1, u_x), (u_2, u_1), (u_3, u_2), \ldots, (u_x, u_{x-1})$ form a negative cycle.

*Proof.* Each of $\text{parent}(u_1), \text{parent}(u_2), \ldots, \text{parent}(u_x)$ was set by an edge relaxation. W.l.o.g., suppose that the edge relaxation for $\text{parent}(u_1)$ happened the latest. Consider the moment right before the relaxation. At this moment, we must have

$$\text{dist}(u_2) + w(u_2, u_1) < \text{dist}(u_1)$$

By Lemma 1, we have

$$\text{dist}(u_3) + w(u_3, u_2) \leq \text{dist}(u_2)$$
$$\text{dist}(u_4) + w(u_4, u_3) \leq \text{dist}(u_3)$$
$$\ldots$$
Let \( d \) gives a contradiction, indicating that our initial assumption (that the lemma is wrong) cannot be.

The above inequalities imply \( w(u_x, u_1) + \sum_{i=1}^{\ell} w(u_i, u_{i+1}) < 0 \).

**Lemma 3.** Consider the moment when the algorithm has come to Line 11. At this moment, if we continuously trace the parent pointers starting from \( v \), we encounter an infinite loop.

**Proof.** Suppose that this is not true. Then, the tracing must stop at \( s \) because every node — except possibly \( s \) — has a parent. This yields a simple path \( \pi \) from \( s \) to \( v \). Denote by \( \ell \) the number of edges on \( \pi \); clearly, \( \ell \leq |V| - 1 \). Denote the vertices on \( \pi \) as \( z_0, z_1, \ldots, z_{\ell} \), where \( z_0 = s \) and \( z_{\ell} = v \).

Let \( d_i \) be the value of \( \text{dist}(z_i) \) at this moment, for each \( i \in [0, \ell] \). Let us make several observations:

- \( \text{parent}(z_i) = z_{i-1} \) for all \( i \in [1, \ell] \), but \( \text{parent}(z_0) = \text{nil} \).
- The fact \( \text{parent}(s) = \text{nil} \) implies \( d_0 = 0 \). To see why, recall that \( \text{dist}(s) \) is set to 0 at the beginning of the algorithm. Thus, if \( d_0 \neq 0 \), then \( \text{dist}(s) \) must have been decreased during the algorithm’s execution, in which case \( \text{parent}(s) \) cannot be \( \text{nil} \).
- For each \( i \in [1, \ell] \), \( d_i \geq d_{i-1} + w(z_{i-1}, z_i) \). At the moment when \( \text{dist}(z_i) \) was reduced to \( d_i \) (which must be due to the relaxation of \( (z_{i-1}, z_i) \)), it held that \( d_i = \text{dist}(z_i) = \text{dist}(z_{i-1}) + w(z_{i-1}, z_i) \). The value of \( \text{dist}(z_{i-1}) \) could then only decrease after that, which implies \( d_i \geq d_{i-1} + w(z_{i-1}, z_i) \).

**Claim:** For each \( i \in [1, \ell] \), we have

- \( d_i = \sum_{i=1}^{\ell} w(z_{i-1}, z_i) \), and
- the value of \( \text{dist}(z_i) \) was exactly \( d_i \) at the end of the \( i \)-th round (and hence has remained so till the end of the algorithm).

We will prove the claim by induction. For the base case, the claim becomes \( \text{dist}(z_1) = d_1 = w(s, z_1) \) at the end of the first round. Right after the edge \((s, z_1)\) was relaxed in the first round, it held that \( \text{dist}(z_1) = w(s, z_1) \). In the rest of the algorithm, \( \text{dist}(z_1) \) could only decrease, indicating that \( d_1 \leq w(s, z_1) \). On the other hand, as observed earlier, we have \( d_1 \geq w(s, z_1) \). Therefore, it must hold that \( d_1 = w(s, z_1) \), and the value of \( \text{dist}(z_1) \) was \( w(s, z_1) \) at the end of the first round.

Assuming the claim’s correctness for \( i \leq k \), next we will prove the claim for \( i = k + 1 \). By the inductive assumption, \( \text{dist}(z_k) = d_k = \sum_{i=1}^{k} w(z_{i-1}, z_i) \) at the end of the \( k \)-th round. Right after the edge \((z_k, z_{k+1})\) was relaxed in the \((k+1)\)-th round, it held that \( \text{dist}(z_{k+1}) = \text{dist}(z_{k}) + w(z_k, z_{k+1}) = d_{k} + w(z_k, z_{k+1}) \). In the rest of the algorithm, \( \text{dist}(z_{k+1}) \) could only decrease, indicating that \( d_{k+1} \leq d_k + w(z_k, z_{k+1}) \). On the other hand, as observed earlier, we have \( d_{k+1} \geq d_k + w(z_k, z_{k+1}) \). Therefore, it must hold that \( d_{k+1} = d_k + w(z_k, z_{k+1}) = \sum_{i=1}^{k+1} w(z_{i-1}, z_i) \), and the value of \( \text{dist}(z_{k+1}) \) was \( d_{k+1} \) at the end of the \((k + 1)\)-th round. This completes the proof of the claim.

However, according to the claim, the edge relaxation at Line 9 should not have happened. This gives a contradiction, indicating that our initial assumption (that the lemma is wrong) cannot be true.

The algorithm’s correctness follows from Lemmas 2 and 3.