CSCI3160: Finding a Negative Cycle

Prepared by Yufei Tao

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

$s \leftarrow$ arbitrary vertex in V
$\mathit{dist}(s) \leftarrow 0$ and $\mathit{dist}(v) \leftarrow \infty$ for every vertex $v \in V \setminus \{s\}$
$\mathit{parent}(v) \leftarrow \mathit{nil}$ for all $v \in V$
for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    for each edge $(u, v) \in E$ **do**
        if $\mathit{dist}(v) > \mathit{dist}(u) + w(u, v)$ **then**
            $\mathit{dist}(v) \leftarrow \mathit{dist}(u) + w(u, v)$; $\mathit{parent}(v) \leftarrow u$
for each edge $(u, v) \in E$ **do**
    if $\mathit{dist}(v) > \mathit{dist}(u) + w(u, v)$ **then**
        $\mathit{parent}(v) \leftarrow u$
        /* start tracing back the parent pointers until seeing a vertex twice */
        initialize a vertex sequence S that contains only v
        while $\mathit{parent}(v) \notin S$ **do**
            append $\mathit{parent}(v)$ to S; $v \leftarrow \mathit{parent}(v)$
        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 $\mathit{parent}(u) \neq \mathit{nil}$, then $\mathit{dist}(\mathit{parent}(u)) + w(\mathit{parent}(u), u) <= \mathit{dist}(u)$.*

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

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

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

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

By Lemma 1, we have

$$\begin{aligned} \mathit{dist}(u_3) + w(u_3, u_2) &\leq \mathit{dist}(u_2) \\ \mathit{dist}(u_4) + w(u_4, u_3) &\leq \mathit{dist}(u_3) \\ &\ldots \end{aligned}$$

$$
\begin{aligned}
dist(u_x) + w(u_x, u_{x-1}) &\leq dist(u_{x-1}) \\
dist(u_1) + w(u_1, u_x) &\leq dist(u_1).
\end{aligned}
$$

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

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 π from s to v. Denote by ℓ the number of edges on π; clearly, $\ell \leq |V| - 1$. Denote the vertices on π as z_0, z_1, ..., z_ℓ, where $z_0 = s$ and $z_\ell = v$. Let d_i be the value of $dist(z_i)$ at this moment, for each $i \in [0, \ell]$. Let us make several observations:

- $parent(z_i) = z_{i-1}$ for all $i \in [1, \ell]$, but $parent(z_0) = \text{nil}$.
- The fact $parent(s) = \text{nil}$ implies $d_0 = 0$. To see why, recall that $dist(s)$ is set to 0 at the beginning of the algorithm. Thus, if $d_0 \neq 0$, then $dist(s)$ must have been decreased during the algorithm's execution, in which case $parent(s)$ cannot be nil.
- For each $i \in [1, \ell]$, $d_i \geq d_{i-1} + w(z_{i-1}, z_i)$. At the moment when $dist(z_i)$ was reduced to d_i (which must be due to the relaxation of (z_{i-1}, z_i)), it held that $d_i = dist(z_i) = dist(z_{i-1}) + w(z_{i-1}, z_i)$. The value of $dist(z_{i-1})$ 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 $dist(z_i)$ 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 $dist(z_1) = d_1 = w(s, z_1)$ at the end of the first round. Right after the edge (s, z_1) was relaxed in the first round, it held that $dist(z_1) = w(s, z_1)$. In the rest of the algorithm, $dist(z_1)$ 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 $dist(z_1)$ 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, $dist(z_k) = d_k = \sum_{i=1}^{k} w(z_{i-1}, z_i)$ 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 $dist(z_{k+1}) = dist(z_k) + w(z_k, z_{k+1}) = d_k + w(z_k, z_{k+1})$. In the rest of the algorithm, $dist(z_{k+1})$ 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 $dist(z_{k+1})$ 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.