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

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

Lemma 2. *Suppose that there is a sequence of $x \geq 2$ vertices u_1, u_2, ..., u_x such that $parent(u_i) = u_{i+1}$ for every $i \in [1, x-1]$ and $parent(u_x) = u_1$. 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 $parent(u_1)$, $parent(u_2)$, ..., $parent(u_x)$ was set by an edge relaxation. W.l.o.g., suppose that the edge relaxation for $parent(u_1)$ happened the latest. Consider the moment right before the relaxation. At this moment, we must have

$$dist(u_2) + w(u_2, u_1) < dist(u_1)$$

By Lemma 1, we have

$$\begin{aligned}
dist(u_3) + w(u_3, u_2) &\leq dist(u_2) \\
dist(u_4) + w(u_4, u_3) &\leq dist(u_3) \\
&\;... \\
dist(u_x) + w(u_x, u_{x-1}) &\leq dist(u_{x-1}) \\
dist(u_1) + w(u_1, u_x) &\leq dist(u_1).
\end{aligned}$$

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

Lemma 3. *Consider the moment when the algorithm has come to Line 11. At this moment, if we trace the parent pointers starting from v, we run into 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 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 $i \in [0, \ell]$. As $parent(s) = \text{nil}$, we know $d_0 = 0$.

By induction, we can prove that $dist(z_i)$ was at most d_i after the i-th round of edge relaxation, for each $i \in [0, \ell]$. This implies that the edge relaxation at Line 9 should not have happened. $\square$

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