Detecting and Finding a Negative Cycle

Yufei Tao's Teaching Team

Department of Computer Science and Engineering
Chinese University of Hong Kong

In the lecture, we have shown how to use Bellman-Ford's algorithm to solve the SSSP problem when the input graph $G = (V, E)$ contains no negative cycles. But what if G **does** contain negative cycles? We will see that the algorithm — with minor modifications — can also **detect** the presence of negative cycles. Better still, it can even **find** a negative cycle and the time complexity remains $O(|V||E|)$.

Let us start by reviewing Bellman-Ford's algorithm. The lecture focused on computing shortest-path distances. Here, we will also explain how to construct the shortest path tree.

Edge Relaxation

For every vertex $v \in V$, maintain a value $dist(v)$ equal to the shortest path length from s to v **found so far**.

Relaxing an edge (u, v) means:

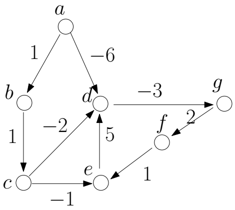
- If $dist(v) \leq dist(u) + w(u, v)$, do nothing;
- Otherwise, reduce $dist(v)$ to $dist(u) + w(u, v)$.

Bellman-Ford's algorithm

1. Set $parent(v) \leftarrow nil$ for all vertices $v \in V$
2. Set $dist(s) \leftarrow 0$, and $dist(v) \leftarrow \infty$ for all other vertices $v \in V$.
3. Repeat the following $|V| - 1$ times
    - Relax all edges (u, v) in E.
      If $dist(v)$ drops after the relaxation, set $parent(v) \leftarrow u$.

Example

Suppose that the source vertex is a.



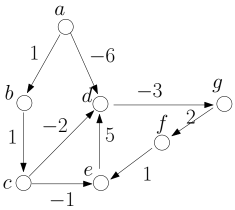
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	∞	nil
c	∞	nil
d	∞	nil
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (f, e):



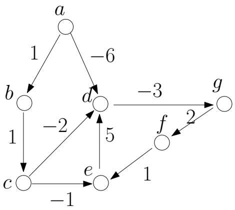
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	∞	nil
c	∞	nil
d	∞	nil
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (d, g):



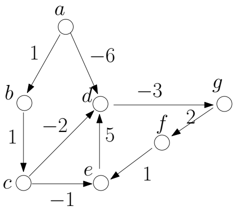
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	∞	nil
c	∞	nil
d	∞	nil
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (a, d):



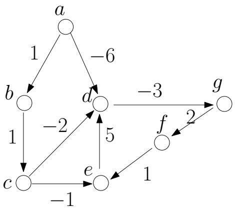
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	∞	nil
c	∞	nil
d	-6	a
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (a, b):



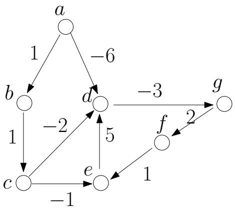
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	∞	nil
d	-6	a
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (b, c):



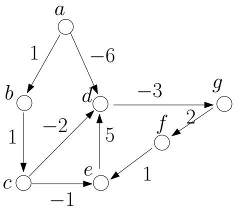
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (c, d):



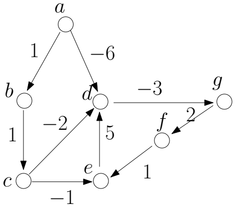
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	∞	nil
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (c, e):



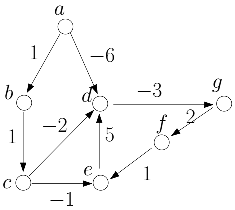
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	1	c
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (g, f):



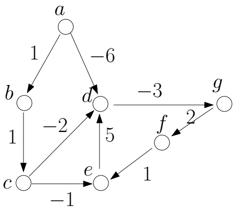
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	1	c
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

First relaxation round

Here is what happens after relaxing (e, d):



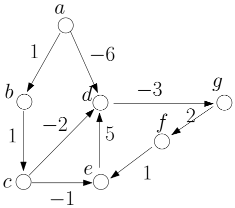
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	1	c
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

Second relaxation round

Here is the table content at the end of this round:



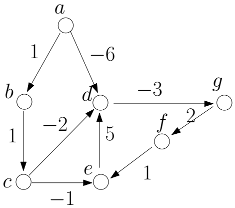
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	1	c
f	-7	g
g	-9	d

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

Third relaxation round

Here is the table content at the end of this round:

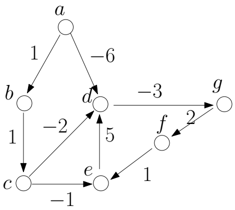


vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	-6	a
e	-6	f
f	-7	g
g	-9	d

Edge relaxation order:
$(f, e), (d, g), (a, d), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

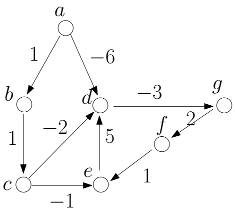
In the same fashion, perform three more relaxation rounds. No more changes to the table:



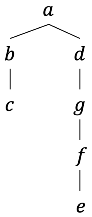
vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	−6	a
e	−6	f
f	−7	g
g	−9	d

Constructing the Shortest Path Tree

For every vertex $v \in V$, if $u = parent(v)$ is not nil, make v a child of u.



vertex v	$parent(v)$
a	nil
b	a
c	b
d	a
e	f
f	g
g	d



Next, we ditch the assumption of no negative cycles. Now, the input graph $G = (V, E)$ **may** or **may not** contain negative cycles. Our new mission is to decide **whether** it does.

We will focus on the situation where G is **strongly connected**, namely, G has only one strongly connected component (in other words, for any $u, v \in V$, G has a path from u to v).

Think: What if G has multiple SCCs?

Detecting Negative Cycles

algorithm negative-cycle-detection

Input: strongly connected $G = (V, E)$ and weight function w

$s \leftarrow$ arbitrary vertex in V

$dist(s) \leftarrow 0$ and $dist(v) \leftarrow \infty$ for every vertex $v \in V \setminus \{s\}$

$parent(v) \leftarrow nil$ for all $v \in V$

for $i \leftarrow 1$ **to** $|V| - 1$ **do**

    for each edge $(u, v) \in E$ **do**

        if $dist(v) > dist(u) + w(u, v)$ **then**

            $dist(v) \leftarrow dist(u) + w(u, v)$; $parent(v) \leftarrow u$

for each edge $(u, v) \in E$ **do**

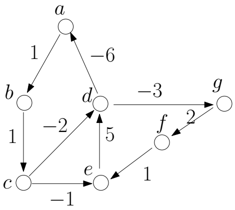
    if $dist(v) > dist(u) + w(u, v)$ **then**

        return "there is a negative cycle"

return "no negative cycles"

Example

Suppose that the source vertex s is set to a.

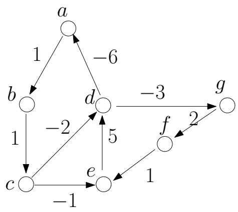


vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	∞	nil
c	∞	nil
d	∞	nil
e	∞	nil
f	∞	nil
g	∞	nil

For illustration purposes, we will relax the edges in this order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

Table content at the end of **first** relaxation round:

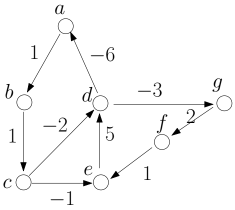


vertex v	$dist(v)$	$parent(v)$
a	0	nil
b	1	a
c	2	b
d	0	c
e	1	c
f	∞	nil
g	∞	nil

Edge relaxation order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

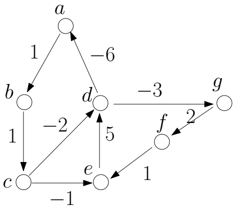
Table content at the end of **second** relaxation round:



vertex v	$dist(v)$	$parent(v)$
a	-6	d
b	-5	a
c	-4	b
d	-6	c
e	-5	c
f	-1	g
g	-3	d

Edge relaxation order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Table content at the end of **third** relaxation round:



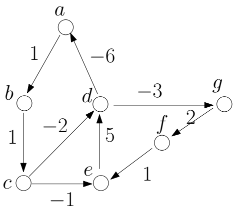
vertex v	$dist(v)$	$parent(v)$
a	-12	d
b	-11	a
c	-10	b
d	-12	c
e	-11	c
f	-7	g
g	-9	d

Edge relaxation order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

Example

In the same fashion, relax all edges for the **fourth time**, **fifth time**, and **sixth time**.

Table content at the end of $|V| - 1$ relaxation round.

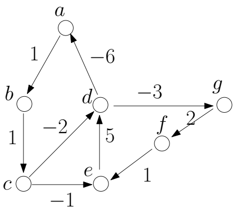


vertex v	$dist(v)$	$parent(v)$
a	-30	d
b	-29	a
c	-28	b
d	-30	c
e	-29	c
f	-25	g
g	-27	d

Example

For negative cycle detection, we perform one more relaxation round. In the absence of negative cycles, no dist values should get improved in this round. Hence, we can safely declare "negative cycles" as soon as **any** dist value is improved.

Edge relaxation order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.

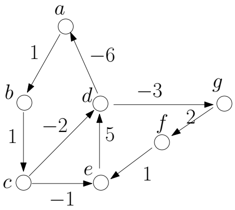


vertex v	$dist(v)$	$parent(v)$
a	-30	d
b	-29	a
c	-28	b
d	-30	c
e	-29	c
f	-25	g
g	-27	d

Relaxing (f, e): $dist(e) < dist(f) + w(f, e)$; no improvement on $dist(e)$.

Example

Edge relaxation order:
$(f, e), (d, g), (d, a), (a, b), (b, c), (c, d), (c, e), (g, f), (e, d)$.



vertex v	$dist(v)$	$parent(v)$
a	-30	d
b	-29	a
c	-28	b
d	-30	c
e	-29	c
f	-25	g
g	$-27 \Rightarrow -33$	d

Relaxing (d, g): $dist(g) > dist(d) + w(d, g)$; improvement on $dist(g)$!

We have now detected a negative cycle.

Correctness

Next, we will prove the correctness of our algorithm. For this purpose, we need establish two directions.

Direction 1 : If the $|V|$-th relaxation round improves the $dist(v)$ of any $v \in V$, there must be a negative cycle.

Direction 2 : If there is a negative cycle, then the $|V|$-th relaxation round must improve the $dist(v)$ of at least one $v \in V$.

Direction 1 : If the $|V|$-th relaxation round improves the $dist(v)$ of any $v \in V$, there must be a negative cycle.

The proof is easy and omitted (see regular exercise solutions).

Direction 2 : If there is a negative cycle, then the $|V|$-th relaxation round must improve the $dist(v)$ of at least one $v \in V$.

Proof: Let the negative cycle be $v_1 \to v_2 \to ... \to v_\ell \to v_1$. Hence:

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

Suppose that the $|V|$-th relaxation round does **not** improve the $dist(v)$ of any vertex $v \in V$. This means:

$$dist(v) \leq dist(u) + w(u, v)$$

holds for every edge in $(u, v) \in E$. Hence:

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

These two bullets lead to:

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

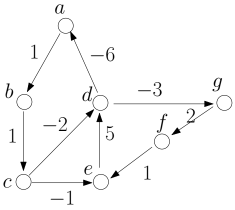
which contradicts (1). $\square$

Finally, we will briefly explain how to **find** a negative cycle in G.

Remark: You will not be tested on the rest of the slides .

Example

Recall: In the 7th relaxation round, we detected a negative cycle in relaxing (d, g) because $dist(g) > dist(d) + w(d, g)$.



vertex v	$dist(v)$	$parent(v)$
a	-30	d
b	-29	a
c	-28	b
d	-30	c
e	-29	c
f	-25	g
g	$-27 \Rightarrow -33$	d

From g, trace the parent pointers until you see a vertex twice.

Tracing back: g, d $(= parent(g))$, c, b, a, d.
So the negative cycle found is : $d \to a \to b \to c \to d$.

The correctness proof is not trivial. See Prof. Tao's proof on the course homepage.