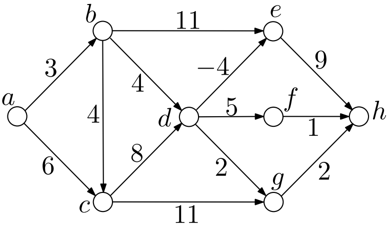
SSSP on DAGs and Negative Cycle Detection

Jianwen Zhao

Department of Computer Science and Engineering
The Chinese University of Hong Kong

SSSP on DAG

Let $G = (V, E)$ be a weighted directed acyclic graph (DAG). Let $w(u, v)$ be the weight of an edge (u, v) in E, which can be positive, 0 or negative.



Given a source vertex $s \in V$, our mission is to design an algorithm which can terminate in $O(|V| + |E|)$ time to find the shortest path distances from s to the vertices in V.

Topological Order

Recall:

- Let $G = (V, E)$ be a DAG, a topological order of G is an ordering of the vertices on V such that, for any edge (u, v), it must hold that u precedes v in the ordering.
- Every DAG has a topological order, and it be obtained by simply running DFS.

An Important Lemma

Recall that the shortest-path distances $spdist(s, v)$ from s to $v \in V$ satisfy:

Lemma.

$$spdist(s, v) = \min_{u \in IN(v)} spdist(s, u) + w(u, v)$$

where $w(u, v)$ denotes the weight of the edge (u, v), and $IN(v)$ is the set of in-neighbors of v.

This lemma implies that SSSP is a dynamic programming problem. What is nontrival is how we should fill in the "matrix". Namely, what is the order of v by which we should compute $spdist(s, v)$?

For a DAG, the answer is "topological order"!

Algorithm

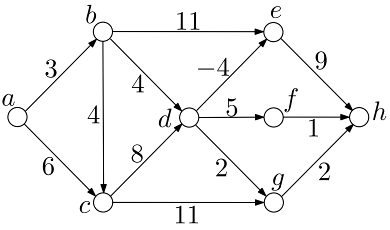
We are now ready to introduce the algorithm. Our algorithm runs as follows.

- Run DFS on G to obtain a topological order of V.
- Initialize $dist(s) = 0$ and $dist(v) = \infty$ for all $v \in V - \{s\}$.
- Process vertices of V according to the topological order. Specifically, when processing a vertex u, relax all the out-going edges (u, v) of u.

After every vertex has been processed, the final $dist(v)$ is the shortest path distance from s to v, for every $v \in V$.

Example

Soppose that the source vertex is a. We first initialize $dist(v)$ for all $v \in V$.

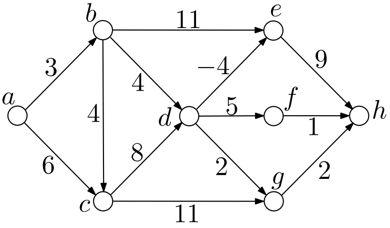


vertex v	$dist(v)$
a	0
b	∞
c	∞
d	∞
g	∞
e	∞
f	∞
h	∞

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Following the topological order, we first process a, namely, relax all the out-going edges of a: $(a, b), (a, c)$.

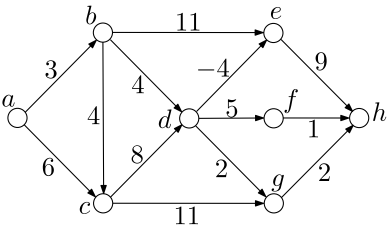


vertex v	$dist(v)$
a	0
b	3
c	6
d	∞
g	∞
e	∞
f	∞
h	∞

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Following the topological order, we then process b, namely, relax all the out-going edges of b: $(b, c), (b, d), (b, e)$.

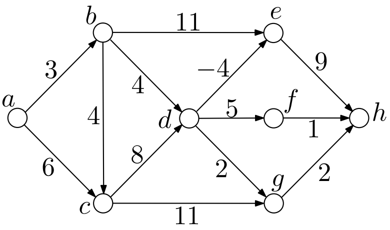


vertex v	$dist(v)$
a	0
b	3
c	6
d	7
g	∞
e	14
f	∞
h	∞

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Following the topological order, we then process c, namely, relax all the out-going edges of c: $(c, d), (c, g)$.

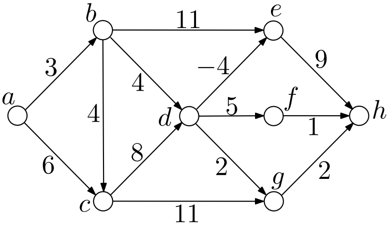


vertex v	$dist(v)$
a	0
b	3
c	6
d	7
g	17
e	14
f	∞
h	∞

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Following the topological order, we then process d, namely, relax all the out-going edges of d: $(d, e), (d, f), (d, g)$.

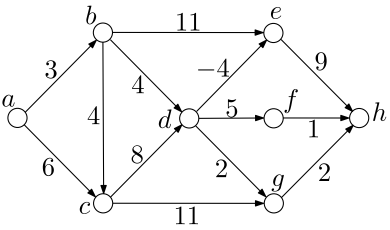


vertex v	$dist(v)$
a	0
b	3
c	6
d	7
g	9
e	3
f	12
h	∞

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Following the topological order, we then process g, namely, relax all the out-going edges of g: (g, h).

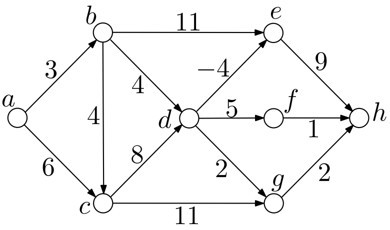


vertex v	$dist(v)$
a	0
b	3
c	6
d	7
g	9
e	3
f	12
h	11

A topological order of V is: a, b, c, d, g, e, f, h.

Example

Similarly, following the topological order, we next process e, f and h and get the final $dist(v)$ for all vertices.



vertex v	$dist(v)$
a	0
b	3
c	6
d	7
g	9
e	3
f	12
h	11

The final $dist(v)$ is the shortest path distance from s (namely, a) to v for every $v \in V$.

Analysis

The correctness of our algorithm follows from the following claim:

Claim. At the moment right before v is processed, $spdist(u)$ has already been computed for every $u \in IN(v)$.

The above claim can be easily established by induction on the number ℓ of edges in a shortest path.

Base case: $\ell = 0$. Trivial.
Inductive case: Suppose that the claim is correct for $\ell < i$. Next we will prove the claim on any vertex v for which there is a shortest path π with length $\ell = i$. Let u be the predecessor of v on π. By the topological order, when v is about to be processed, u must have already been processed. Think about what happens when we relax the edge (u, v).

Negative Cycle Detection

Let $G = (V, E)$ be a weighted directed graph where the weight of an edge (u, v) is $w(u, v)$. Define $\lambda = \infty$.

Consider the following algorithm:

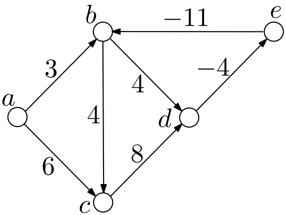
- Pick an arbitrary $s \in V$, initialize $dist(s) = 0$ and $dist(v) = \lambda$ for all $v \in V - \{s\}$.
- Repeat the following $|V| - 1$ times:
  - relax all the edges in E.

Prove: when the algorihtm terminates, G contains a negative cycle **if and only if** there is an edge $(u, v) \in E$ such that

$$dist(v) > dist(u) + w(u, v)$$

Example

Consider the following graph which contains a negative cycle $b \to d \to e$. Note that $\lambda = 3 + 6 + 4 + 8 + 4 = 25$. Let the source vertex be a, we initialize $dist(a) = 0$ and $dist(v) = \lambda$ for every other vertex v.

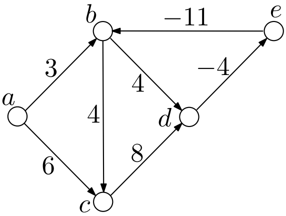


vertex v	$dist(v)$
a	0
b	25
c	25
d	25
e	25

There is no ∞ in actual programming. Here it suffices to use $\lambda = 25$ as the "infinity". **Think:** why?

Example

After relax all the edges $|V| - 1 = 4$ times by following the alphabetical order of the edges, we have:



vertex v	$dist(v)$
a	0
b	-41
c	-26
d	-26
e	-30

Since the edge (b, d) satisfies $dist(d) > dist(b) + w(b, d)$, we know that there must be a negative cycle!

Negative Cycle Detection

Prove: when the algorihtm terminates, G contains a negative cycle **if and only if** there is an edge $(u, v) \in E$ such that

$$dist(v) > dist(u) + w(u, v) \qquad (1)$$

Proof. 1. The if direction.

We already know that Bellman-Fords algorithm correctly finds all shortest path distances (from s) after $|V| - 1$ rounds of edge relaxations if there are no negative cycles in G.

Hence, if inequality (1) still holds after $|V| - 1$ rounds, it means that an even shorter path from s to v has just been discovered. Therefore, in such a case, G must contain a negative cycle.

Negative Cycle Detection

Proof (cont.).

2. The only if direction.

Suppose that there is a negative cycle $v_1 \to v_2 \to \cdots \to v_\ell \to v_1$. Then we have

$$w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) < 0 \tag{2}$$

Assume on the contrary that (1) does not hold on any edge in E, then we have

$$dist(v_{i+1}) \leq dist(v_i) + w(v_i, v_{i+1}) \quad \text{for all} \quad i \in [1, \ell - 1] \tag{3}$$

$$dist(v_1) \leq dist(v_\ell) + w(v_\ell, v_1) \tag{4}$$

Negative Cycle Detection

Proof (cont.).

The inequalities (3) and (4) together imply:

$$\sum_{i=1}^{\ell} dist(v_i) \leq \sum_{i=1}^{\ell} dist(v_i) + w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \tag{5}$$

$\Rightarrow$

$$w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \geq 0 \tag{6}$$

which contradicts (2). $\square$