Introduction to Algorithms

Meng-Tsung Tsai

11/28/2018

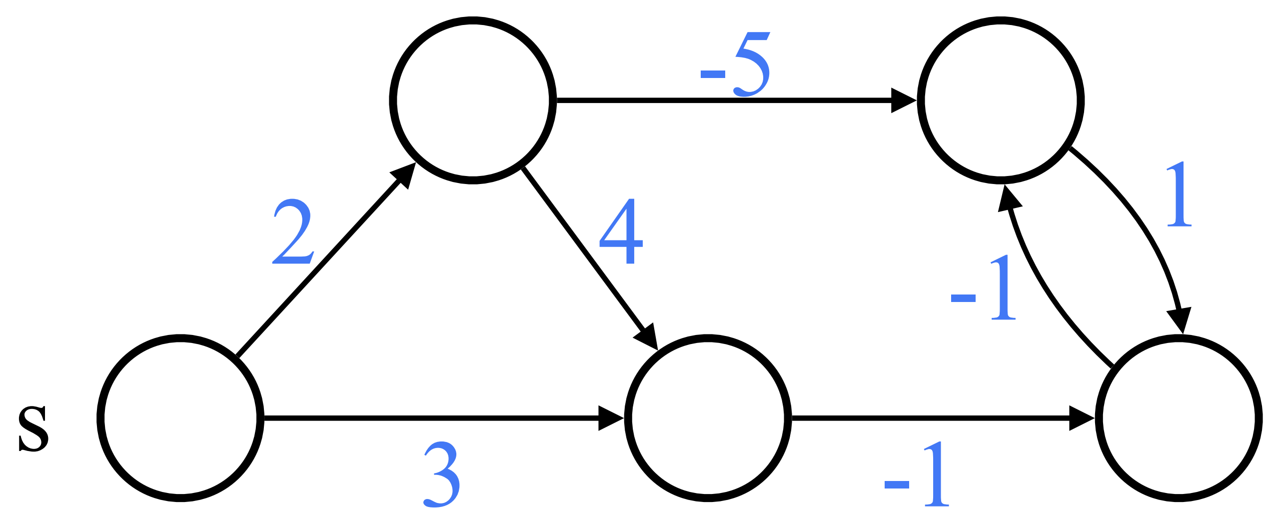
Single-Source Shortest Paths

Problem

Input: a directed graph G, a weight function $\omega : E \rightarrow \mathbf{R}$, and a (source) node s in G.

Output: for each node v in G, output the shortest (may be not simple) path P from s to v ($\omega(P) = \sum_{e \in P} \omega(e)$ is minimized).

<u>Example</u>.



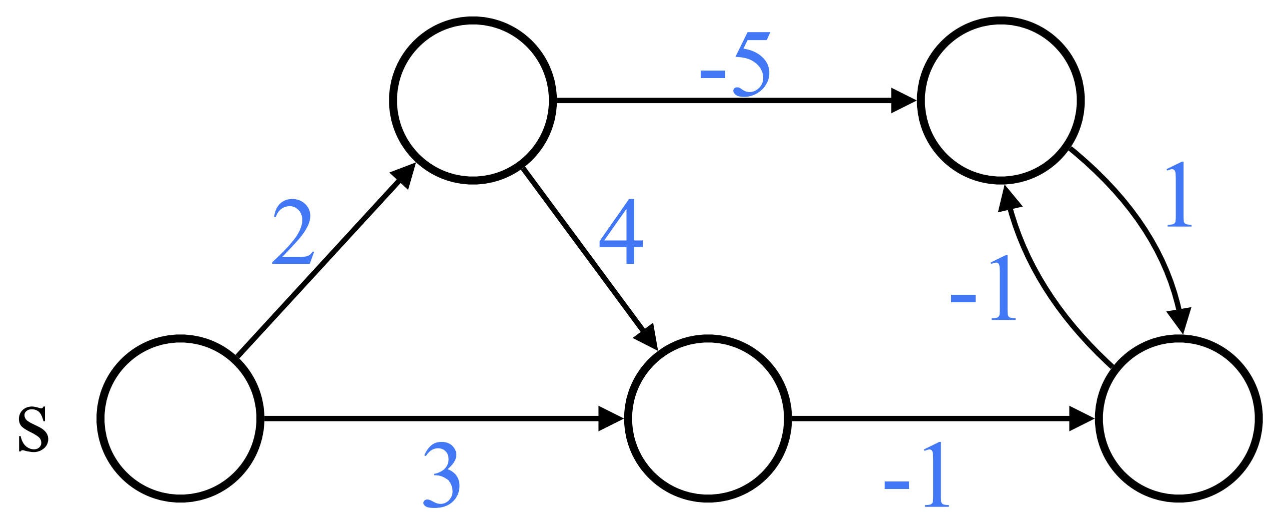
Problem

Input: a directed graph G, a weight function $\omega : E \rightarrow \mathbf{R}$, and a (source) node s in G.

Output: for each node v in G, output the shortest (may be not simple) path P from s to v ($\omega(P) = \sum_{e \in P} \omega(e)$ is minimized).

Example.

A path is simple if no node on it repeats.

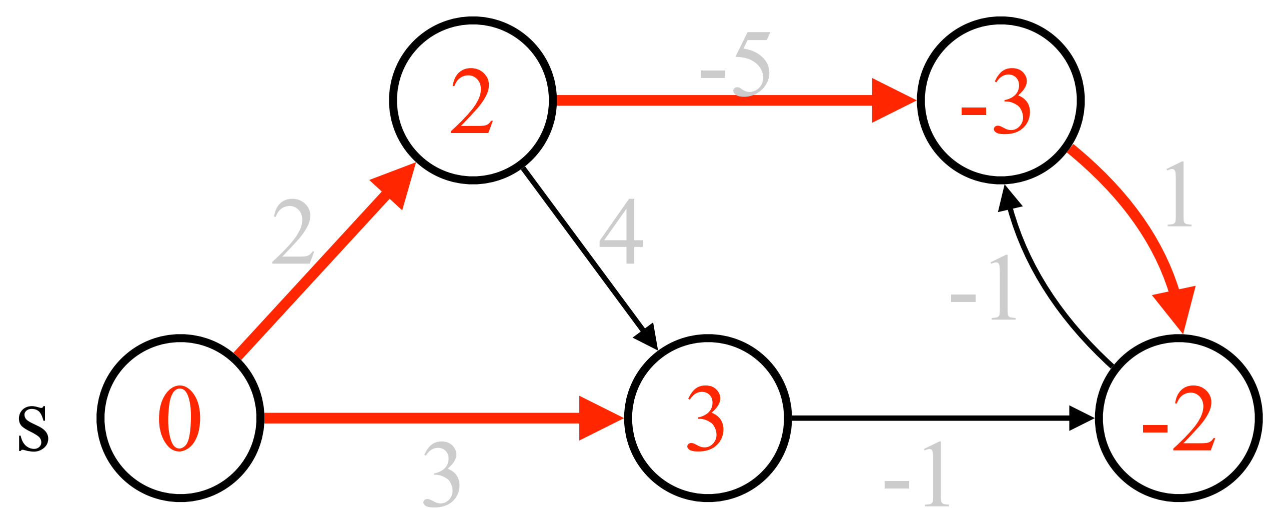


Problem

Input: a directed graph G, a weight function $\omega : E \to \mathbf{R}$, and a (source) node s in G.

Output: for each node v in G, output the shortest (may be not simple) path P from s to v ($\omega(P) = \sum_{e \in P} \omega(e)$ is minimized).

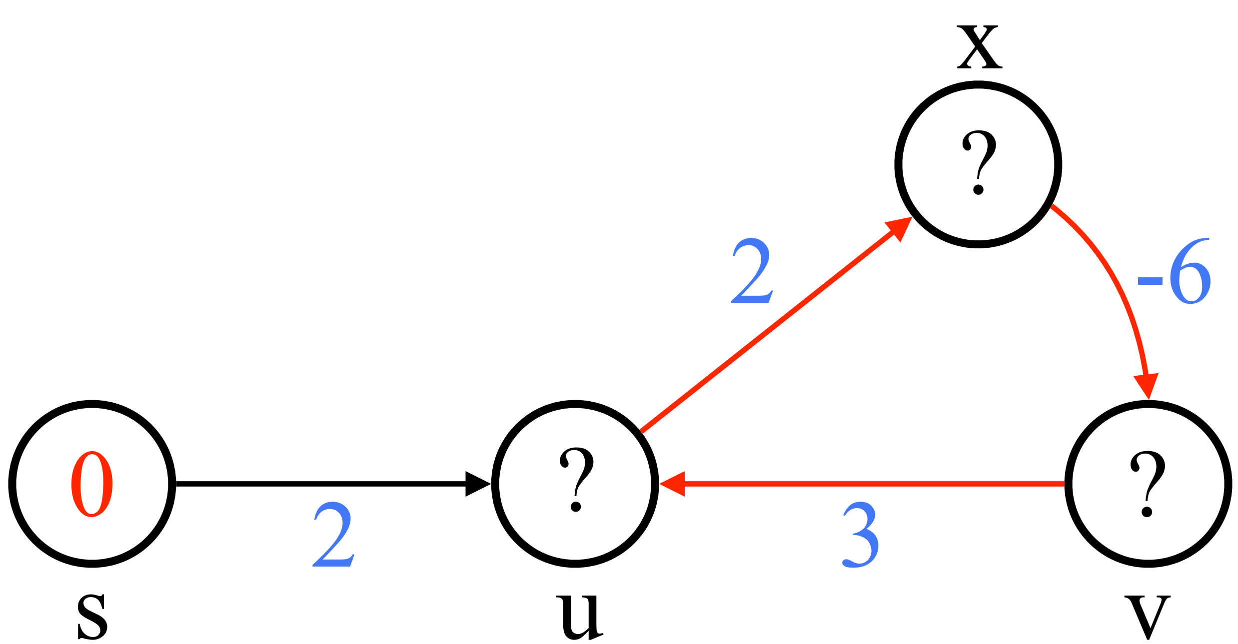
Example.



Negative Cycles

We say a cycle C **negative** if $\omega(C) = \sum_{e \in C} \omega(e) < 0$.

Example.

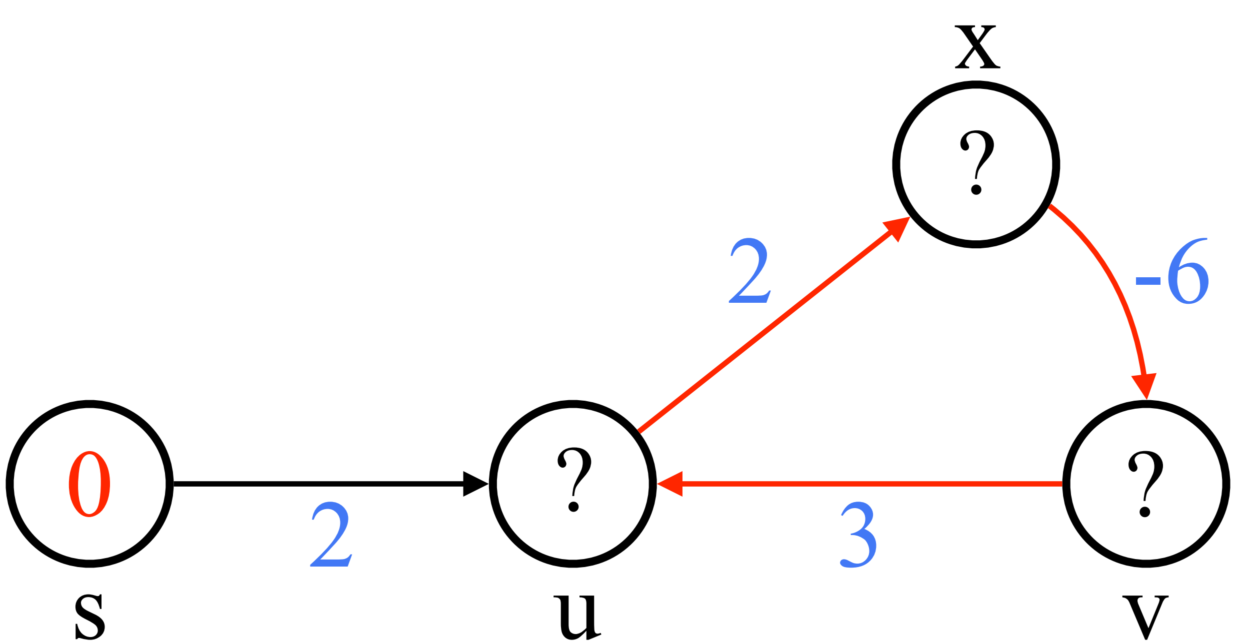


Let dis(v) be the shortest distance between s and v. That is, there exists a path P from s to v whose $\omega(P) = \mathrm{dis}(v)$ and there exists no P' from s to v whose $\omega(P') < \mathrm{dis}(v)$.

Negative Cycles

We say a cycle C **negative** if $\omega(C) = \sum_{e \in C} \omega(e) < 0$.

<u>Example</u>.



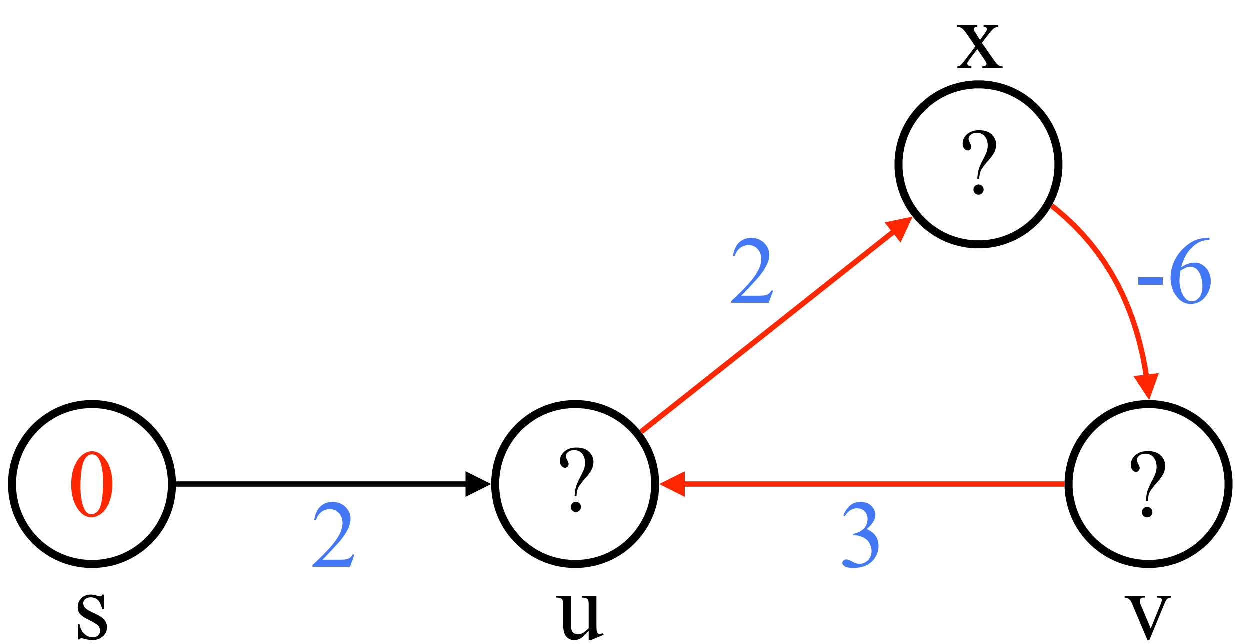
Let dis(v) be the shortest distance between s and v. That is, there exists a path P from s to v whose $\omega(P) = \text{dis}(v)$ and there exists no P' from s to v whose $\omega(P') < \text{dis}(v)$.

Clearly, dis(s) = 0. What is dis(u)?

Negative Cycles

We say a cycle C **negative** if $\omega(C) = \sum_{e \in C} \omega(e) < 0$.

Example.



Let dis(v) be the shortest distance between s and v. That is, there exists a path P from s to v whose $\omega(P)$ = dis(v) and there exists no P' from s to v whose $\omega(P')$ < dis(v).

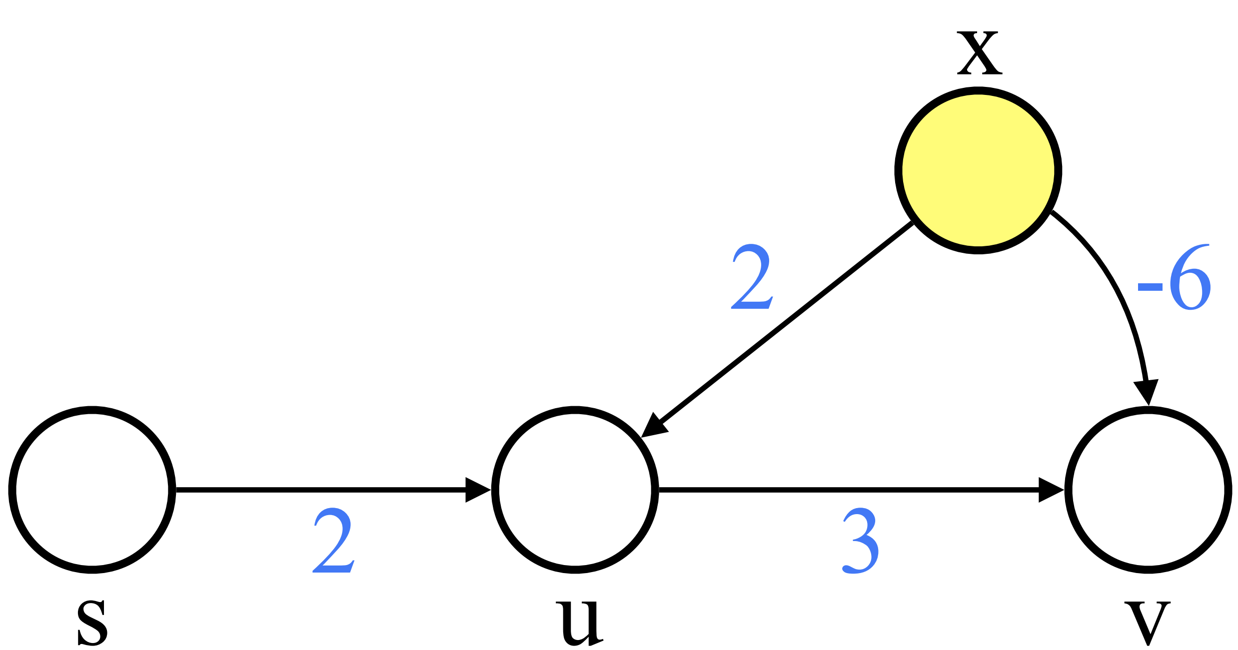
Clearly, dis(s) = 0. What is dis(u)?

dis(u)=dis(x)=dis(v)=$-\infty$.

Reachability

There may be some nodes in G not reachable from s.

<u>Example</u>.



Let dis(v) = ∞ for any node v that is not reachable from s.

Preliminaries

Building blocks (1/3)

```
Initialization(G, s){
    foreach node v in G {
        est(v) ← ∞; // an upper bound of dis(v)
        pred(v) ← NIL;
    }
    est(s) ← 0;
}
```

The estimated distance est(v) is initialized as ∞, and it will be modified by some shortest-path algorithm A. When A terminates, est(v) = dis(v).

Building blocks (1/3)

```
Initialization(G, s){
    foreach node v in G {
        est(v) ← ∞; // an upper bound of dis(v)
        pred(v) ← NIL;
    }
    est(s) ← 0;
}
```

pred(v) is initialized as NIL, and it will be modified by some shortest-path algorithm A. When A terminates, prev(v) will be the predecessor of v in a shortest path P from s to v.

Building blocks (2/3)

```
Print-Path(G, s, v){
    if(s == v){
        output s; return;
    }
    Print-Path(G, s, pred(v));
    output v; return;
}
```

The shortest path $s \rightsquigarrow v$ =
the shortest path $s \rightsquigarrow pred(v)$ + the edge $(pred(v), v)$.

Building blocks (3/3)

```
Relax(u, v, ω){
    if(est(v) > est(u) + ω(u, v)){ // the inequality does not hold
if est(u) = est(v) = ∞ because ∞ and ∞+C are incomparable
        est(v) ← est(u) + ω(u, v);
        pred(v) ← u;
    }
}
```

If $est(v) \neq \infty$, then it implies that we found a path P ($s \rightsquigarrow v$) whose $\omega(P) = est(v)$.

Building blocks (3/3)

```
Relax(u, v, ω){
    if(est(v) > est(u) + ω(u, v)){ // the inequality does not hold
if est(u) = est(v) = ∞ because ∞ and ∞+C are incomparable
        est(v) ← est(u) + ω(u, v);
        pred(v) ← u;
    }
}
```

Let P_u be the shortest path $s \rightsquigarrow u$ found so far.

Let P_v be the shortest path $s \rightsquigarrow v$ found so far.

If $\omega(P_u) + \omega(u, v) < \omega(P_v)$, we replace P_v with $P_u + (u, v)$.

The Bellman-Ford Algorithm

Restriction

We assume that the input graph G contains no negative cycle reachable from s for the Bellman-Ford algorithm. Note that G may have negative cycles not reachable from s.

<u>Claim</u>. No negative cycle reachable from s $\Rightarrow$ for any node v in G reachable from s, there exists a shortest path $s \rightsquigarrow v$ of length $< n$. (length: # of edges)

Proof. If $s \rightsquigarrow v$ is a path P of length $\geq n$, then $s \rightsquigarrow v$ contains a cycle C. Since $\omega(C) \geq 0$, we can obtain P' by removing C from P so that $\omega(P') \leq \omega(P)$. We repeatedly remove cycles from P until the resulting P' has length $< n$.

Pseudocode

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
    foreach edge (u, v) in G
        if (est(v) > est(u) + ω(u, v)) // can be further relaxed
            return There-Exists-a-Negative-Cycle-Reachable-from-s;
    return No-Negative-Cycle-Reachable-from-s;
}
```

Pseudocode

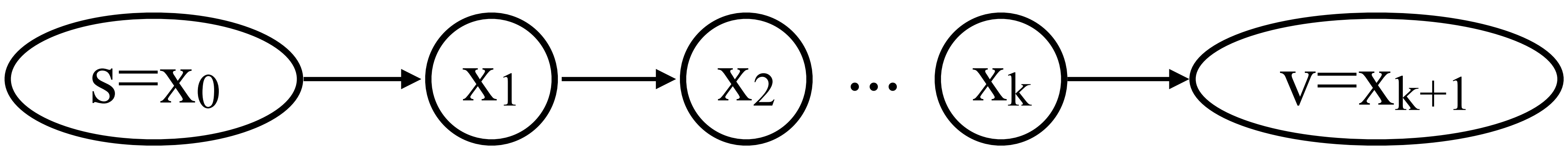
```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
    foreach edge (u, v) in G
        if (est(v) > est(u) + ω(u, v)) // can be further relaxed
            return There-Exists-a-Negative-Cycle-Reachable-from-s;
    return No-Negative-Cycle-Reachable-from-s;
}
```

When Bellman-Ford terminates, est(v)=dis(v) for all v's. In addition, est(v)=est(pred(v))+ω(pred(v), v) for all v's.

Correctness

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
```

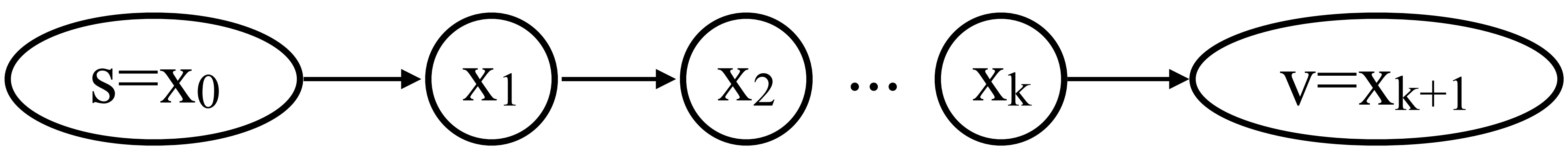
Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) … (x_k, v).



Correctness

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
```

Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) ... (x_k, v).

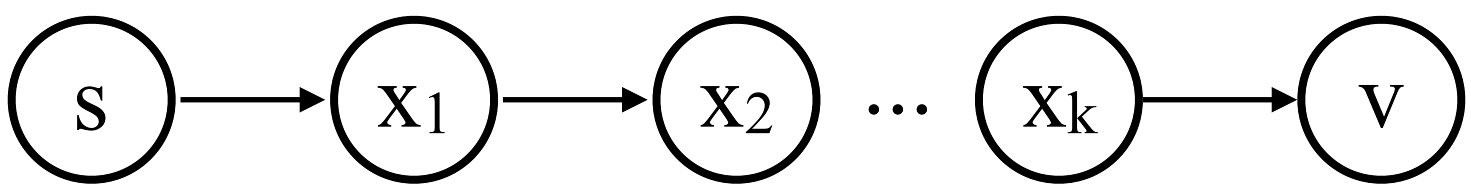


Any subpath between x_i and x_j for $0 \leq i < j \leq k+1$ in the shortest path $s \rightarrow v$ is a shortest path $x_i \rightarrow x_j$. Why?

Correctness

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
```

Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) … (x_k, v).

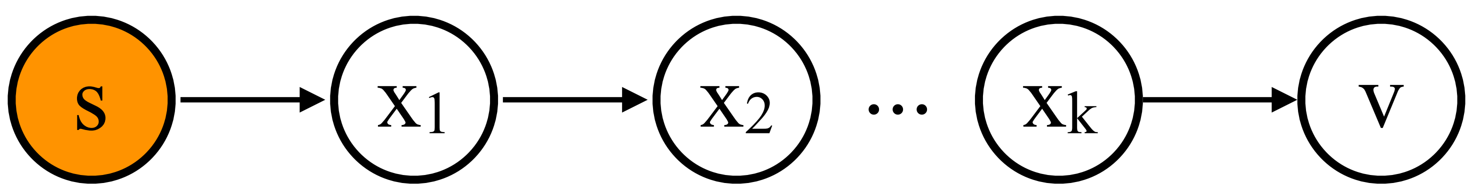


est(s) = 0 before the loop. Since G contains no cycle C from s to s whose $\omega(C) < 0$, we have est(s) = dis(s) = 0 as an invariant in the loop. We color any node v orange when est(v) becomes an invariant in the loop.

Correctness

Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);

Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) ... (x_k, v).

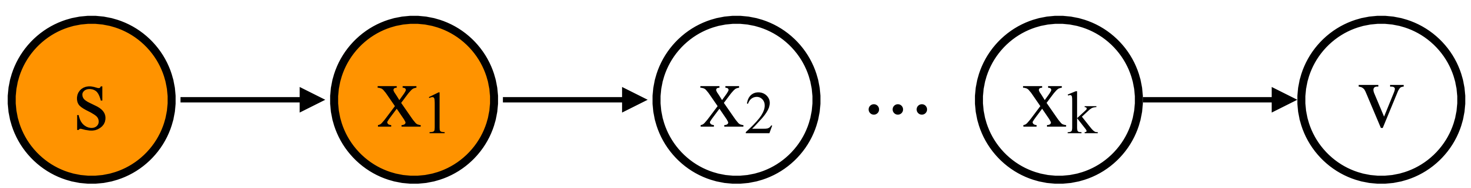


$est(x_1) = \infty$ before the loop. After seeing the edge (s, x_1) in the 1st iteration of the loop, $est(x_1) \leq est(s) + \omega(s, x_1) = dis(x_1)$. The last equality holds because edge (s, x_1) is a shortest path $s \rightarrow x_1$. (Why?) Since we can't have $est(v) < dis(v)$ for any v, $est(x_1) = dis(x_1)$ after the 1st iteration of the loop. We color x_1 orange since then.

Correctness

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
```

Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) ... (x_k, v).

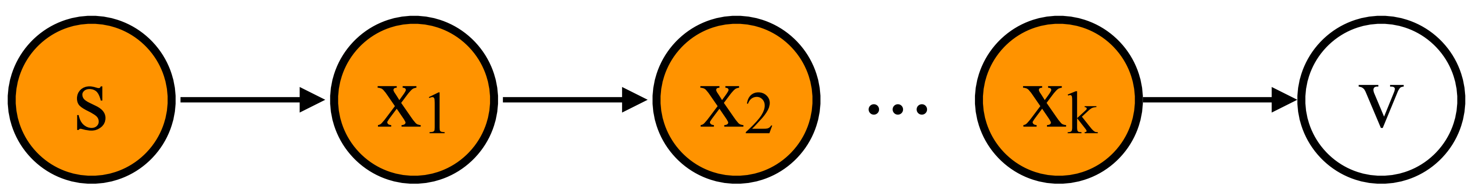


$est(x_2) = \infty$ before the loop. After seeing the edge (x_1, x_2) in the 2nd iteration of the loop, $est(x_2) \leq est(x_1) + \omega(x_1, x_2) = dis(x_2)$. The last equality holds because a shortest path $s \rightarrow x_1$ + edge (s, x_1) is a shortest path $s \rightarrow x_2$. We color x_2 orange since then.

Correctness

Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);

Let the shortest path $s \rightarrow v$ be edges (s, x_1) (x_1, x_2) ... (x_k, v).



By induction, we have est(v) = dis(v) after the (k+1)-th iteration of the loop. Because $k+1 \leq n-1$ (why?), after the loop is finished we have est(v) = dis(v) and est(pred(v)) = dis(pred(v)). This holds for any v and the predecessor of any v.

Correctness

```
    foreach edge (u, v) in G
        if (est(v) > est(u) + ω(u, v)) // can be further relaxed
            return There-Exists-a-Negative-Cycle-Reachable-from-s;
    return No-Negative-Cycle-Reachable-from-s;
}
```

<u>If G has no negative cycle reachable from s</u>, we can't Relax an edge in the loop because est(v) becomes an invariant for all v's.

If G has some negative cycle reachable from s, let the cycle C be $(x_1, x_2, x_3, ..., x_k, x_1)$ and $\omega(C) < 0$. If G passes the test in the loop, then $est(x_2) \leq est(x_1) + \omega(x_1, x_2)$, $est(x_3) \leq est(x_2) + \omega(x_2, x_3)$, ..., and $est(x_1) \leq est(x_k) + \omega(x_k, x_1)$. Summing all inequalities, we get $0 \leq \omega(C)$. $\rightarrow\leftarrow$ <u>This case</u> can't pass the test.

Running time

```
Bellman-Ford(G, ω, s){
    Initialization(G, s);
    for i = 1 to n-1
        foreach edge (u, v) in G
            Relax(u, v, ω);
    foreach edge (u, v) in G
        if (est(v) > est(u) + ω(u, v)) // can be further relaxed
            return There-Exists-a-Negative-Cycle-Reachable-from-s;
    return No-Negative-Cycle-Reachable-from-s;
}
```

The nested loops have O(nm) iterations, and Relax() needs O(1) time. Thus, the Bellman-Ford algorithm runs in O(nm) time.

Exercise

Input: a directed graph G and a weight function $\omega : E \to \mathbf{R}$.

Output: If G has a negative cycle, output "Yes." Otherwise, output "No."

Is this problem solvable in $O(nm)$ time? Why or why not?