CMPSC 465: LECTURE XVII

Shortest Path with Negative Weights

Ke Chen

October 08, 2025

Negative edge weights

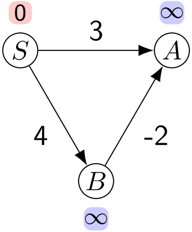
- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.

Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?

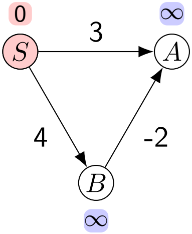
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



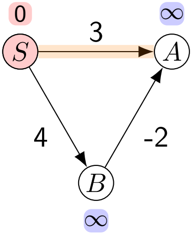
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



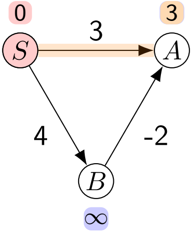
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



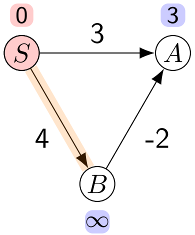
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



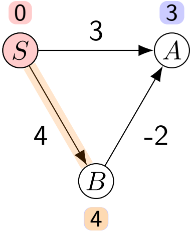
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



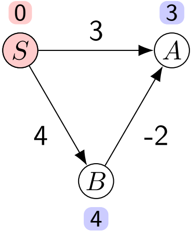
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



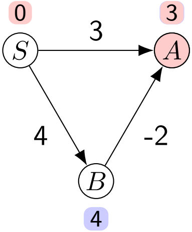
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



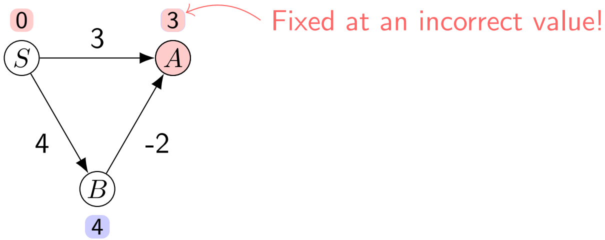
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



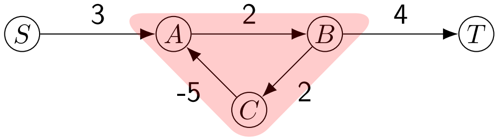
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?



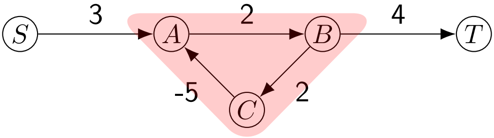
Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?
- Even worse, if there are negative cycles, the "shortest distance" is not well-defined.



Negative edge weights

- Recall that the correctness of Dijkstra's algorithm relies on all edge weights being nonnegative.
- What happens if there are negative edges?
- Even worse, if there are negative cycles, the "shortest distance" is not well-defined.



- Note that in directed graphs, negative edges $\neq$ negative cycles; however, in undirected graphs, a negative edge $=$ a negative cycle. (Why?)

Update is fine with negative edges

Recall the Update operation:

Update($(v, w) \in E$)

if $dist[w] > dist[v] + \ell(v, w)$ **then**

$\quad dist[w] = dist[v] + \ell(v, w)$

Update is fine with negative edges

Recall the Update operation:

Update($(v, w) \in E$)

- **if** $dist[w] > dist[v] + \ell(v, w)$ **then**
  - $dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path.

Update is fine with negative edges

Recall the Update operation:

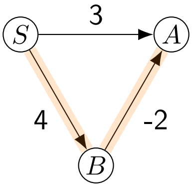
Update($(v, w) \in E$)

if $dist[w] > dist[v] + \ell(v, w)$ **then**

$\quad dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path. Remains valid with negative edges.

Update is fine with negative edges



Recall the Update operation:

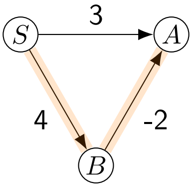
Update($(v, w) \in E$)

if $dist[w] > dist[v] + \ell(v, w)$ **then**

$\quad dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path. Remains valid with negative edges.

Update is fine with negative edges



Recall the Update operation:

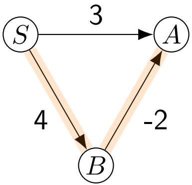
Update($(v, w) \in E$)

if $dist[w] > dist[v] + \ell(v, w)$ **then**

$\quad dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path. Remains valid with negative edges.
- Having additional Update calls doesn't hurt.

Update is fine with negative edges



Recall the Update operation:

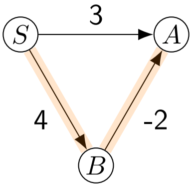
Update($(v, w) \in E$)

if $dist[w] > dist[v] + \ell(v, w)$ **then**

$\quad dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path. Remains valid with negative edges.
- Having additional Update calls doesn't hurt.
- Dijkstra applies a smart sequence of only $O(|E|)$ Update calls that's guaranteed to include the required sequence for each node if no negative weights.

Update is fine with negative edges



Recall the Update operation:

Update($(v, w) \in E$)

- **if** $dist[w] > dist[v] + \ell(v, w)$ **then**
  - $dist[w] = dist[v] + \ell(v, w)$

- The shortest distance from S to any node can be correctly computed by a sequence of Update calls along a shortest path. Remains valid with negative edges.
- Having additional Update calls doesn't hurt.
- Dijkstra applies a smart sequence of only $O(|E|)$ Update calls that's guaranteed to include the required sequence for each node if no negative weights. It may fail with negative edges.

One sequence to rule them all

Can we find a sequence of Update calls that always work?

One sequence to rule them all

Can we find a sequence of Update calls that always work?

Yes! The following sequence contains ALL possible sequences of Update calls of length at most $|V| - 1$:

$$\left.\begin{array}{l}
\mathrm{edge}_1 \longrightarrow \mathrm{edge}_2 \longrightarrow \mathrm{edge}_3 \longrightarrow \cdots \longrightarrow \mathrm{edge}_{|E|} \\
\longrightarrow \mathrm{edge}_1 \longrightarrow \mathrm{edge}_2 \longrightarrow \mathrm{edge}_3 \longrightarrow \cdots \longrightarrow \mathrm{edge}_{|E|} \\
\longrightarrow \mathrm{edge}_1 \longrightarrow \mathrm{edge}_2 \longrightarrow \mathrm{edge}_3 \longrightarrow \cdots \longrightarrow \mathrm{edge}_{|E|} \\
\longrightarrow \\
\qquad\qquad \cdots\cdots \\
\longrightarrow \mathrm{edge}_1 \longrightarrow \mathrm{edge}_2 \longrightarrow \mathrm{edge}_3 \longrightarrow \cdots \longrightarrow \mathrm{edge}_{|E|}
\end{array}\right\} |V| - 1 \text{ times}$$

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V| - 1$ times.

- Why $|V| - 1$?

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V| - 1$ times.

- Why $|V| - 1$? A shortest path contains at most $|V| - 1$ edges.

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V| - 1$ times.

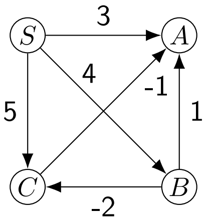
- Why $|V| - 1$? A shortest path contains at most $|V| - 1$ edges.
- The order of edges doesn't matter.

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V| - 1$ times.

- Why $|V| - 1$? A shortest path contains at most $|V| - 1$ edges.
- The order of edges doesn't matter.

Example



$(S, A) \twoheadrightarrow (B, A) \twoheadrightarrow (C, A) \twoheadrightarrow (S, C) \twoheadrightarrow (B, C) \twoheadrightarrow (S, B)$

$\twoheadrightarrow (S, A) \twoheadrightarrow (B, A) \twoheadrightarrow (C, A) \twoheadrightarrow (S, C) \twoheadrightarrow (B, C) \twoheadrightarrow (S, B)$

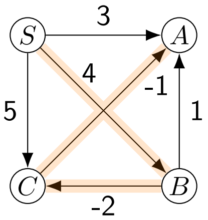
$\twoheadrightarrow (S, A) \twoheadrightarrow (B, A) \twoheadrightarrow (C, A) \twoheadrightarrow (S, C) \twoheadrightarrow (B, C) \twoheadrightarrow (S, B)$

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V|-1$ times.

- Why $|V|-1$? A shortest path contains at most $|V|-1$ edges.
- The order of edges doesn't matter.

Example



$$(S,A) \twoheadrightarrow (B,A) \twoheadrightarrow (C,A) \twoheadrightarrow (S,C) \twoheadrightarrow (B,C) \twoheadrightarrow (S,B)$$

$$\twoheadrightarrow (S,A) \twoheadrightarrow (B,A) \twoheadrightarrow (C,A) \twoheadrightarrow (S,C) \twoheadrightarrow (B,C) \twoheadrightarrow (S,B)$$

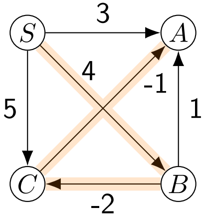
$$\twoheadrightarrow (S,A) \twoheadrightarrow (B,A) \twoheadrightarrow (C,A) \twoheadrightarrow (S,C) \twoheadrightarrow (B,C) \twoheadrightarrow (S,B)$$

One sequence to rule them all

Idea Call Update on each edge, and repeat $|V| - 1$ times.

- Why $|V| - 1$? A shortest path contains at most $|V| - 1$ edges.
- The order of edges doesn't matter.

Example



$(S, A) \to (B, A) \to (C, A) \to (S, C) \to (B, C) \to (S, B)$

$\to (S, A) \to (B, A) \to (C, A) \to (S, C) \to (B, C) \to (S, B)$

$\to (S, A) \to (B, A) \to (C, A) \to (S, C) \to (B, C) \to (S, B)$

Bellman-Ford algorithm

Input: Graph $G = (V, E, \ell)$, starting vertex s
Output: Shortest path from s to any other vertex

```
Bellman-Ford(G, s)
    // dist stores distances from s
    foreach v ∈ V do
        dist[v] = ∞
    dist[s] = 0

    repeat |V| − 1 times do
        foreach e ∈ E do
            Update(e)
```

Bellman-Ford algorithm

Input: Graph $G = (V, E, \ell)$, starting vertex s
Output: Shortest path from s to any other vertex

Bellman-Ford(G, s)

```
// dist stores distances from s
foreach v ∈ V do
    dist[v] = ∞
dist[s] = 0

repeat |V| − 1 times do
    foreach e ∈ E do
        Update(e)
```

Time complexity?

Bellman-Ford algorithm

Input: Graph $G = (V, E, \ell)$, starting vertex s
Output: Shortest path from s to any other vertex

Bellman-Ford(G, s)

```
// dist stores distances from s
foreach v ∈ V do
    dist[v] = ∞
dist[s] = 0

repeat |V| − 1 times do
    foreach e ∈ E do
        Update(e)
```

Time complexity? $O(|V| \cdot |E|)$

Bellman-Ford algorithm

Input: Graph $G = (V, E, \ell)$, starting vertex s
Output: Shortest path from s to any other vertex

Bellman-Ford(G, s)

```
// dist stores distances from s
foreach v ∈ V do
    dist[v] = ∞
dist[s] = 0

repeat |V| − 1 times do
    foreach e ∈ E do
        Update(e)
```

Time complexity? $O(|V| \cdot |E|)$

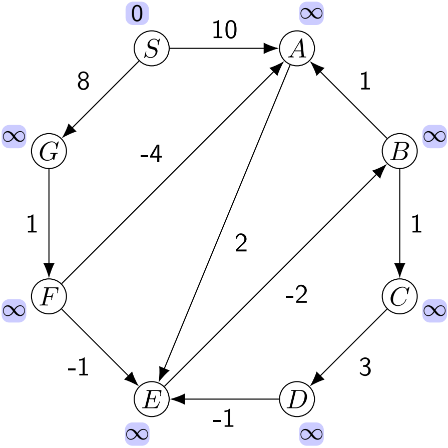
Can we do better?

Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round:

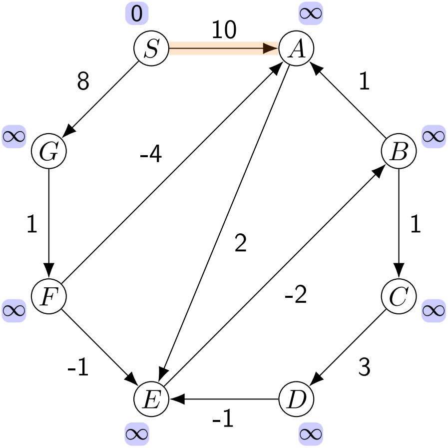


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

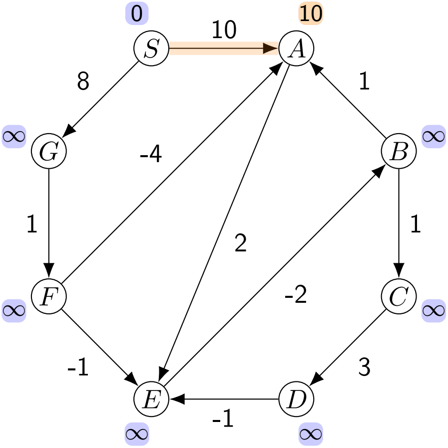


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

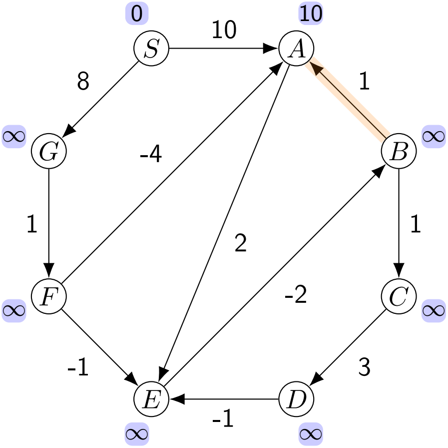


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

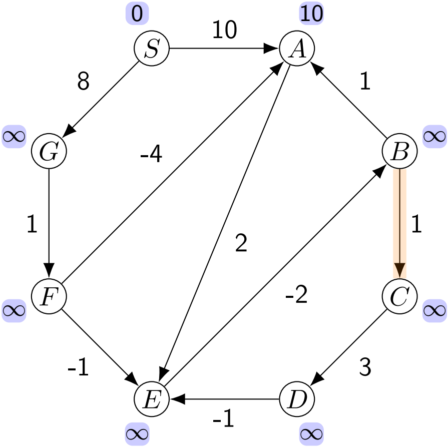


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

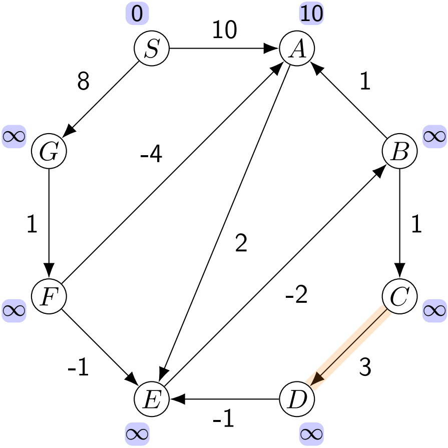


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

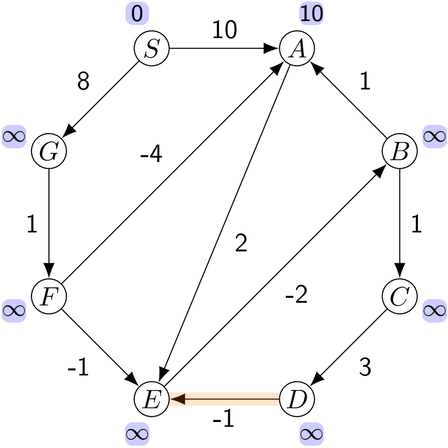


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

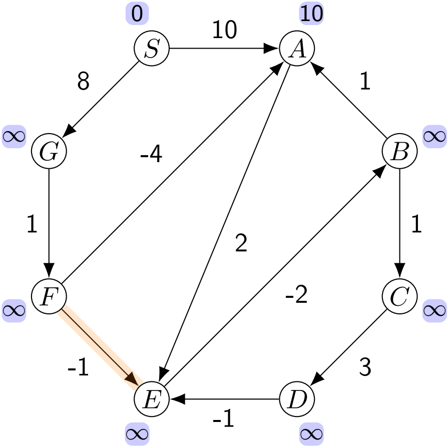


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

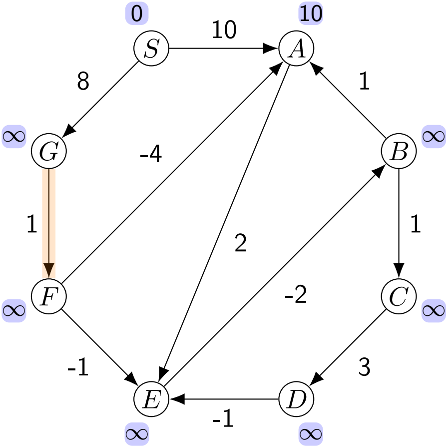


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

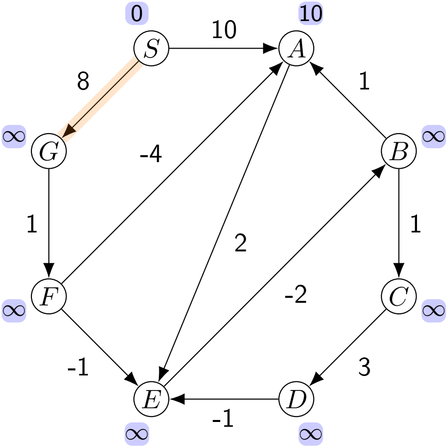


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

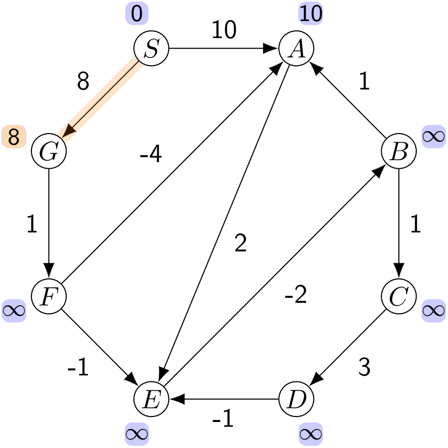


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

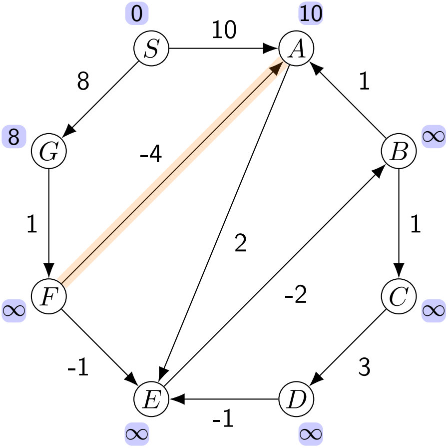


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

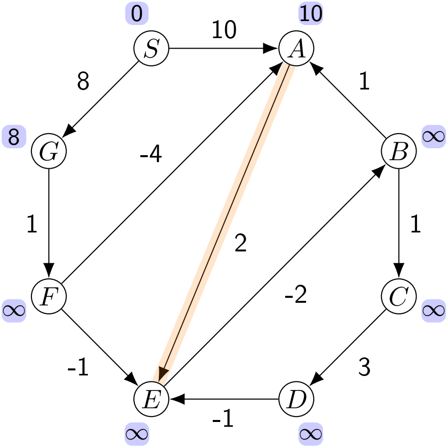


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

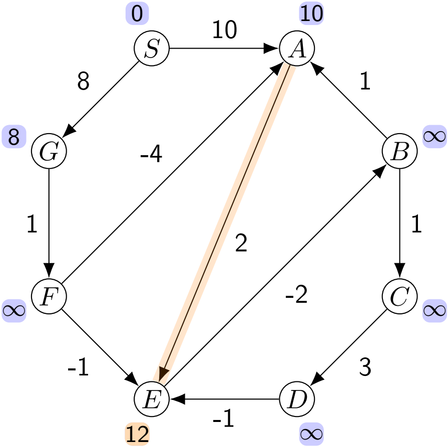


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

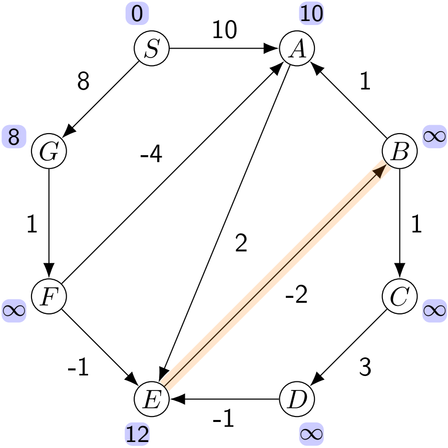


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

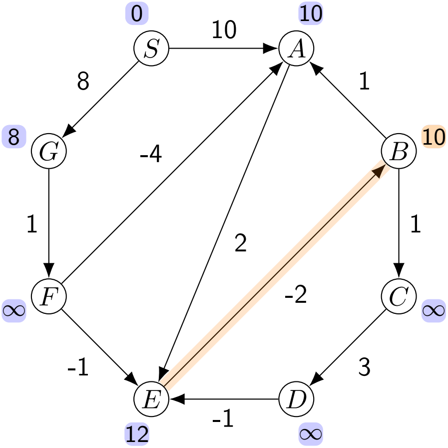


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 1

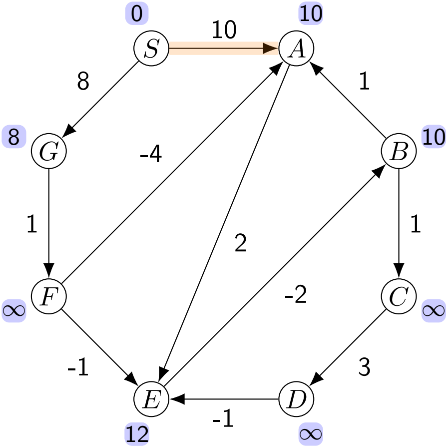


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

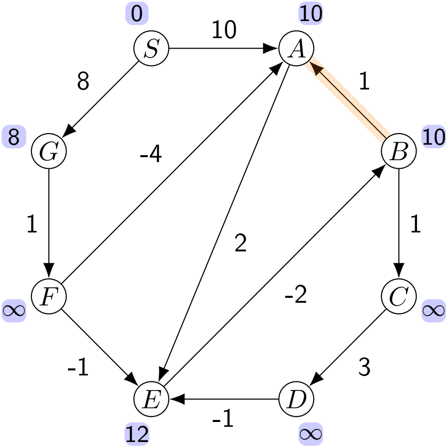


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

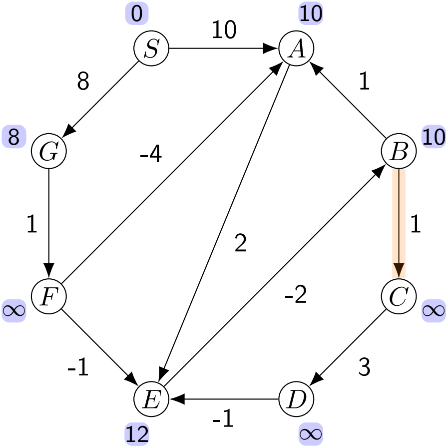


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

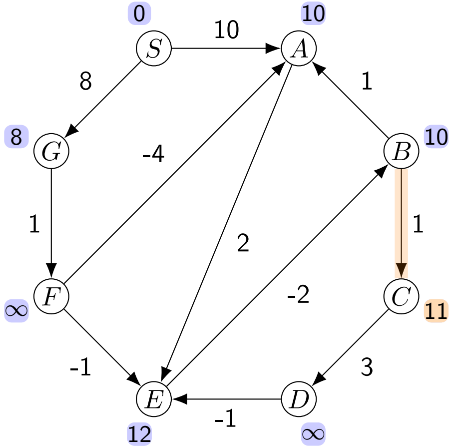


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

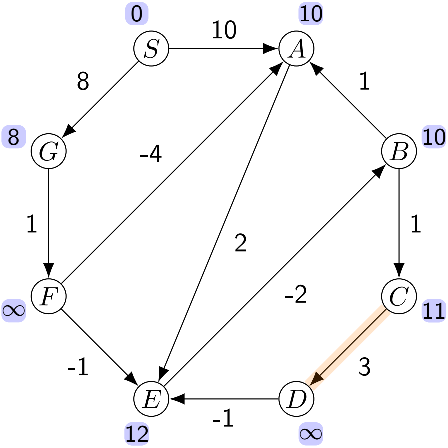


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

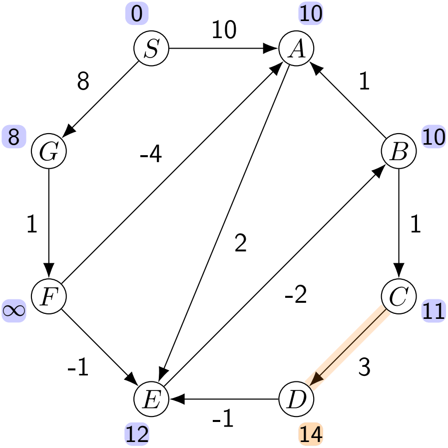


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S,A), (B,A), (B,C), (C,D), (D,E), (F,E), (G,F), (S,G), (F,A), (A,E), (E,B)

Round: 2

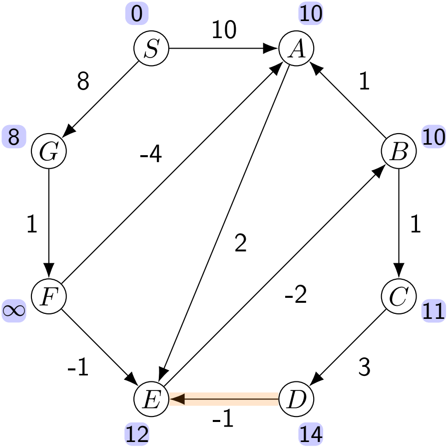


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

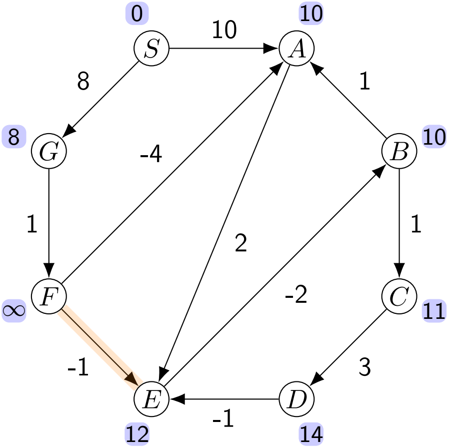


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

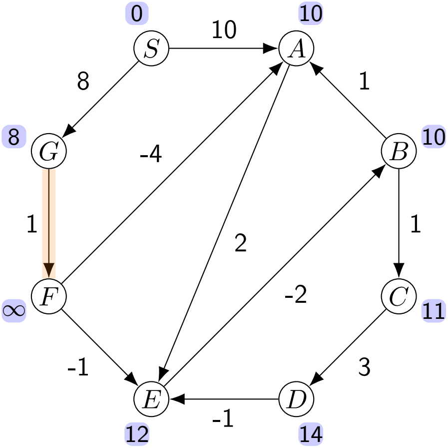


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

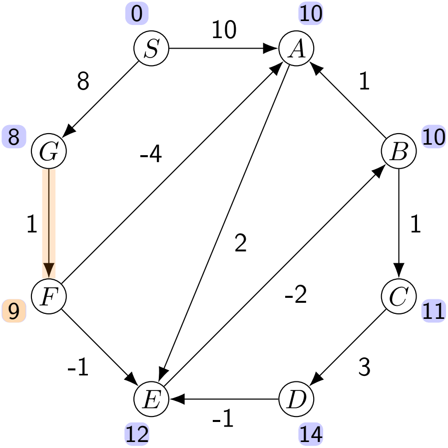


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

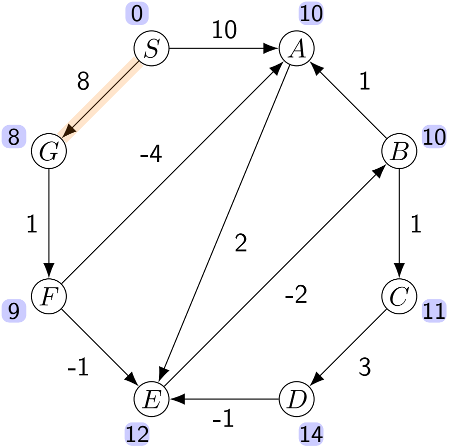


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

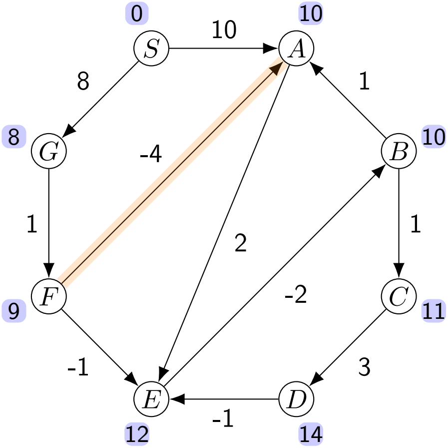


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

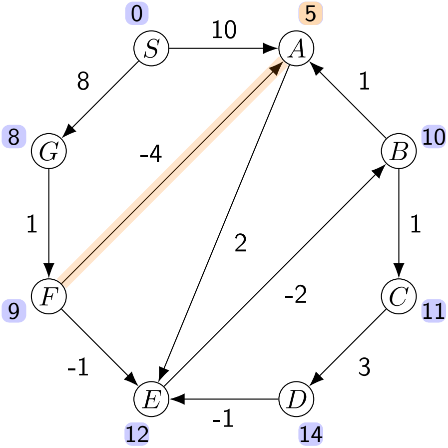


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

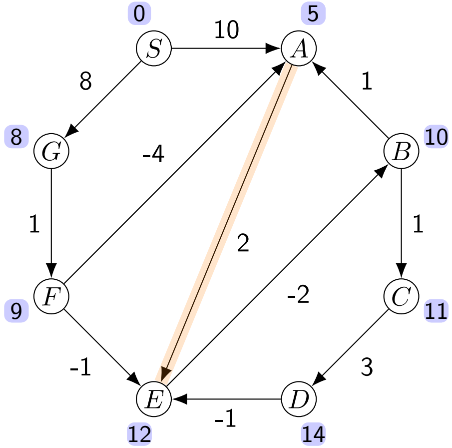


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

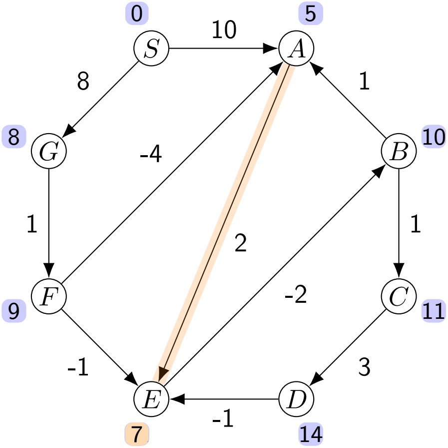


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

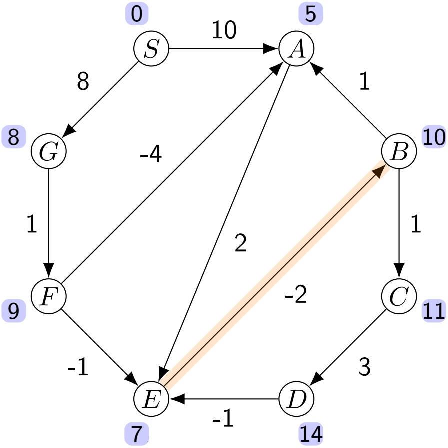


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

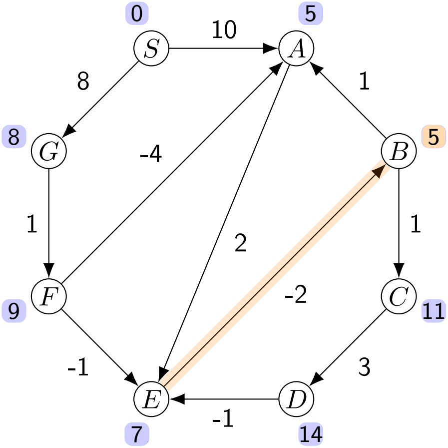


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 2

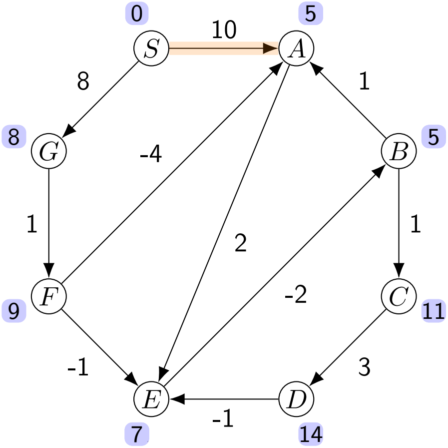


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

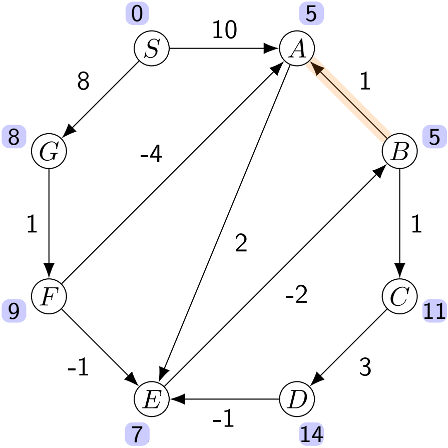


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

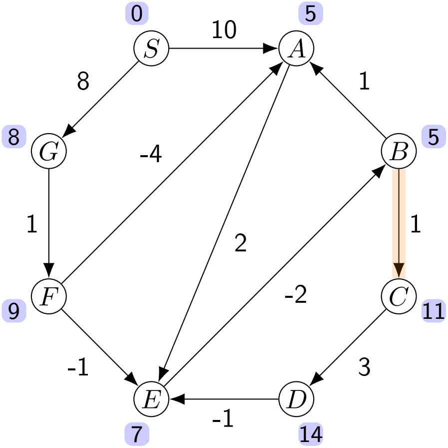


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

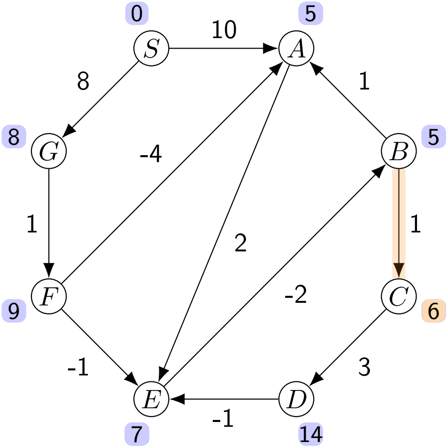


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

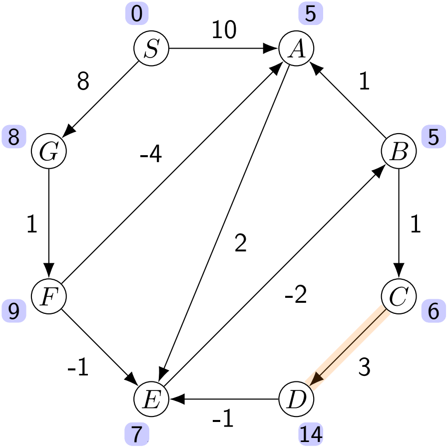


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S,A), (B,A), (B,C), (C,D), (D,E), (F,E), (G,F), (S,G), (F,A), (A,E), (E,B)

Round: 3

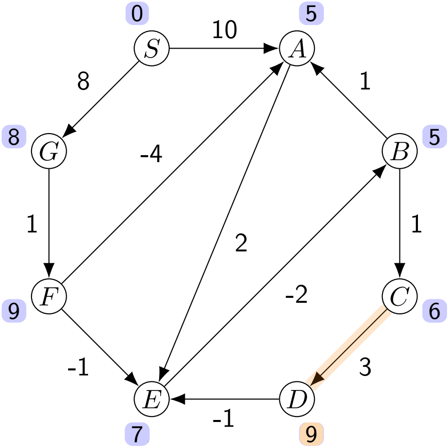


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

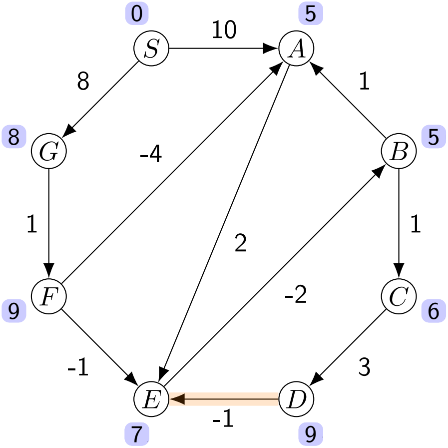


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

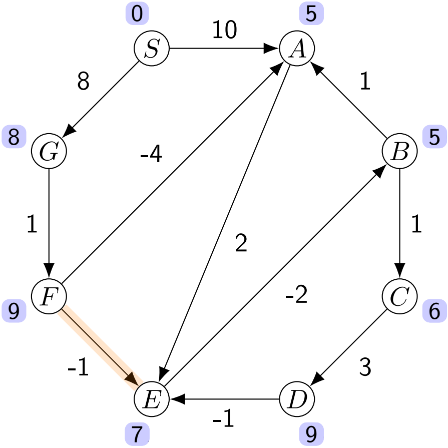


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

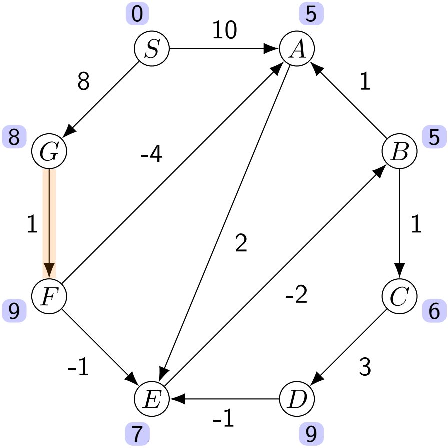


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

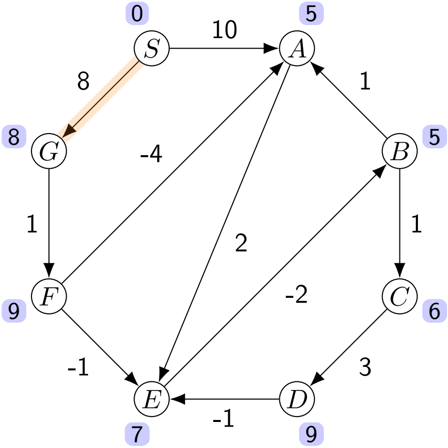


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

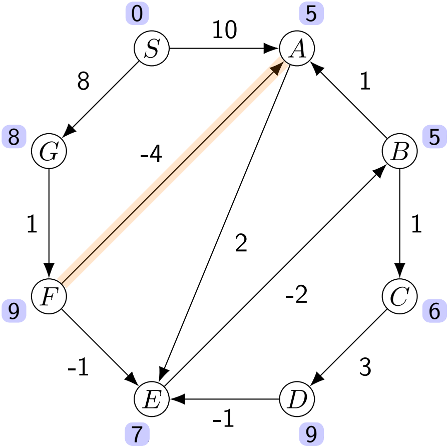


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

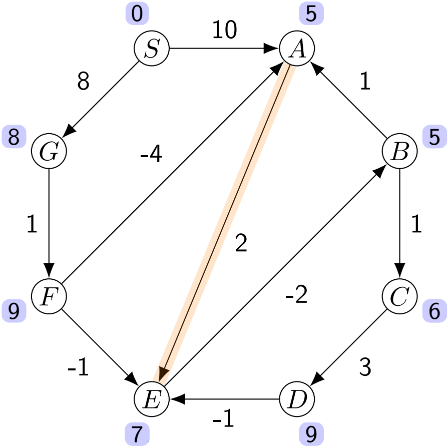


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

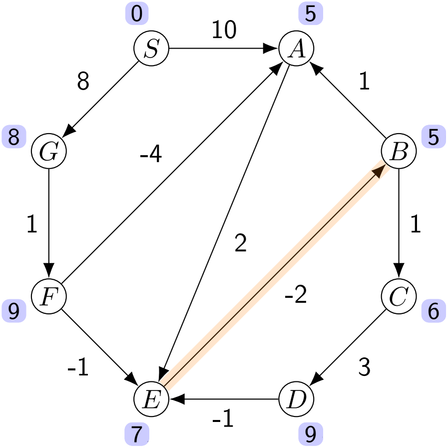


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 3

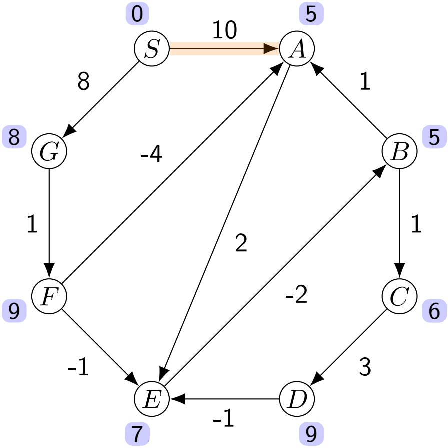


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

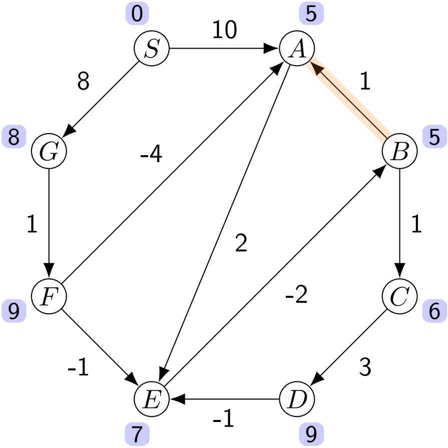


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

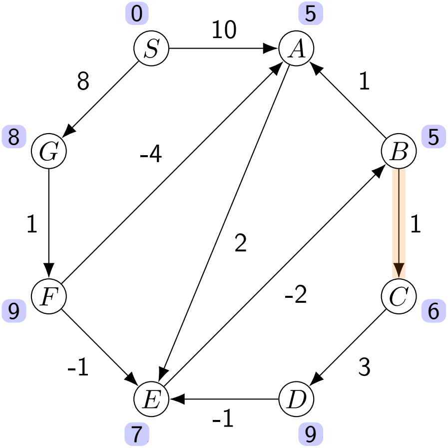


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

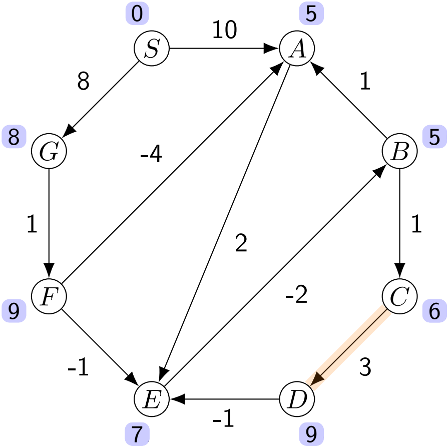


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

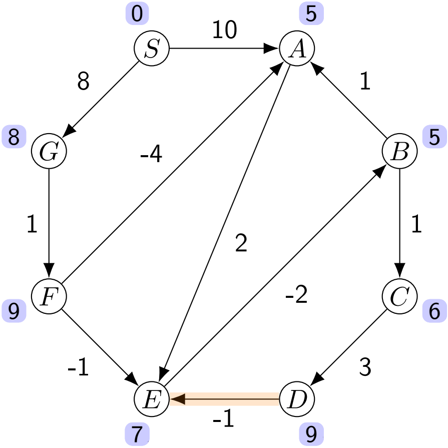


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

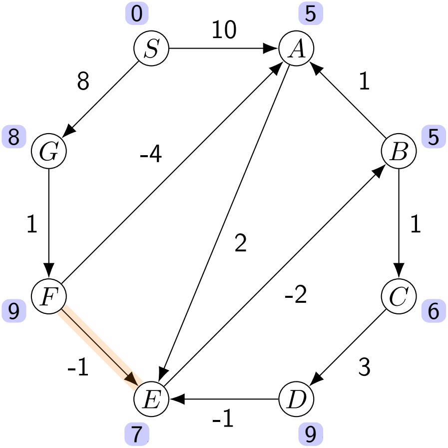


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

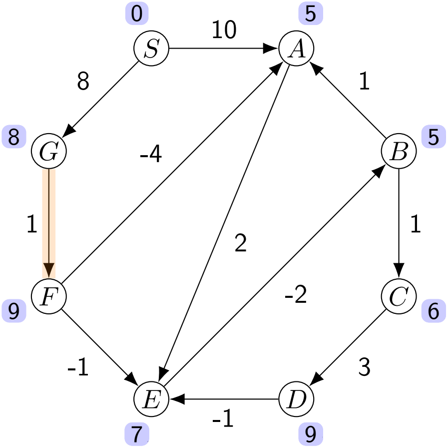


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

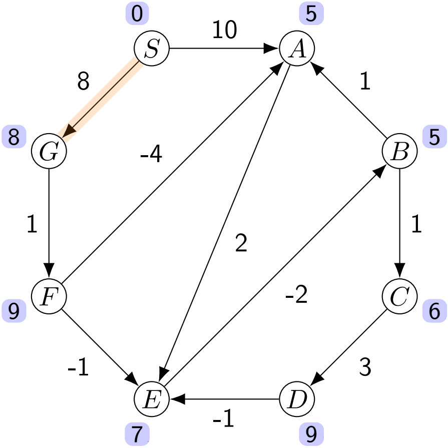


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

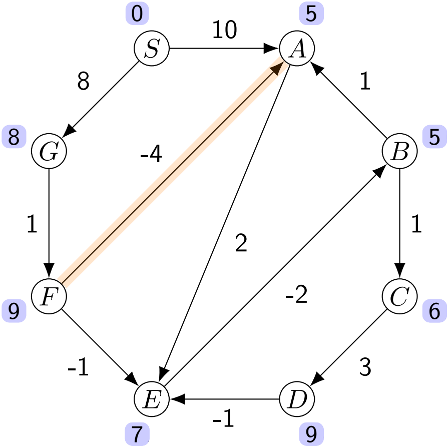


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

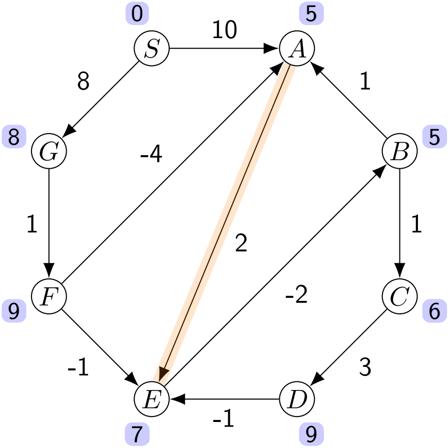


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S,A), (B,A), (B,C), (C,D), (D,E), (F,E), (G,F), (S,G), (F,A), (A,E), (E,B)

Round: 4

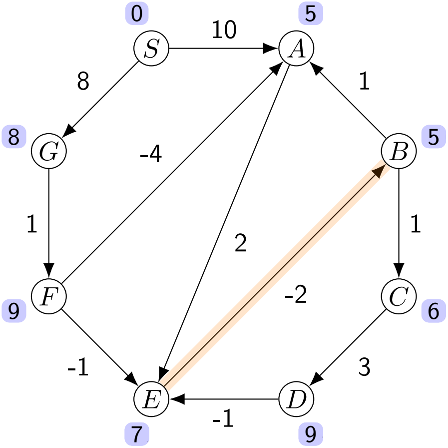


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4

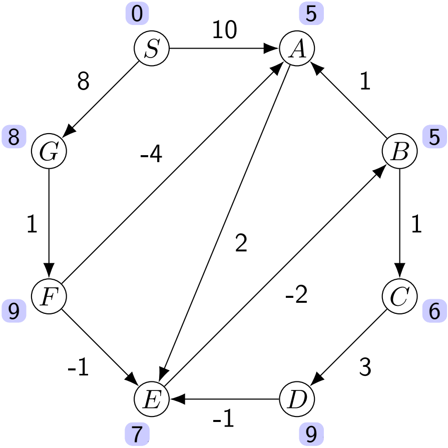


Bellman-Ford algorithm

Consider the following example:

Assume we Update edges in the order (S, A), (B, A), (B, C), (C, D), (D, E), (F, E), (G, F), (S, G), (F, A), (A, E), (E, B)

Round: 4



Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.
- JT Fineman improved it to $\tilde{O}\left(|V|^{\frac{8}{9}}|E|\right)$ in STOC 2024.

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.
- JT Fineman improved it to $\tilde{O}\left(|V|^{\frac{8}{9}}|E|\right)$ in STOC 2024.

What happens if we run $|V|$ iterations?

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.
- JT Fineman improved it to $\tilde{O}\left(|V|^{\frac{8}{9}}|E|\right)$ in STOC 2024.

What happens if we run $|V|$ iterations?

- Shouldn't be any Update in the last round...

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.
- JT Fineman improved it to $\tilde{O}\left(|V|^{\frac{8}{9}}|E|\right)$ in STOC 2024.

What happens if we run $|V|$ iterations?

- Shouldn't be any Update in the last round... unless there are negative cycles!

Bellman-Ford algorithm

Can we do better?

- Can stop earlier if no Update happened in the previous iteration.
- Can skip edges from a node whose $dist$ was not Updated in the previous iteration.
- Worst-case remains $O(|V| \cdot |E|)$.
- JT Fineman improved it to $\tilde{O}\left(|V|^{\frac{8}{9}}|E|\right)$ in STOC 2024.

What happens if we run $|V|$ iterations?

- Shouldn't be any Update in the last round... unless there are negative cycles!
- This can be used to detect negative cycles in a graph.

All pairs shortest path

What if we need pairwise shortest paths, not just from s?

All pairs shortest path

What if we need pairwise shortest paths, not just from s?

- We can run Bellman-Ford starting from each vertex. The running time is

All pairs shortest path

What if we need pairwise shortest paths, not just from s?

- We can run Bellman-Ford starting from each vertex. The running time is $O\left(|V|^2 \cdot |E|\right)$, which can be $O\left(|V|^4\right)$ if the graph is dense.

All pairs shortest path

What if we need pairwise shortest paths, not just from s?

- We can run Bellman-Ford starting from each vertex. The running time is $O\left(|V|^2 \cdot |E|\right)$, which can be $O\left(|V|^4\right)$ if the graph is dense.

Can we do better?

All pairs shortest path

- To store pairwise shortest distances, we need to upgrade our $dist[\cdot]$ array to a $dist[\cdot,\cdot]$ matrix.

All pairs shortest path

- To store pairwise shortest distances, we need to upgrade our $dist[\cdot]$ array to a $dist[\cdot,\cdot]$ matrix.
- Initially, $dist[a,b] = \begin{cases} \ell(a,b) & \text{if } (a,b) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$.

All pairs shortest path

- To store pairwise shortest distances, we need to upgrade our $dist[\cdot]$ array to a $dist[\cdot,\cdot]$ matrix.
- Initially, $dist[a,b] = \begin{cases} \ell(a,b) & \text{if } (a,b) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.

All pairs shortest path

- To store pairwise shortest distances, we need to upgrade our $dist[\cdot]$ array to a $dist[\cdot,\cdot]$ matrix.
- Initially, $dist[a,b] = \begin{cases} \ell(a,b) & \text{if } (a,b) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.
- Repeating Bellman-Ford fills $dist$ row by row, first compute all correct values for $dist[1,\cdot]$, then for $dist[2,\cdot]$, ..., each row takes $O(|V| \cdot |E|)$ time.

All pairs shortest path

- To store pairwise shortest distances, we need to upgrade our $dist[\cdot]$ array to a $dist[\cdot,\cdot]$ matrix.
- Initially, $dist[a,b] = \begin{cases} \ell(a,b) & \text{if } (a,b) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.
- Repeating Bellman-Ford fills $dist$ row by row, first compute all correct values for $dist[1,\cdot]$, then for $dist[2,\cdot]$, ..., each row takes $O(|V| \cdot |E|)$ time.
- Can we use information from other rows to speed things up?