

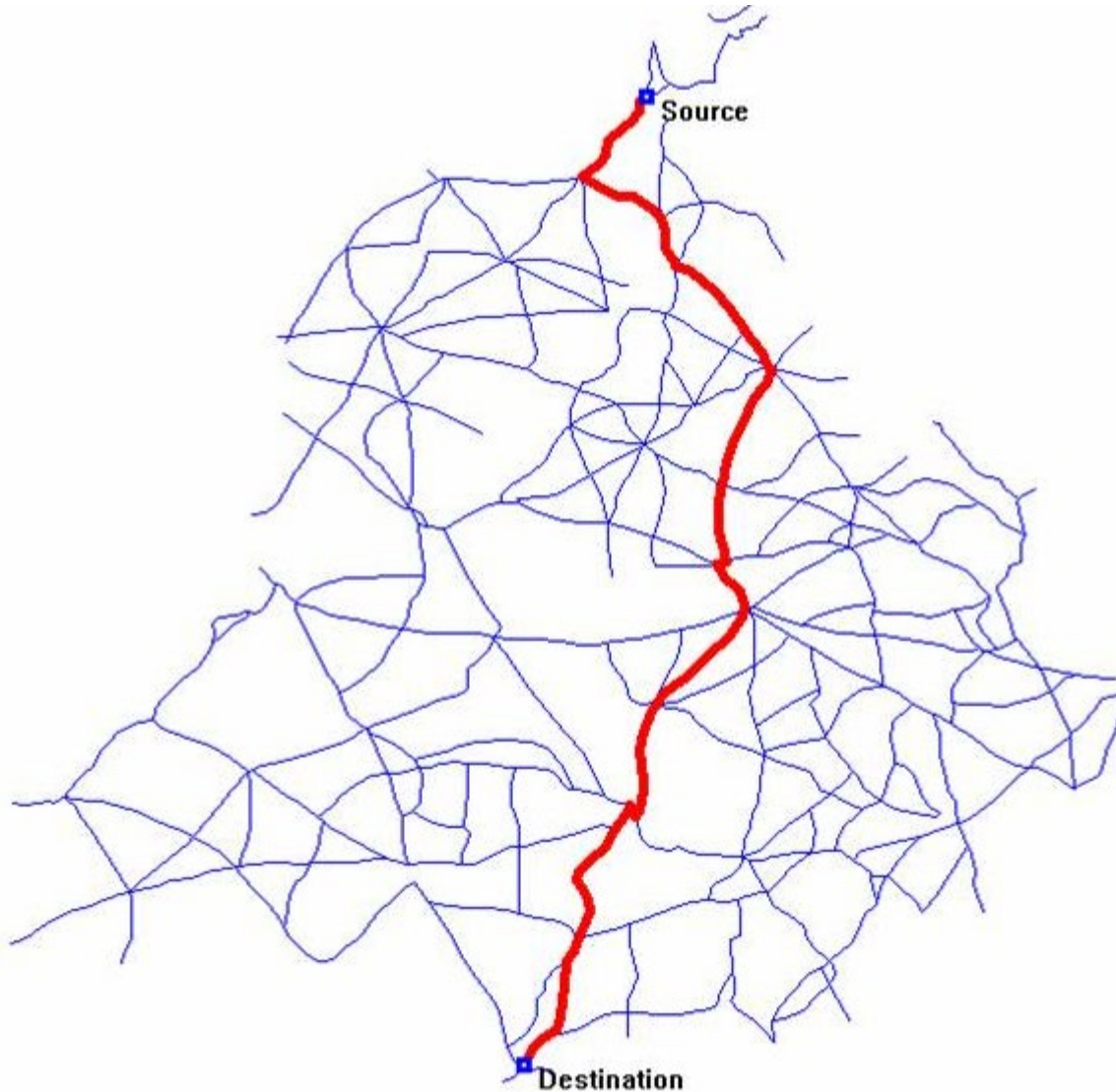
# Roadmap

---

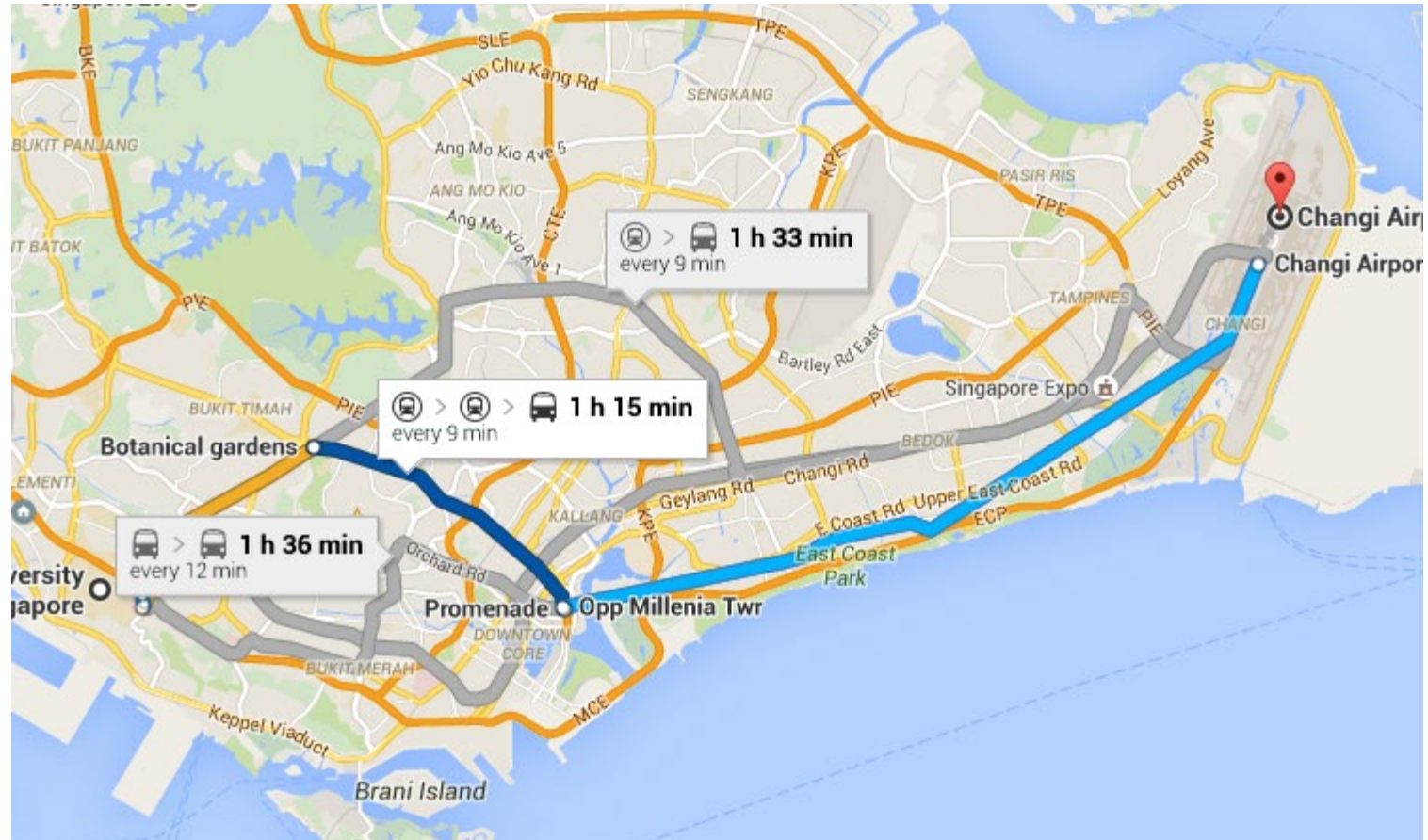
## Shortest Paths

- The SSSP Problem
- Bellman-Ford

# SHORTEST PATHS

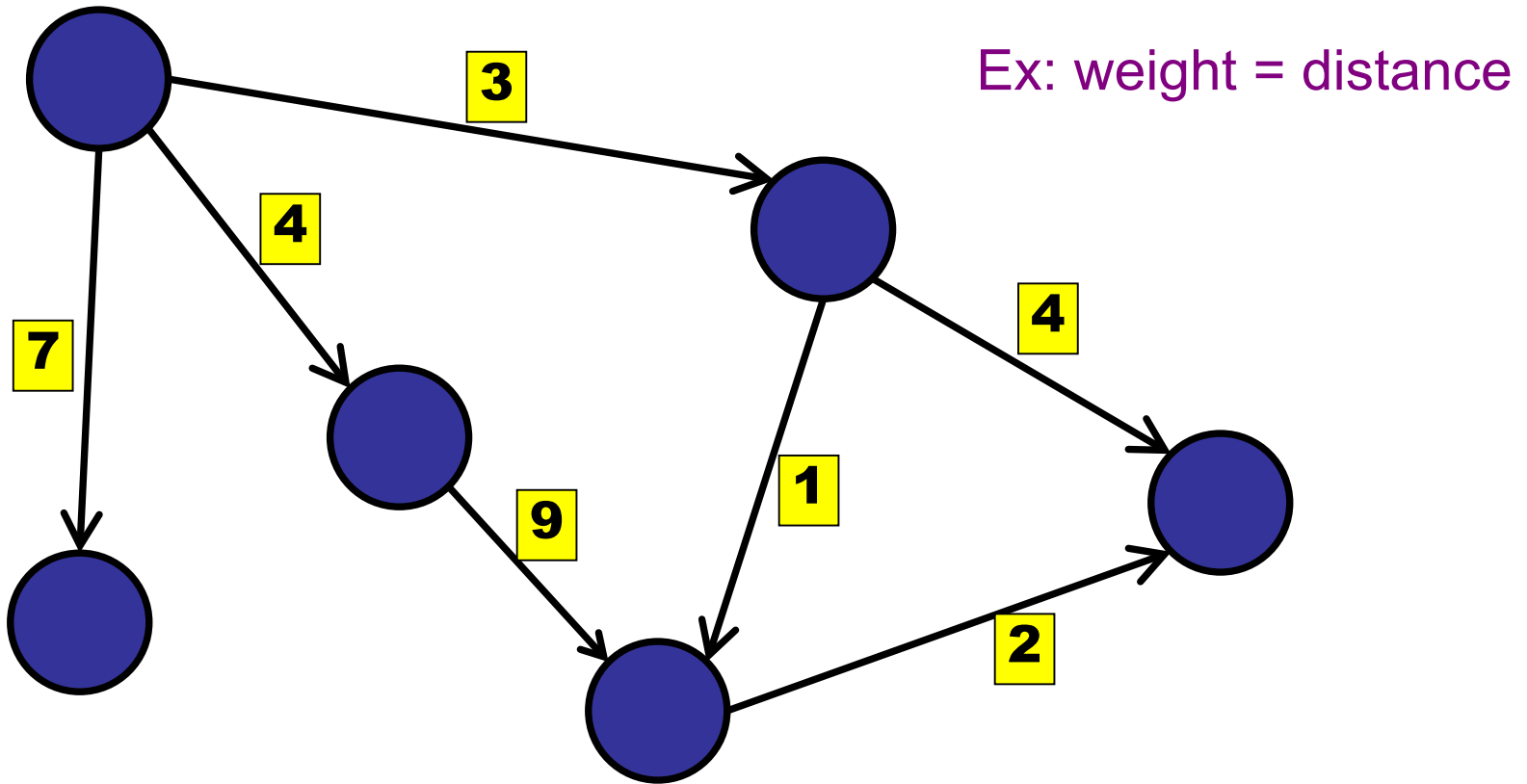


# SHORTEST PATHS



# Weighted Graphs

**Edge weights:**  $w(e) : E \rightarrow \mathbb{R}$

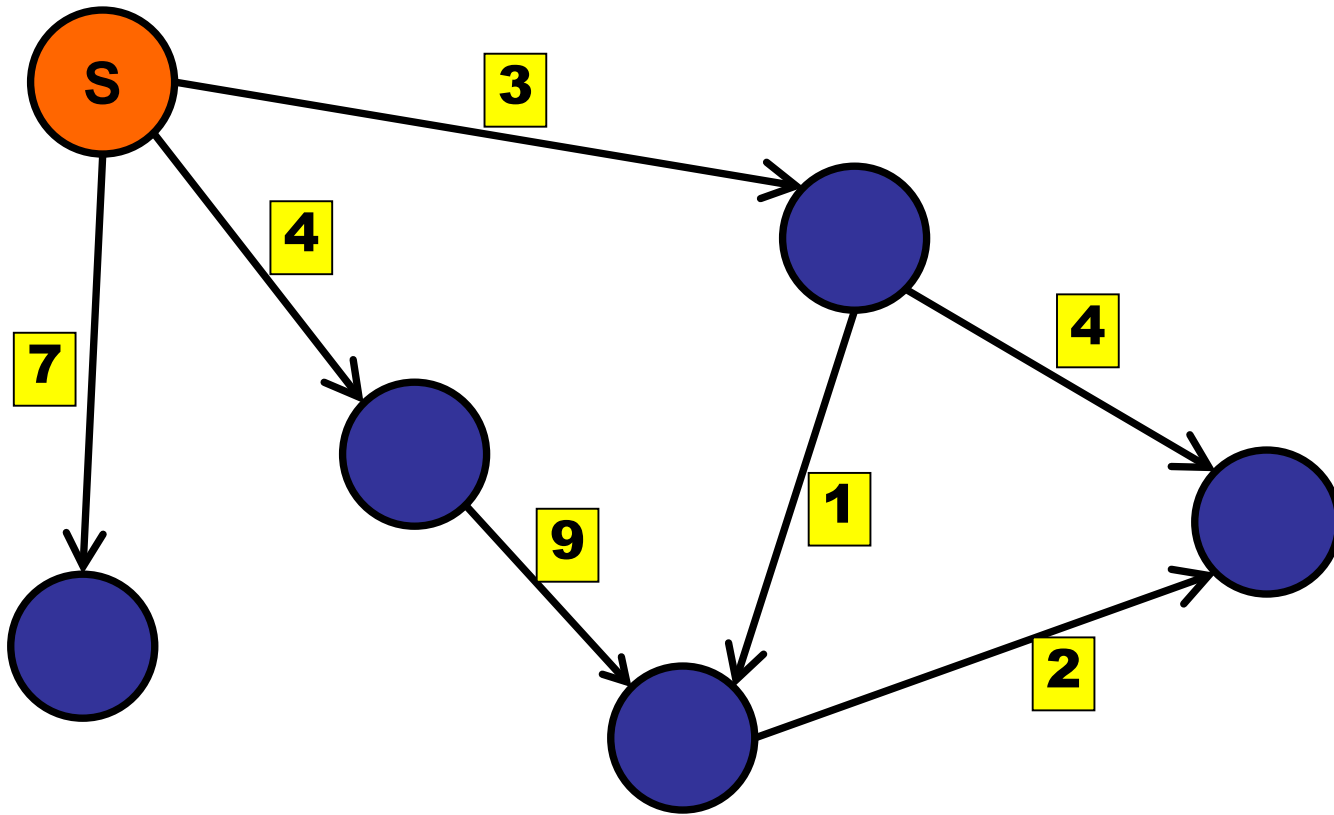


Adjacency list: stores weights with edge in NbrList

# Shortest Paths

---

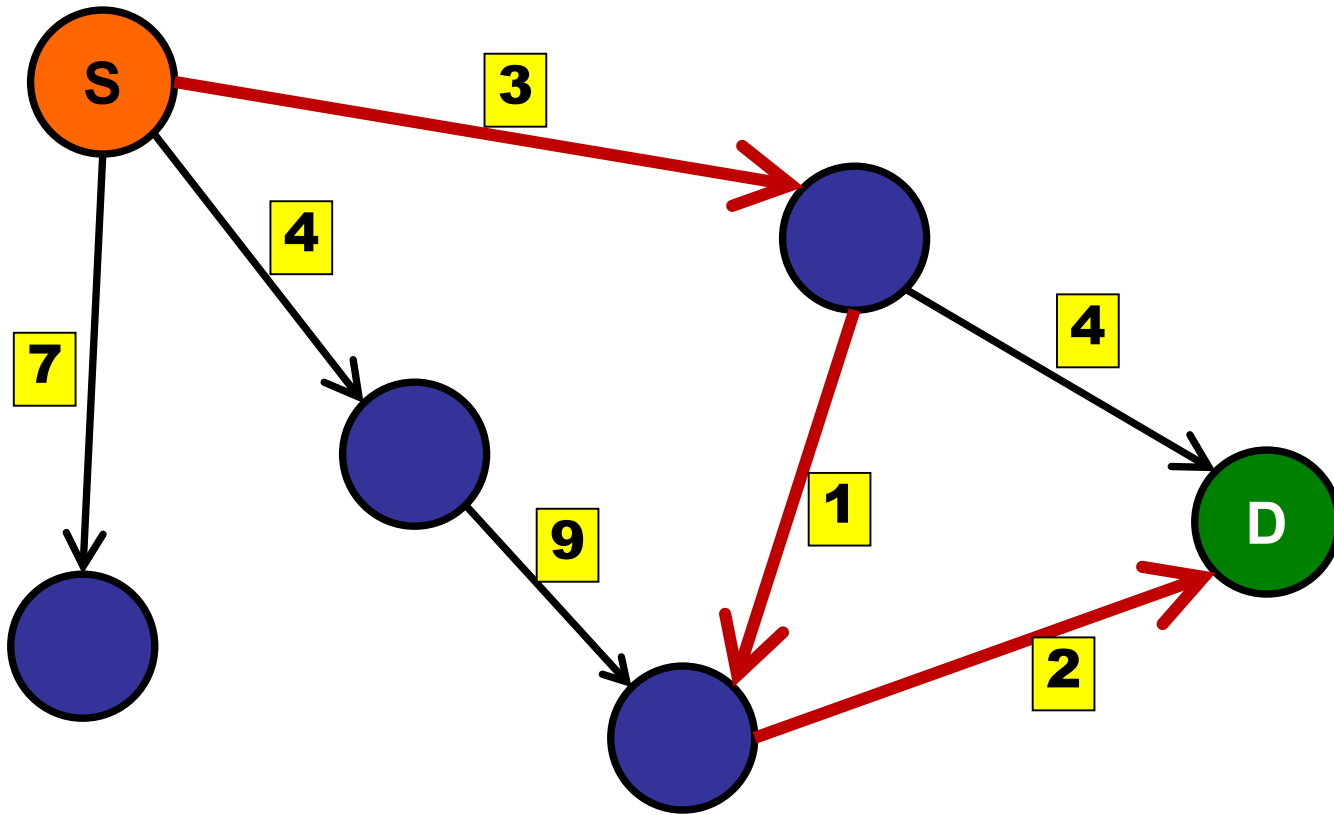
Distance from source?



# Shortest Paths

---

Distance from source?

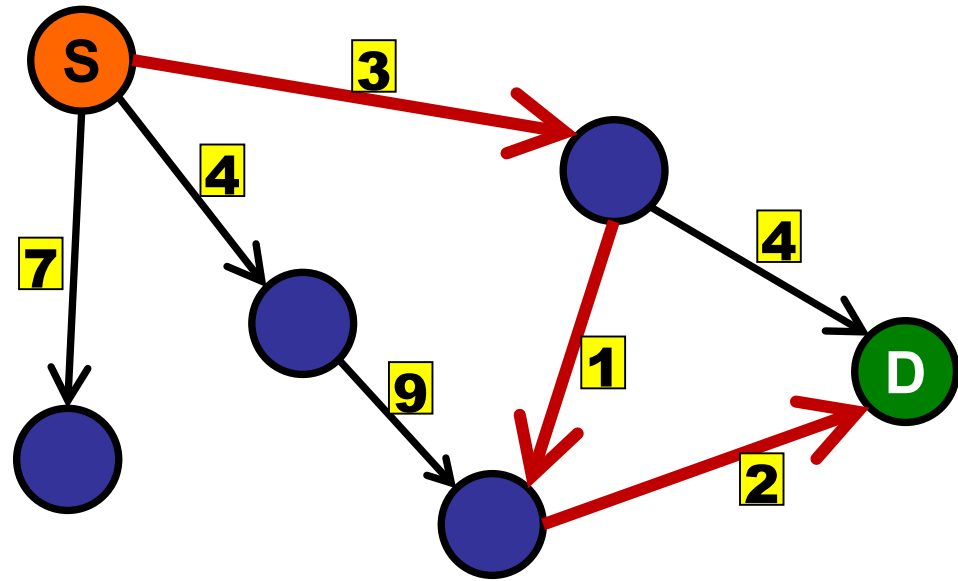


# Shortest Paths

---

## Questions:

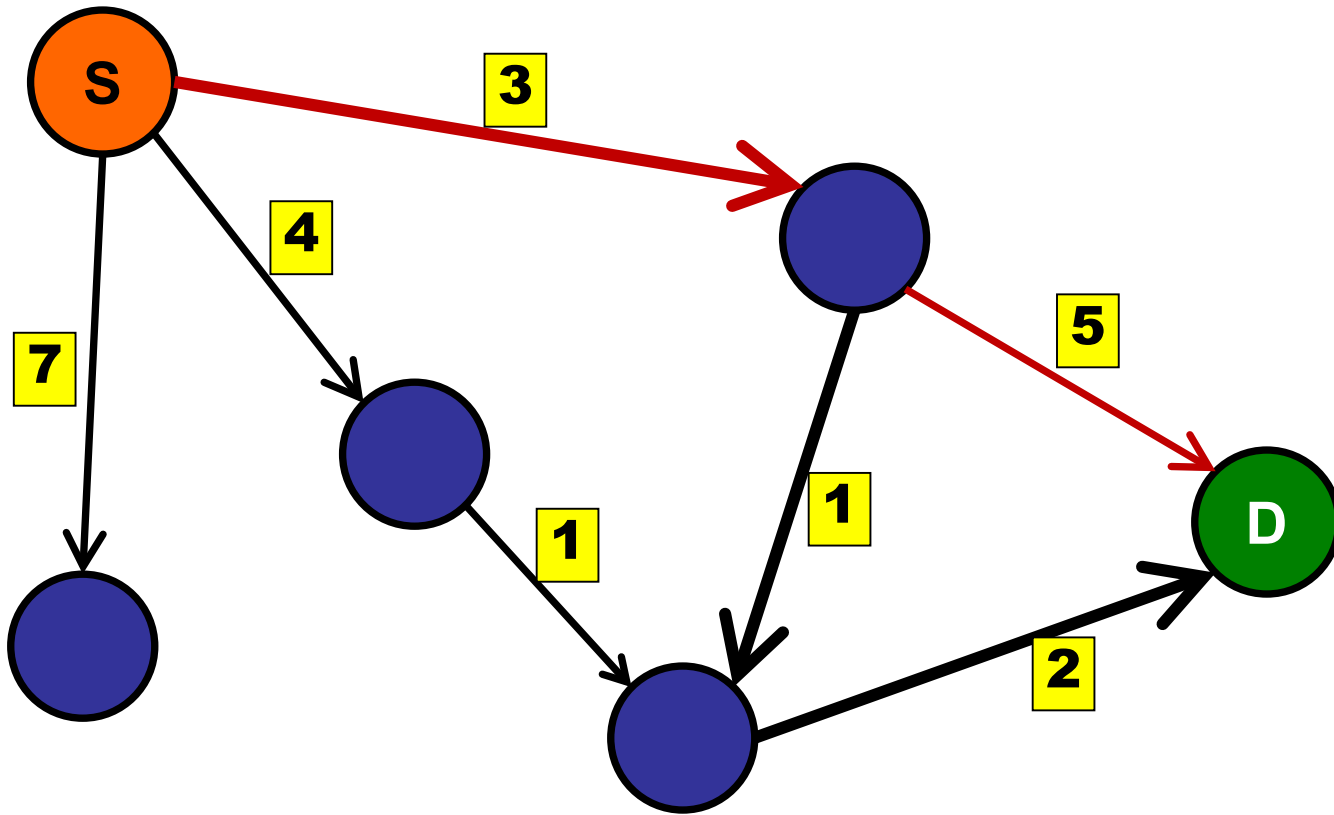
- How far is it from S to D?
- What is the shortest path from S to D?
- Find the shortest path from S to every node.
- Find the shortest path between every pair of nodes.



# Shortest Paths

---

Common mistake: "Why can't I use BFS?"

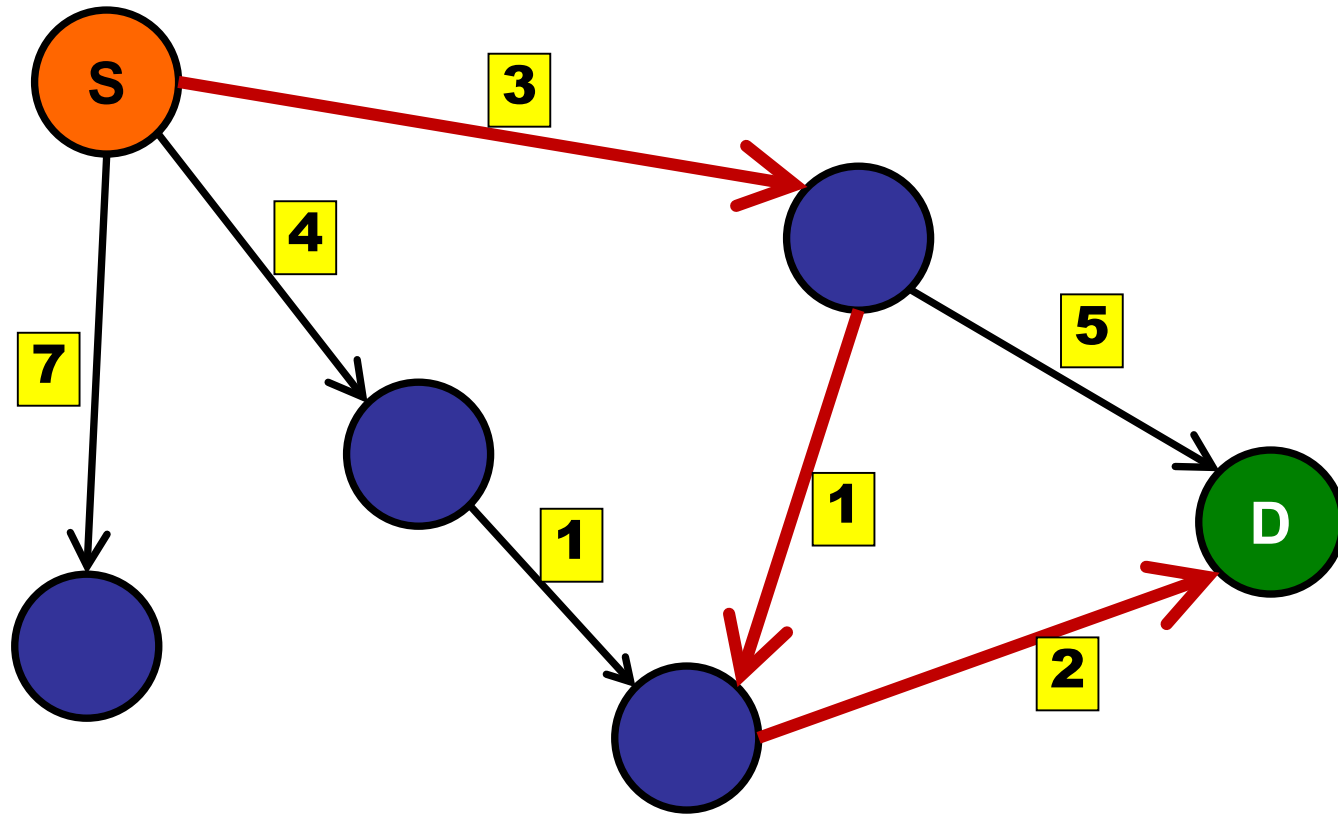




# Shortest Paths

---

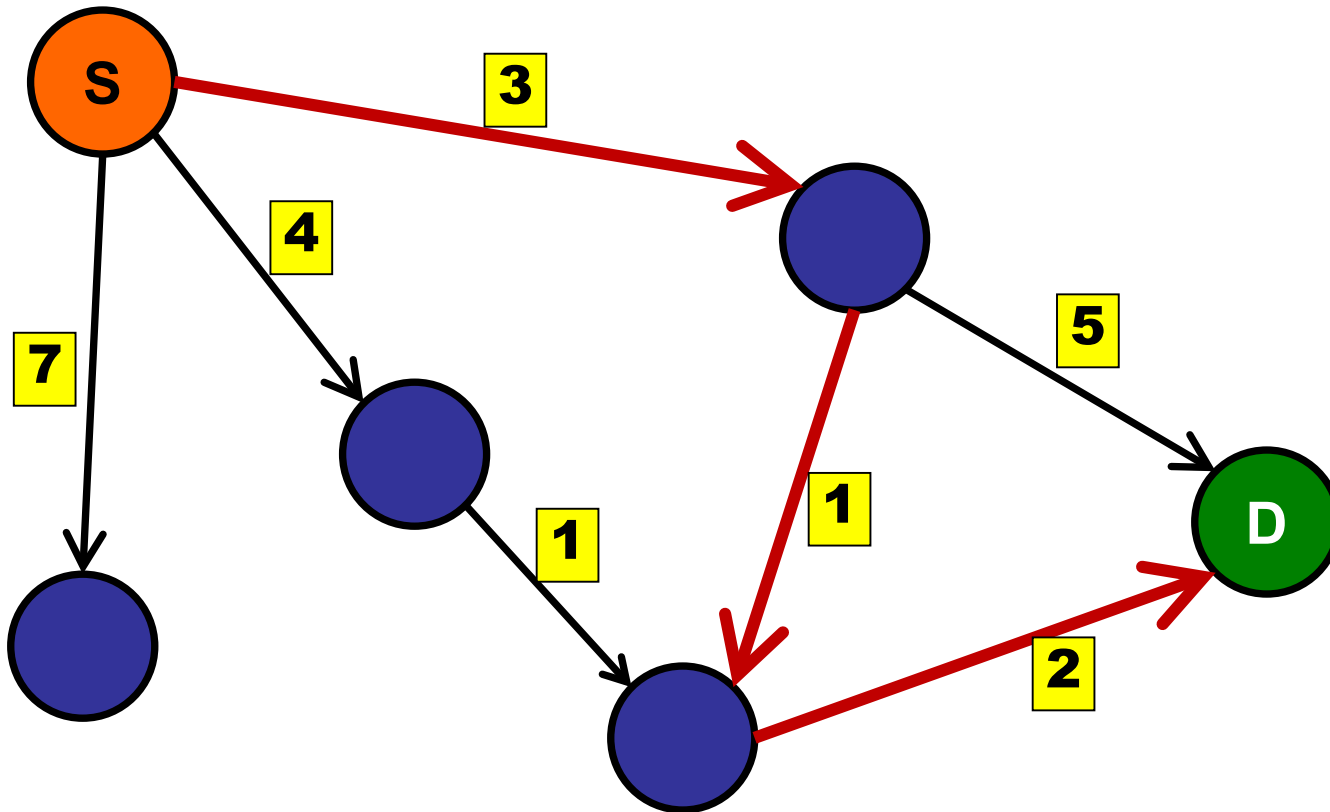
Common mistake: "Why can't I use BFS?"



# Shortest Paths

---

Common mistake: "Why can't I use BFS?"

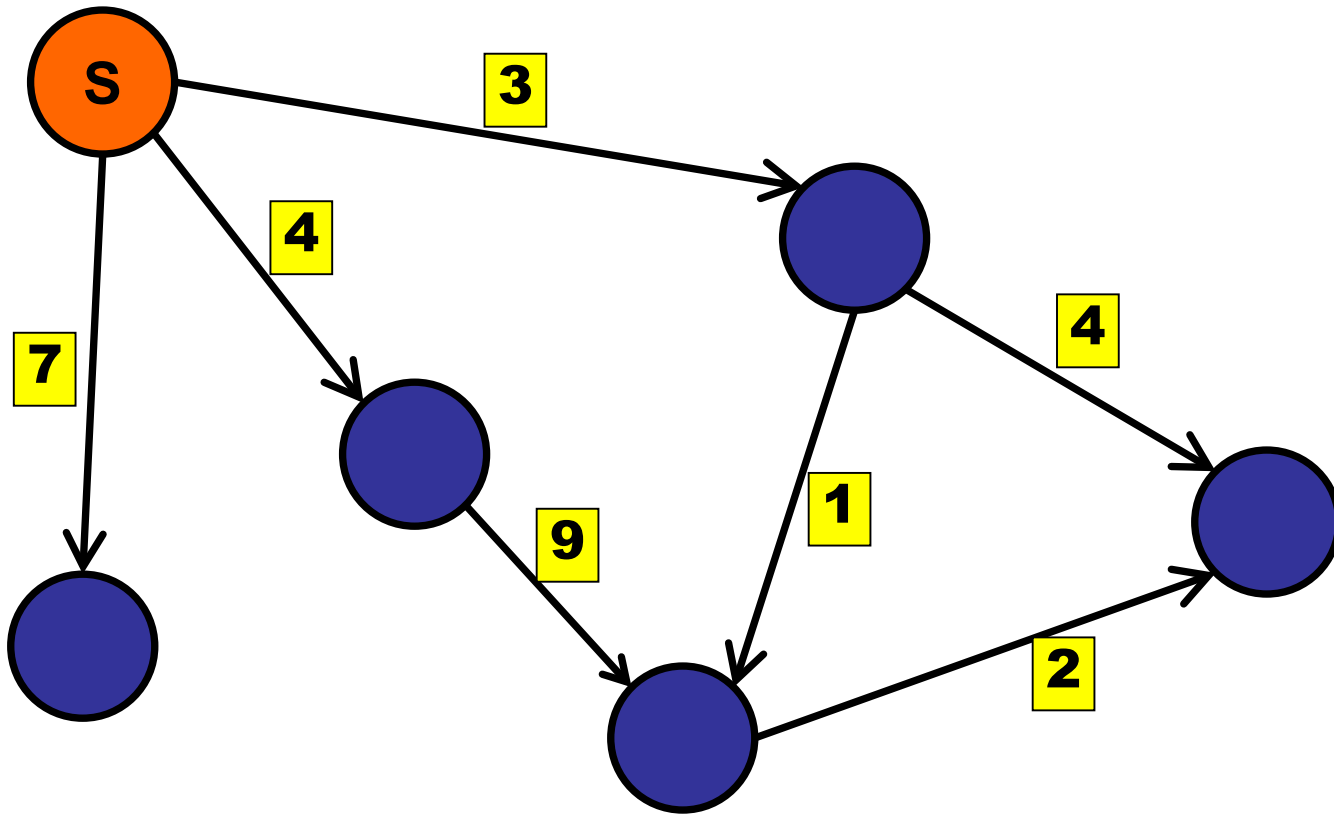


BFS finds minimum number of **HOPS** not minimum **DISTANCE**.

# Shortest Paths

---

Notation:  $\delta(u,v)$  = distance from u to v



# Shortest Paths

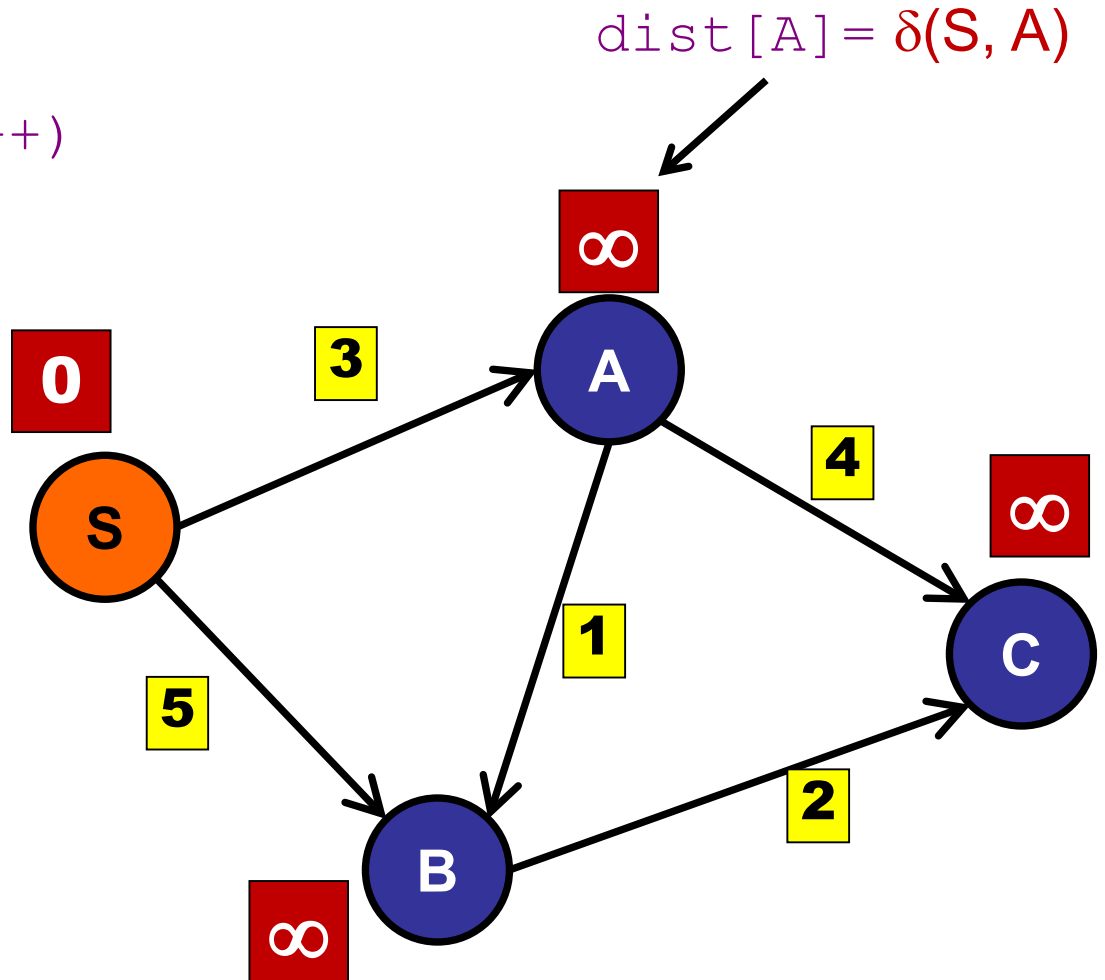
Maintain estimate for each distance:

```
int dist[numNode];
```

```
for(i=0;i<numNode;i++)
```

```
    dist = infinity;
```

```
dist[start] = 0;
```

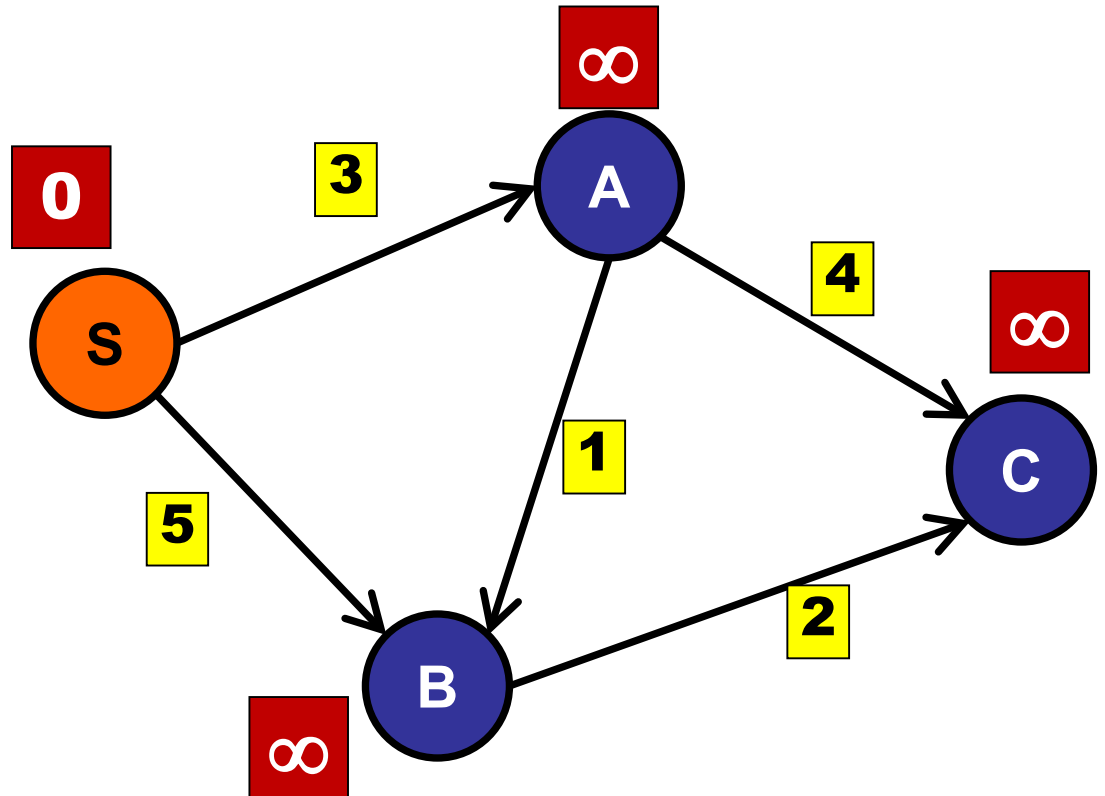


# Shortest Paths

---

Maintain estimate for each distance:

- Reduce estimate
- Invariant: estimate  $\geq$  **actual shortest distance**



# Shortest Paths

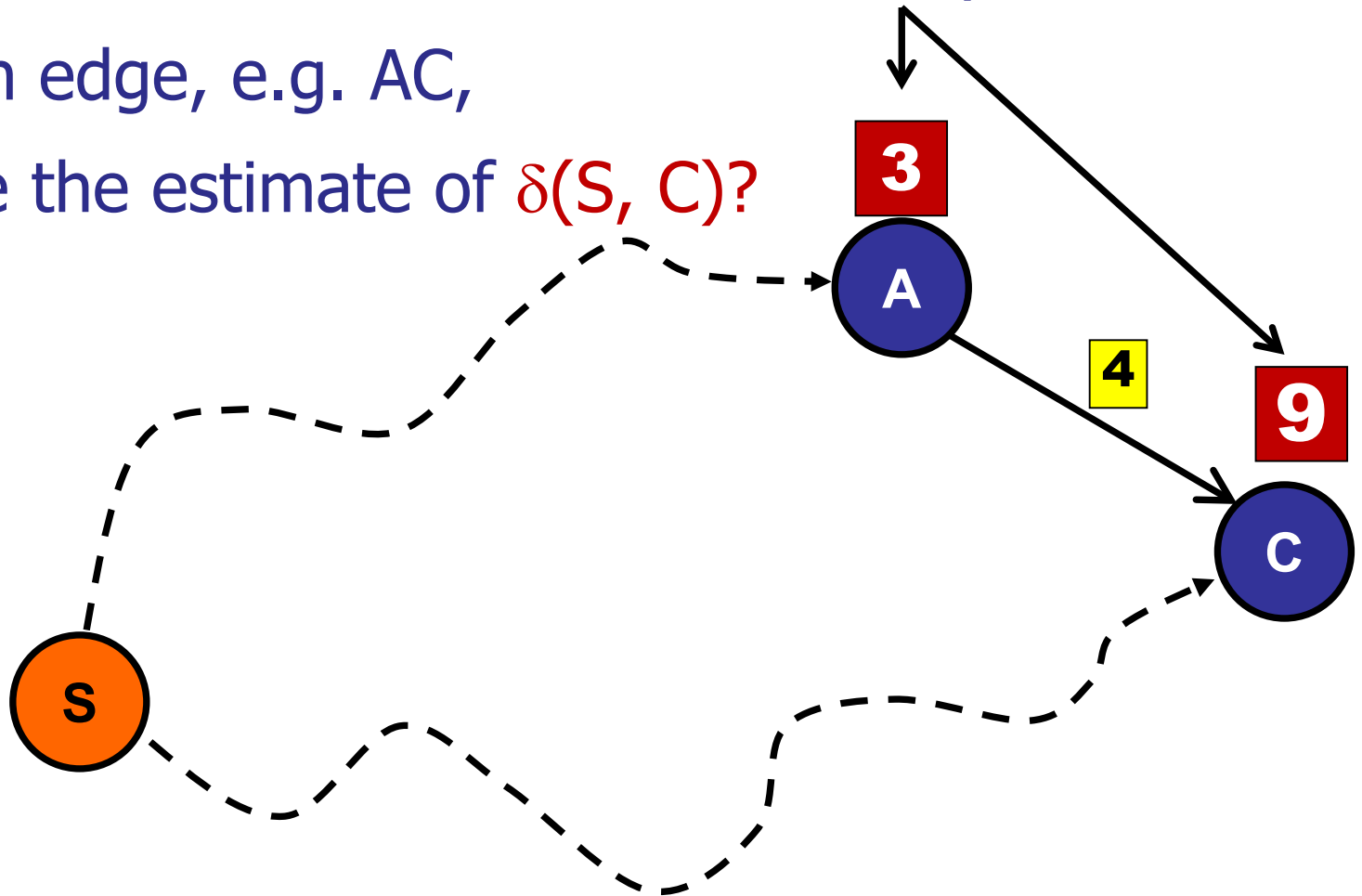
---

Maintain estimate for each distance:

Lets say we have some estimate already

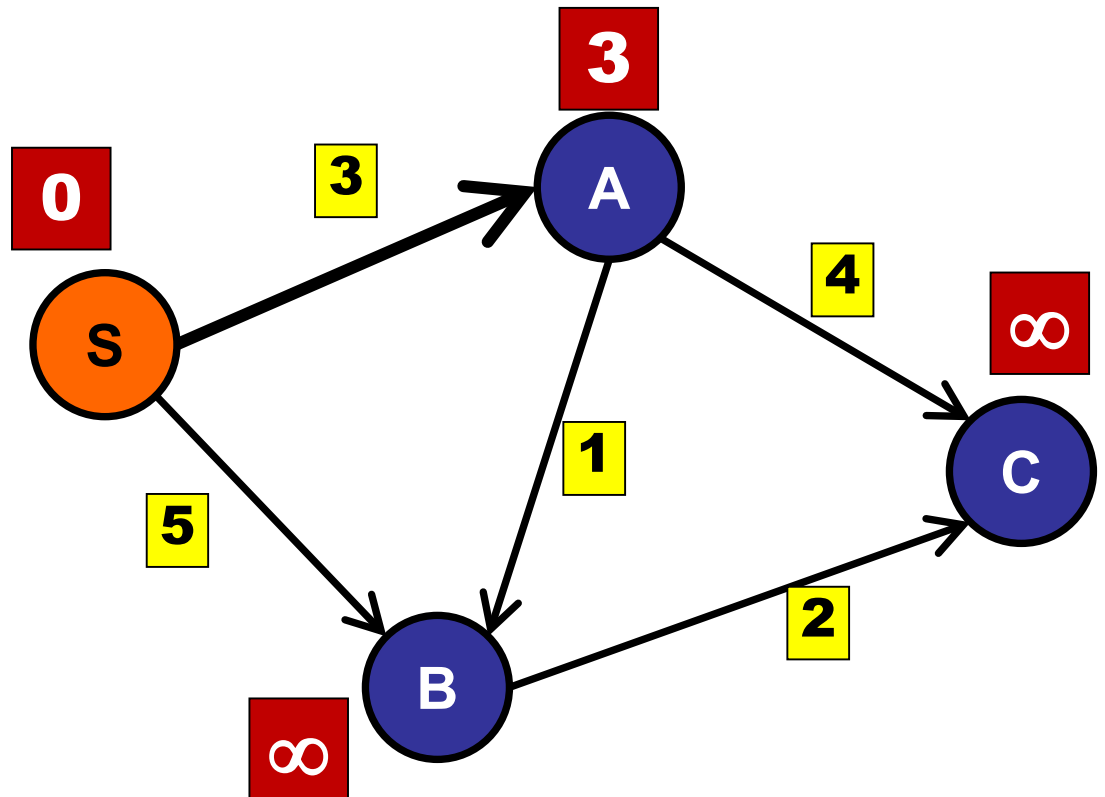
For each edge, e.g. AC,

Improve the estimate of  $\delta(S, C)$ ?



# Shortest Paths

```
relax(int u, int v){  
    if (dist[v] > dist[u] + weight(u,v))  
        dist[v] = dist[u] + weight(u,v);  
}
```

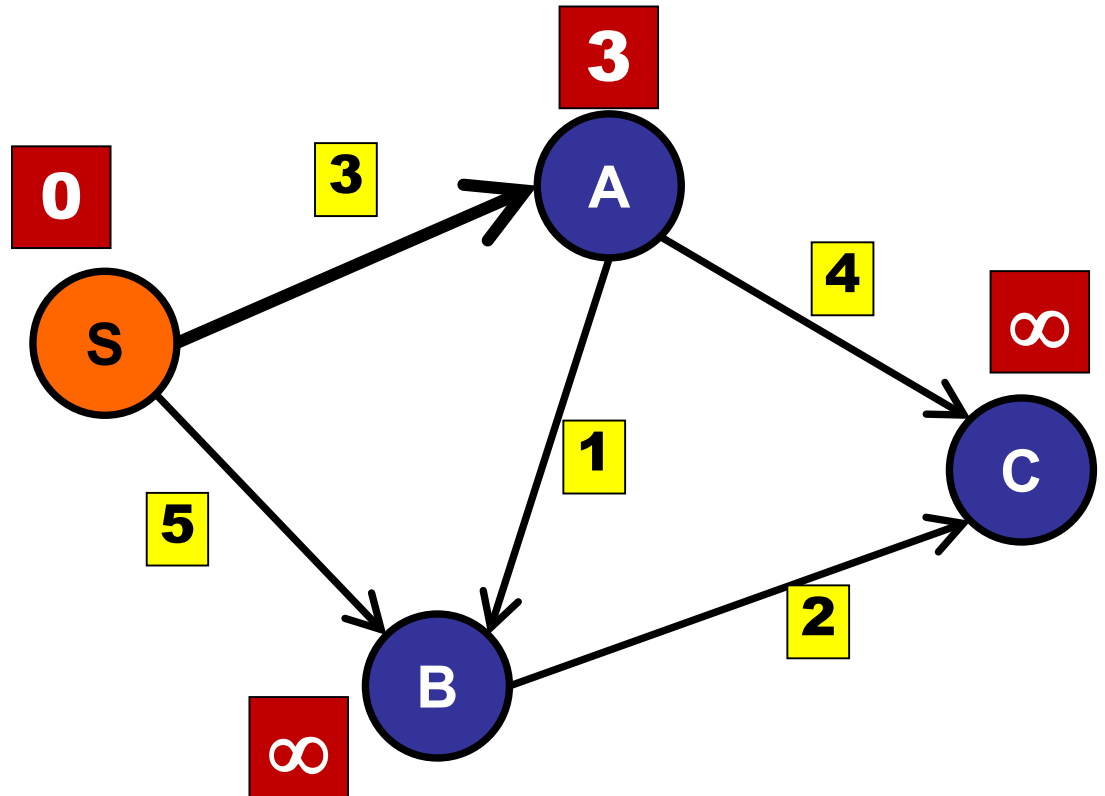


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(S, A)$



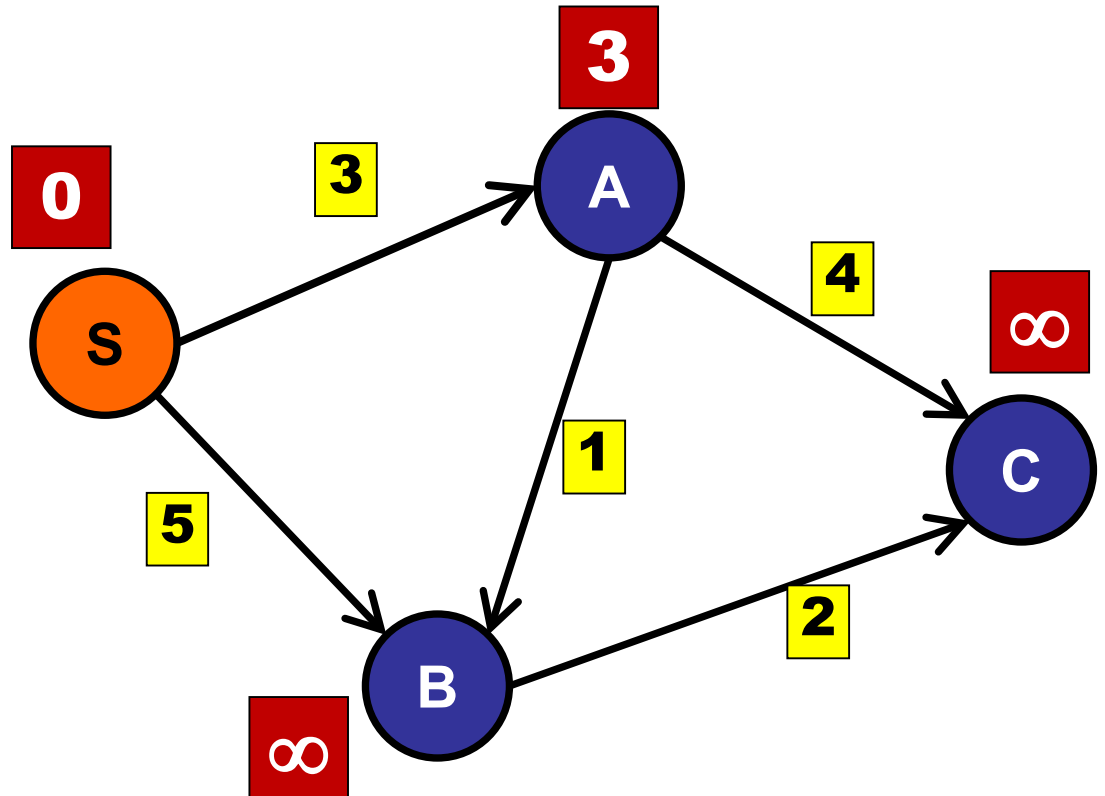


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(A, C)$

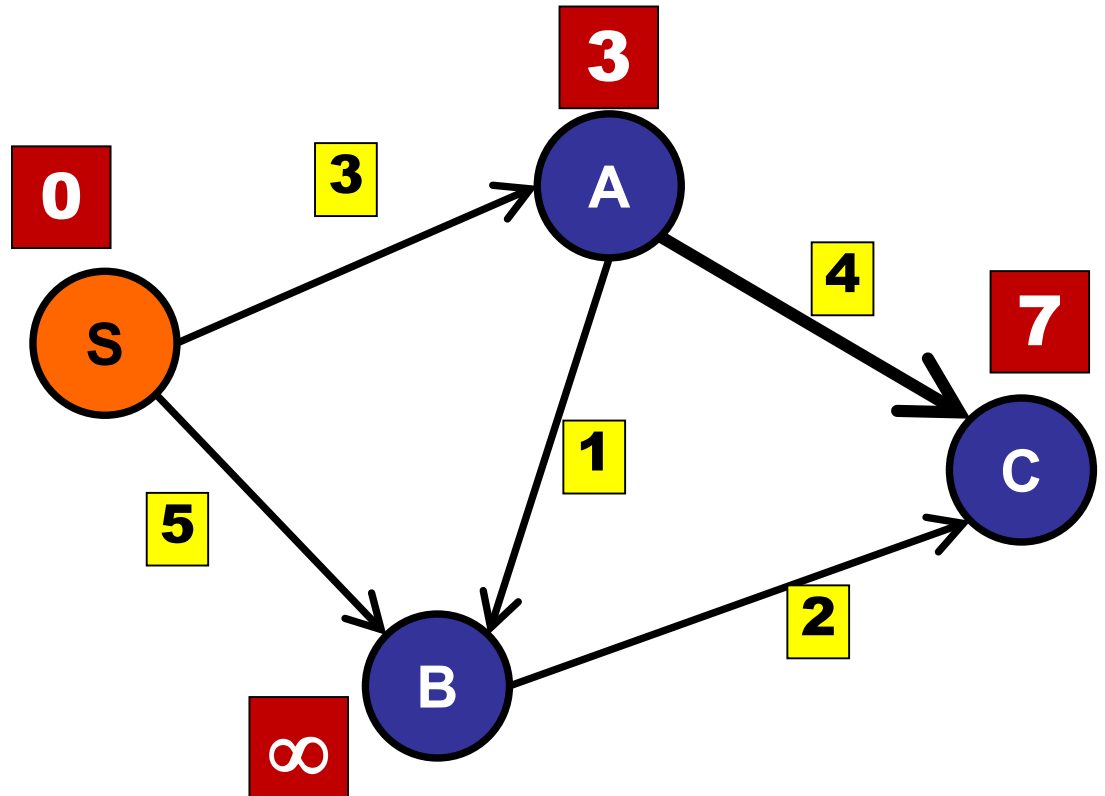


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(A, C)$

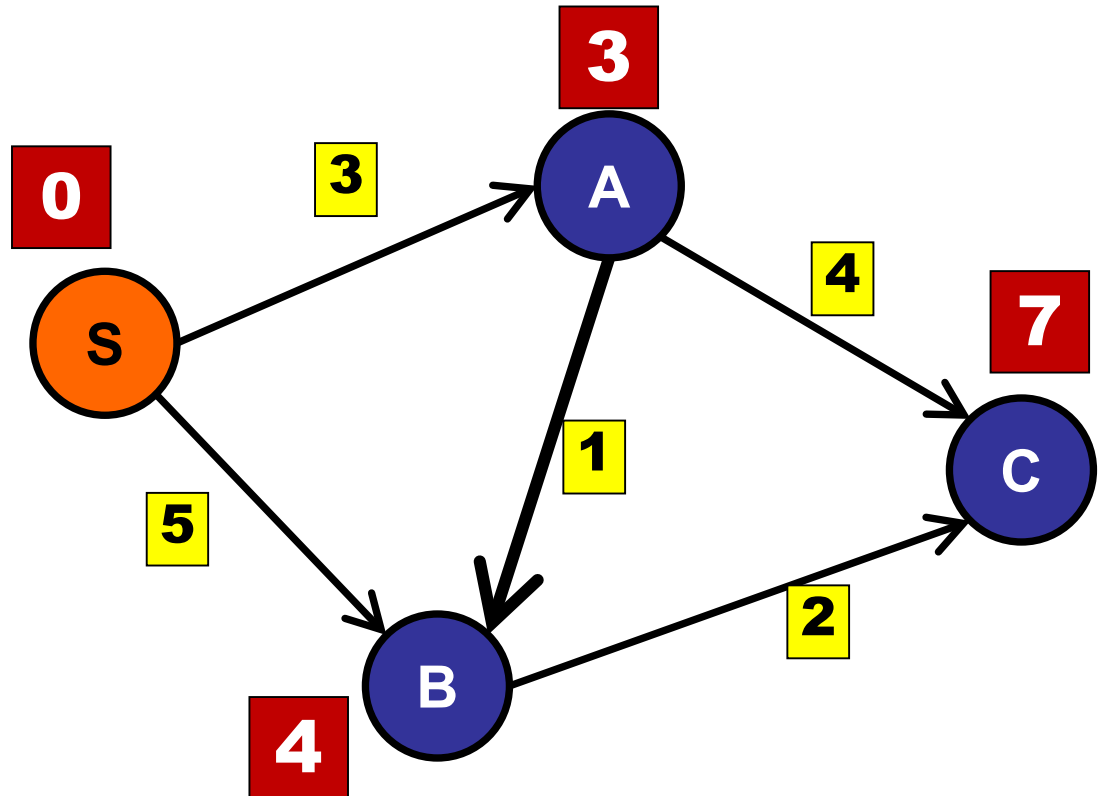


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(A, B)$

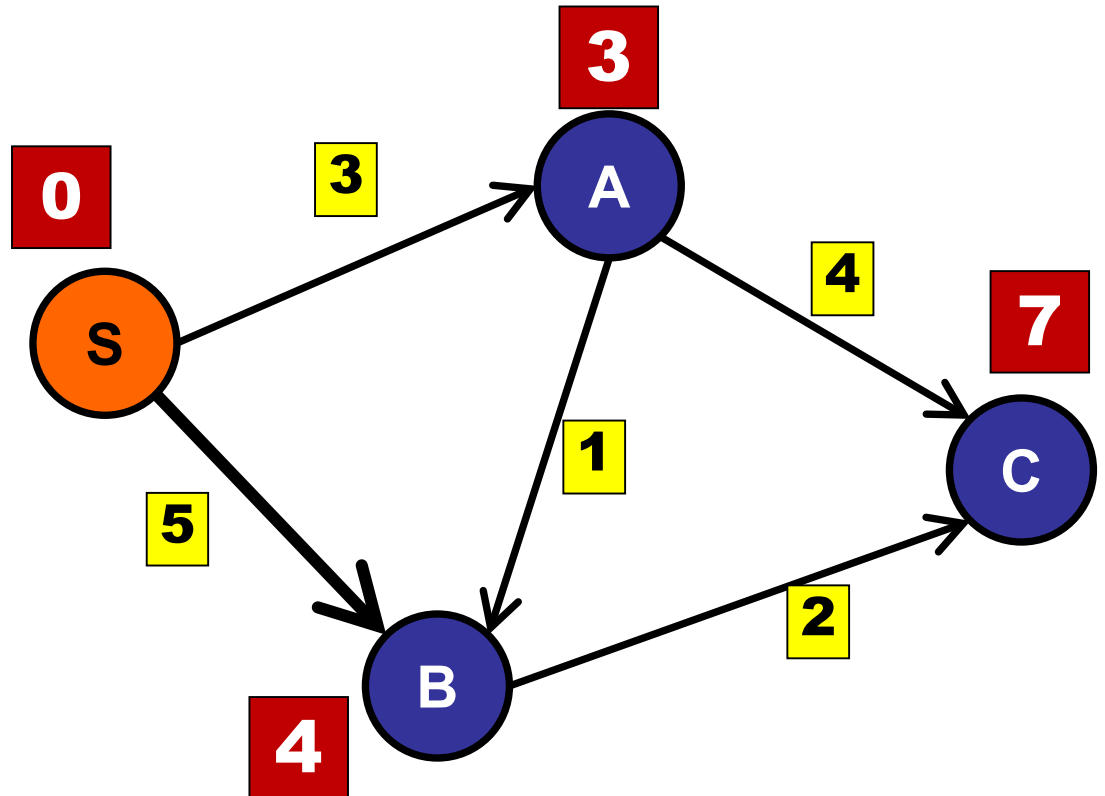


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(S, B)$

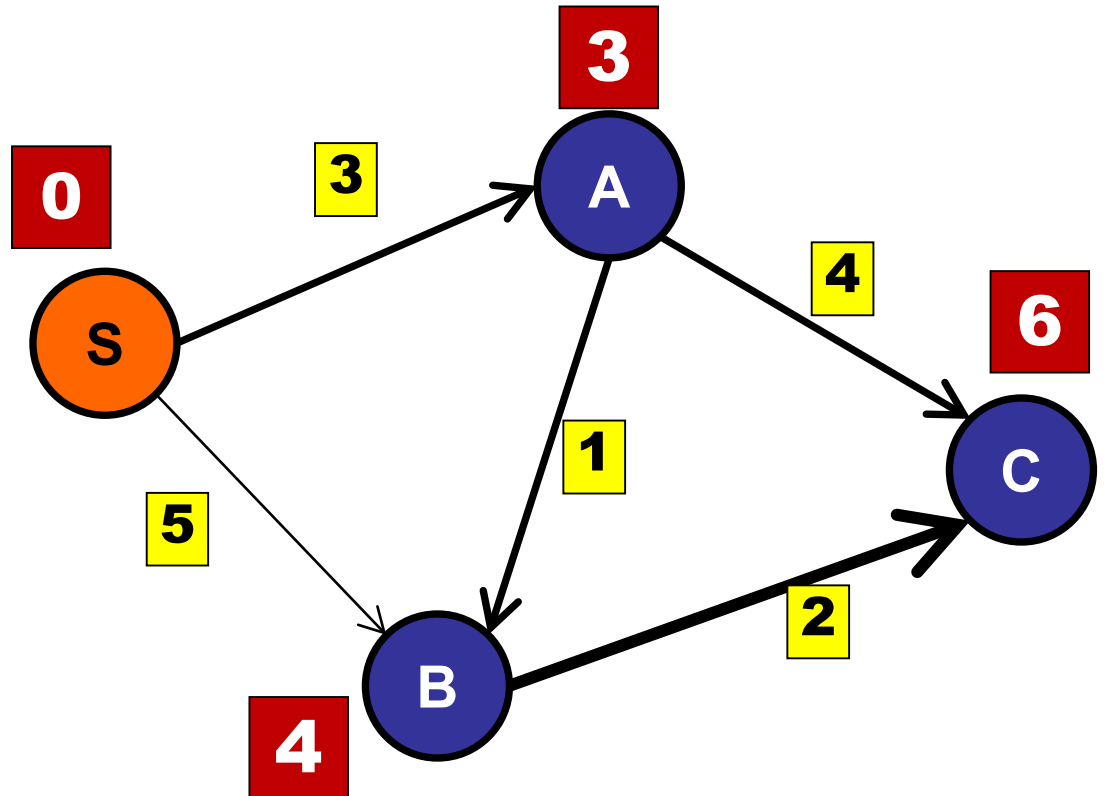


# Shortest Paths

---

Maintain estimate for each distance:

$\text{relax}(B, C)$

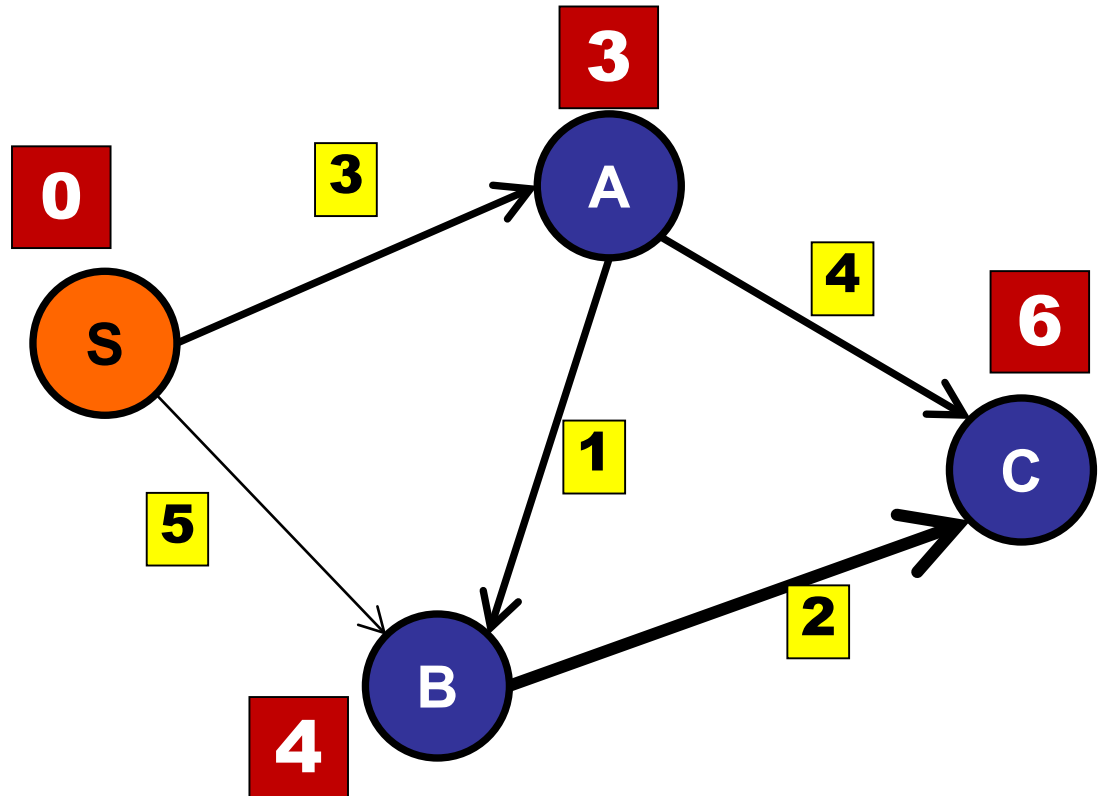


# Shortest Paths

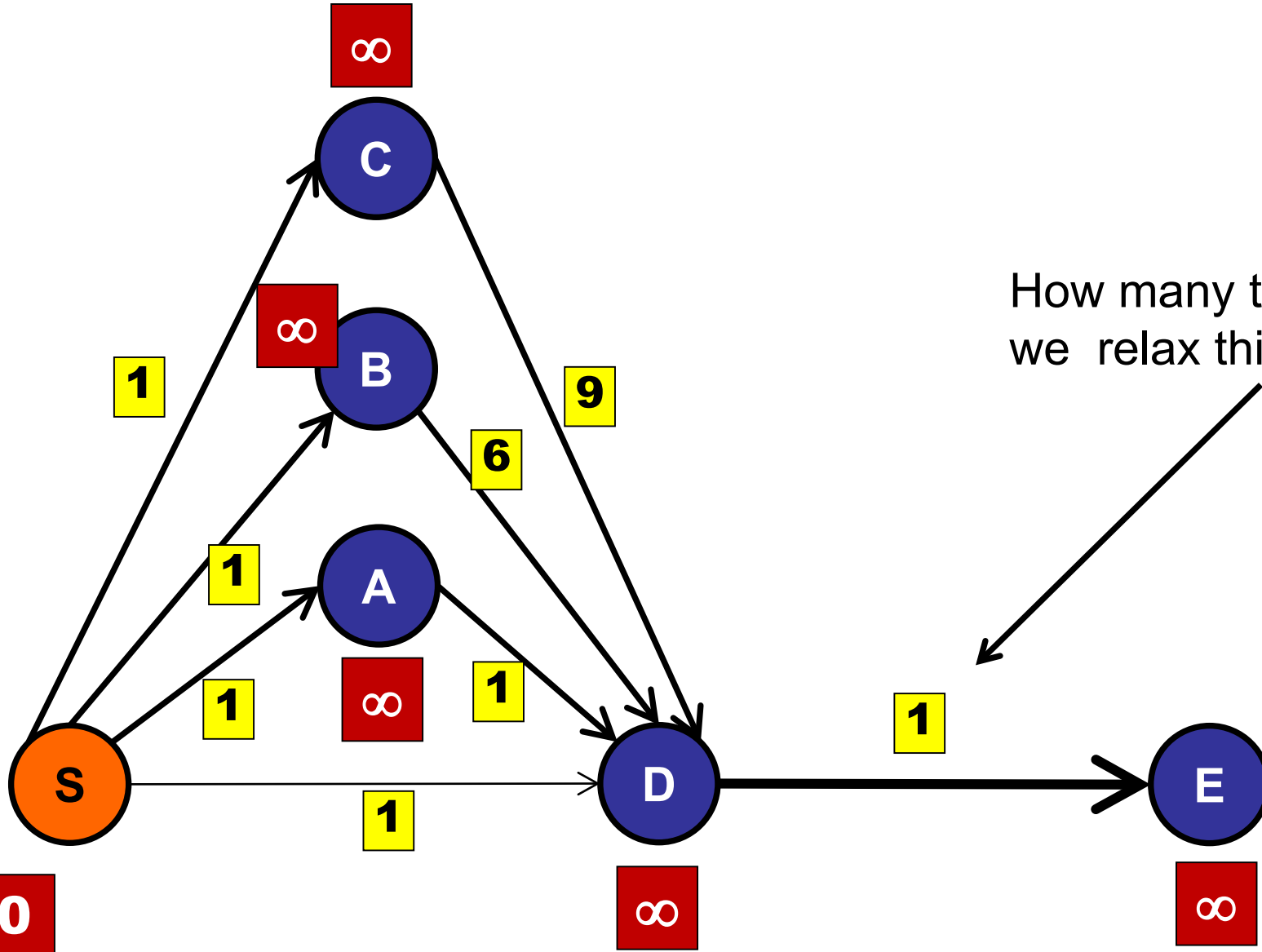
---

for (every edge  $e$  in a graph)

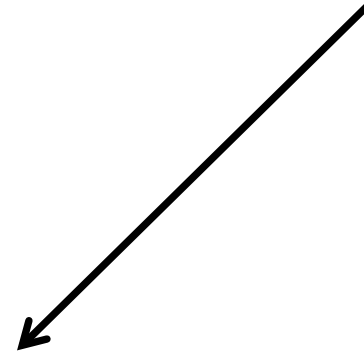
$\text{relax}(e)$



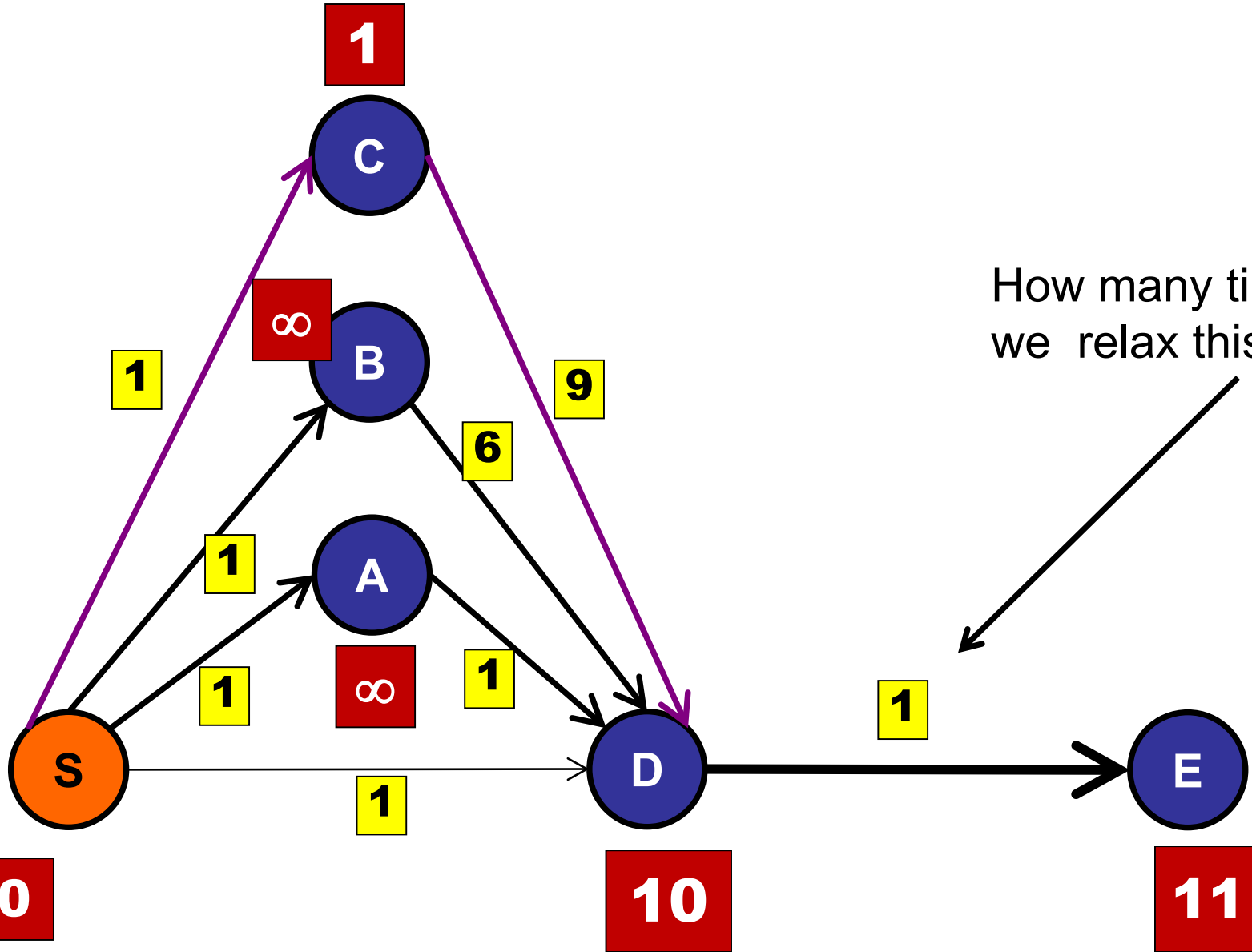
# Shortest Paths



How many times might we relax this edge?



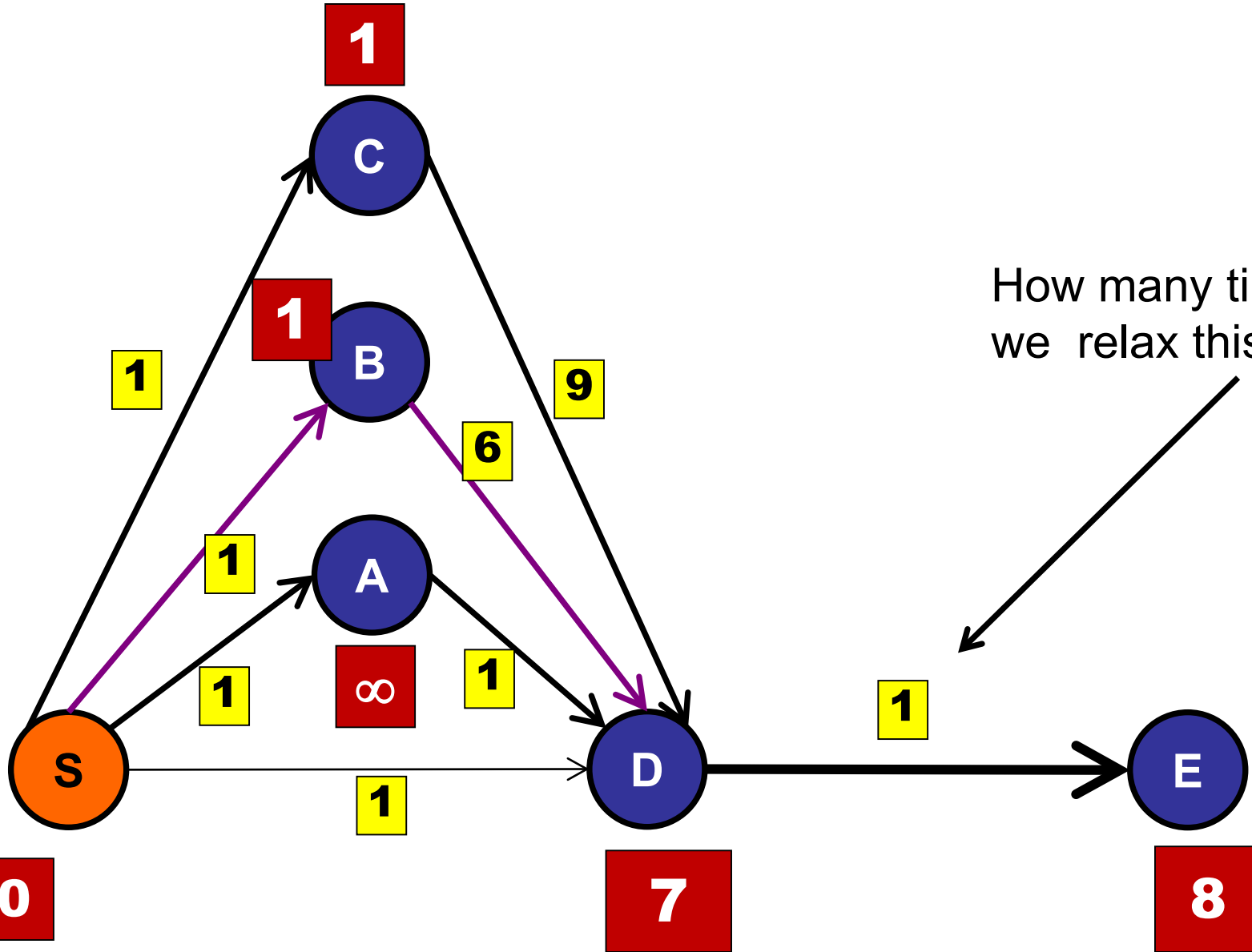
# Shortest Paths



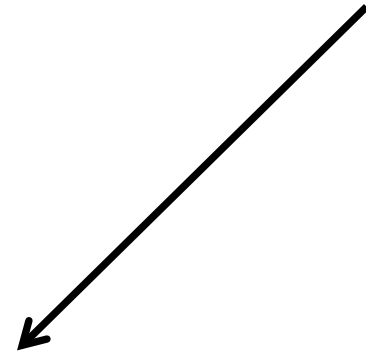
How many times might we relax this edge?



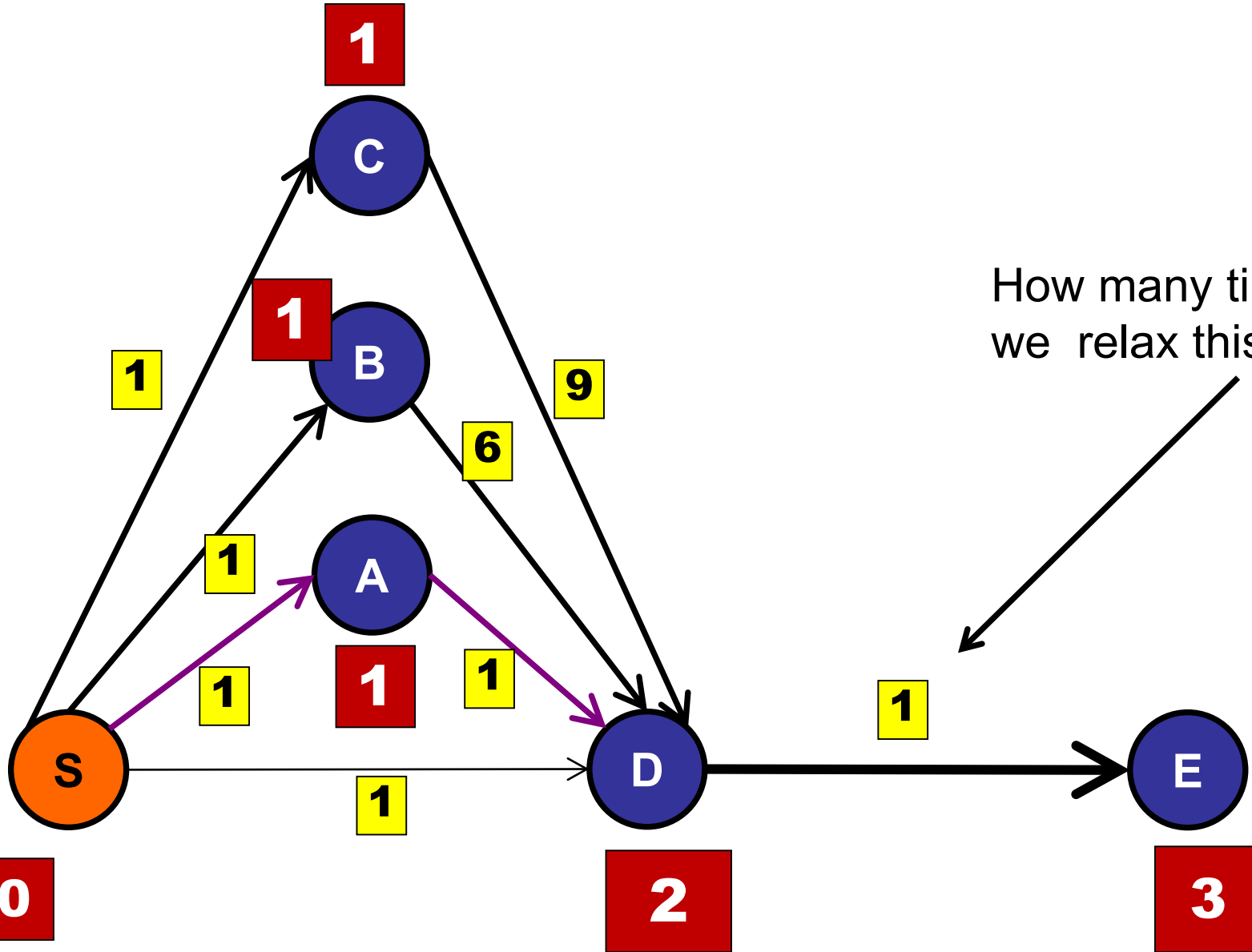
# Shortest Paths



How many times might we relax this edge?

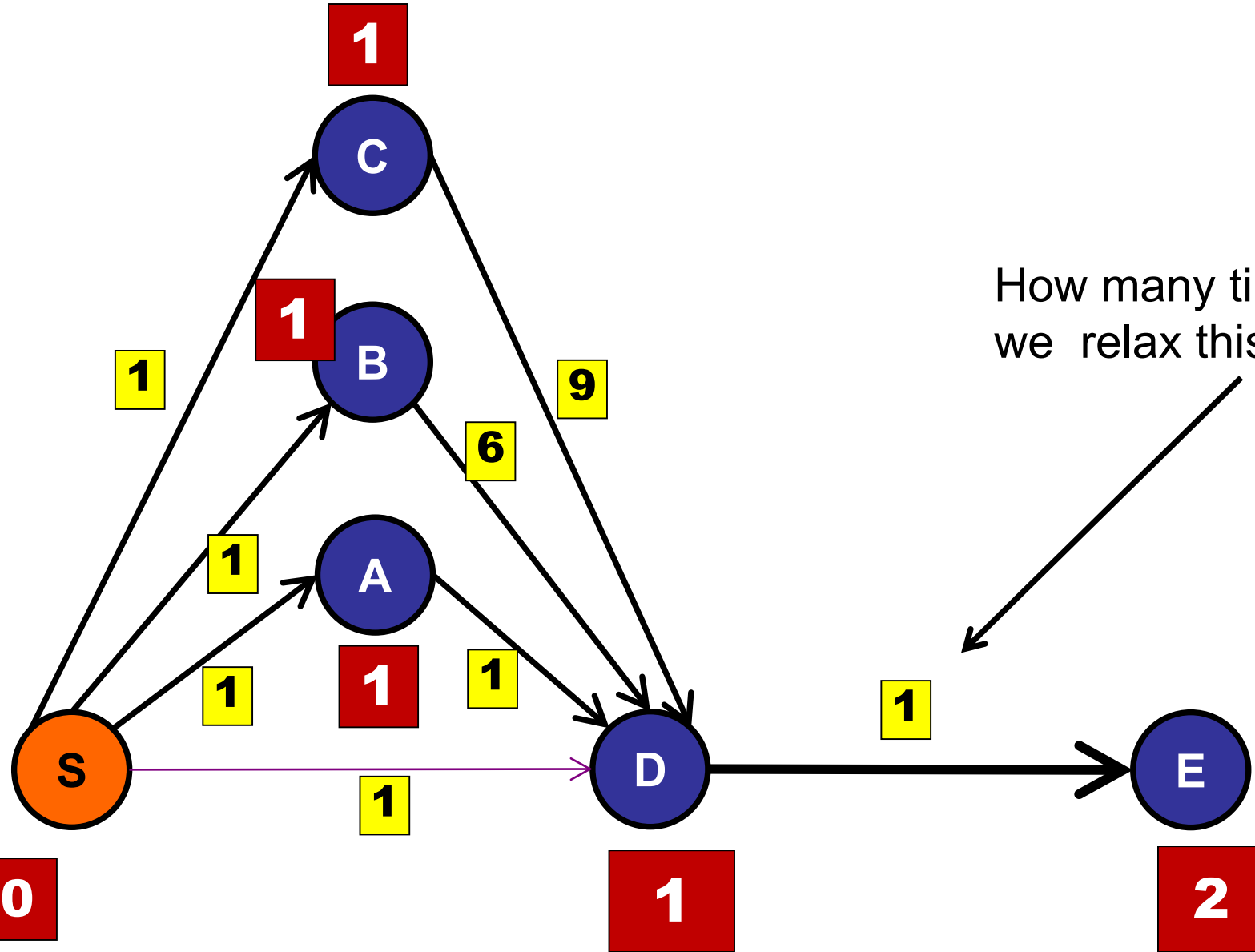


# Shortest Paths



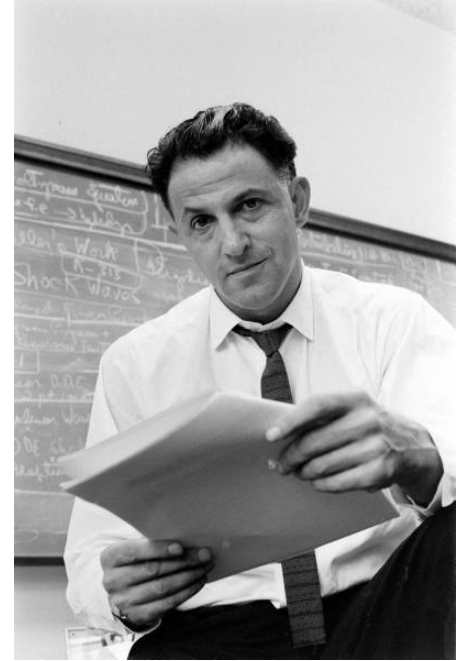
How many times might we relax this edge?

# Shortest Paths

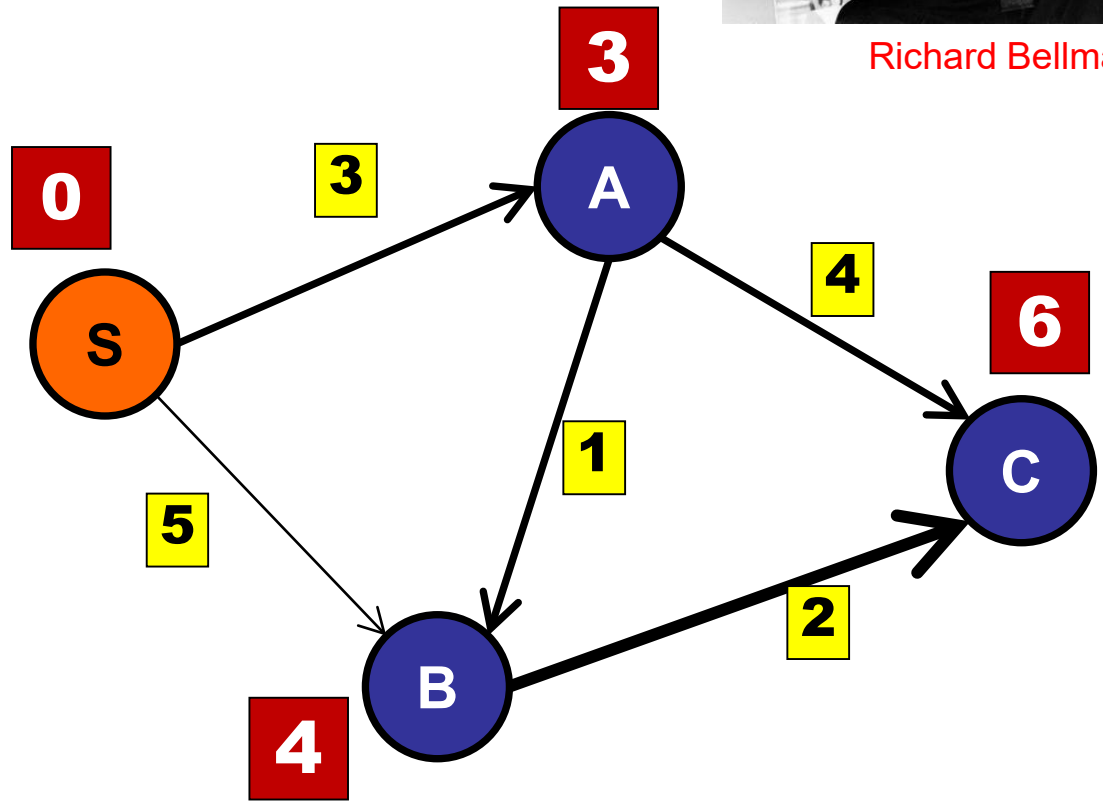


# Bellman-Ford

```
for (i=0; i<numNode; i++)  
    for (every edge e in graph)  
        relax(e)
```

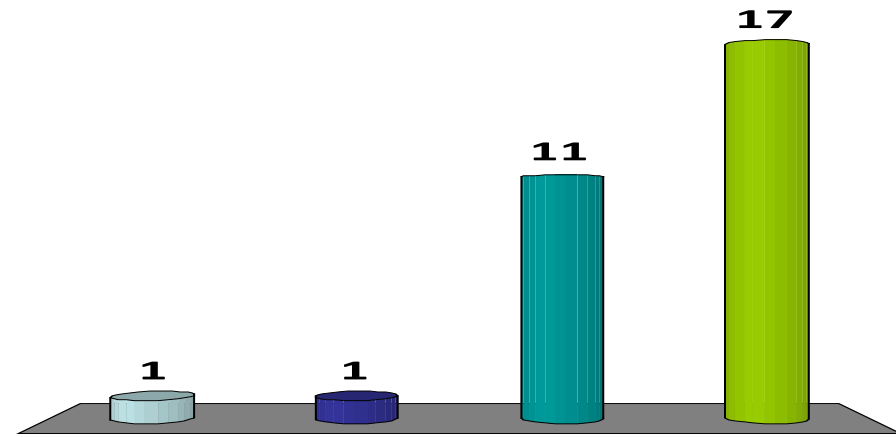


Richard Bellman



# When can you terminate early?

1. When a relax operation has no effect.
2. When two consecutive relax operations have no effect.
- ✓ 3. When an entire sequence of  $|E|$  relax operations have no effect.
4. Never. Only after  $|V|$  complete iterations.



# Bellman-Ford

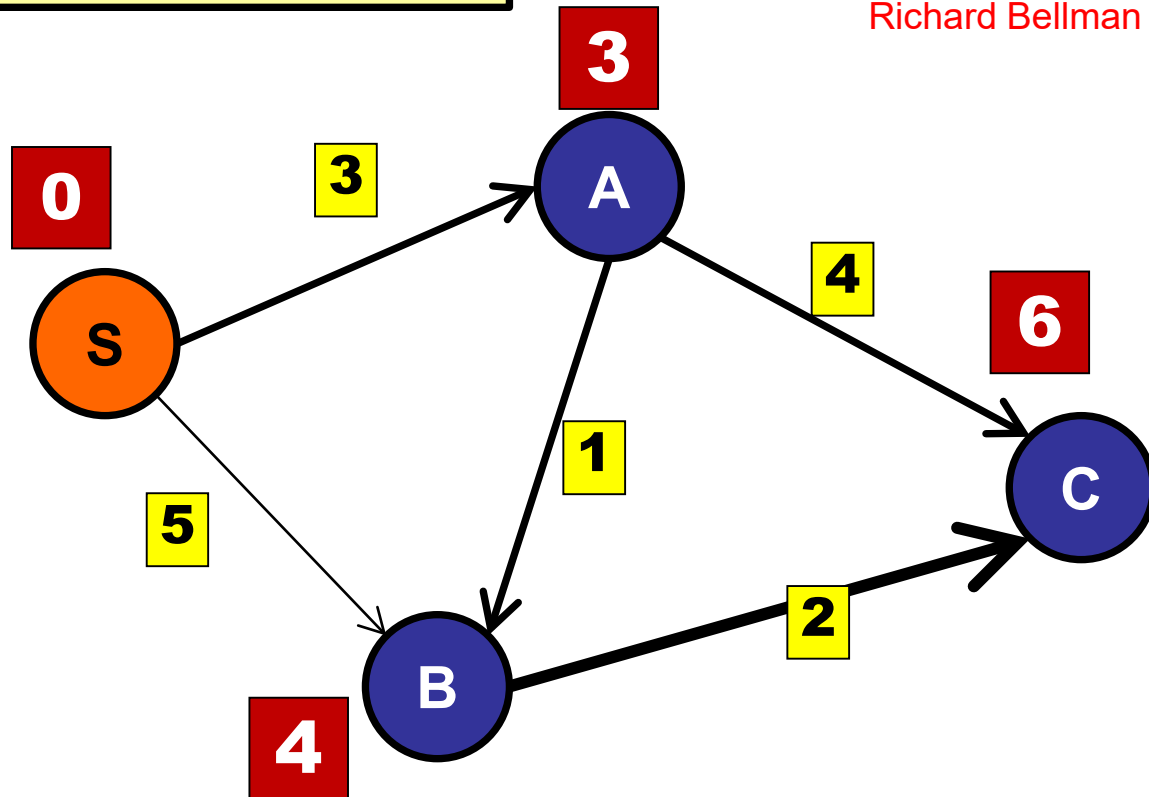
```
for (i=0; i<numNode; i++)
```

```
    for (every edge e in graph)  
        relax(e)
```



Richard Bellman

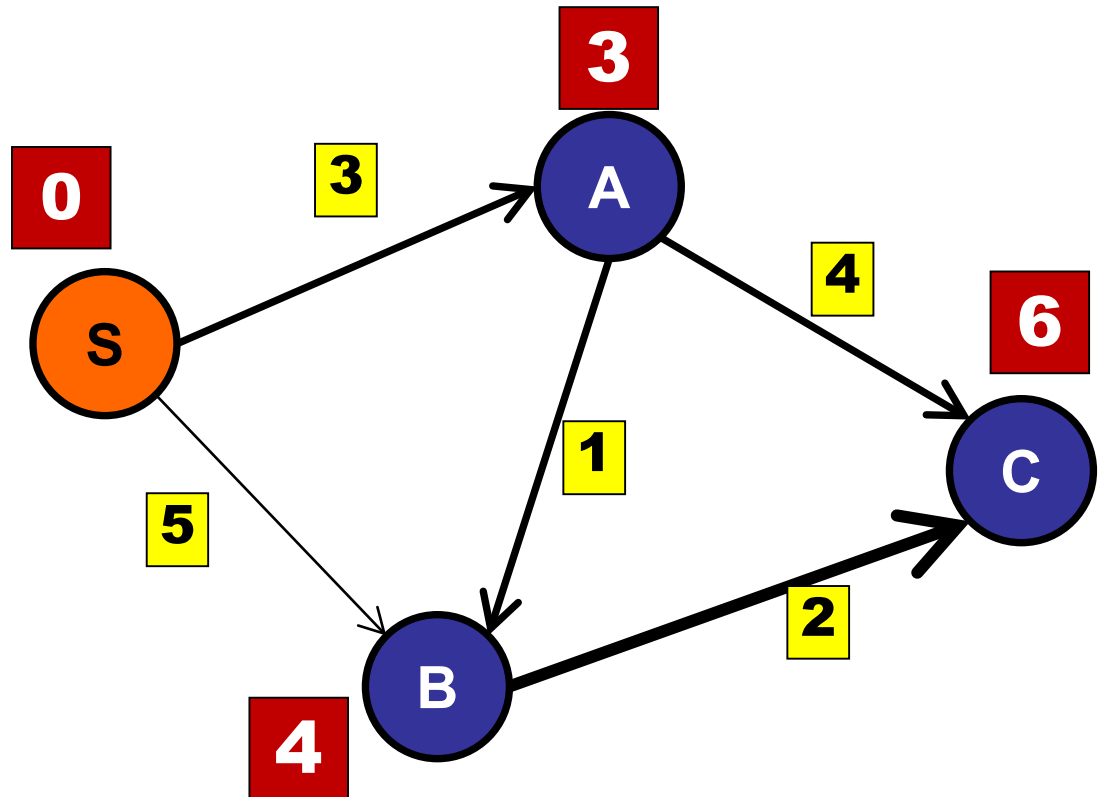
Terminate early if no  
more improvement



# Bellman-Ford

---

```
for (i=0; i<numNode; i++)  
    for (every edge e in graph)  
        relax(e)
```



# Bellman-Ford

---

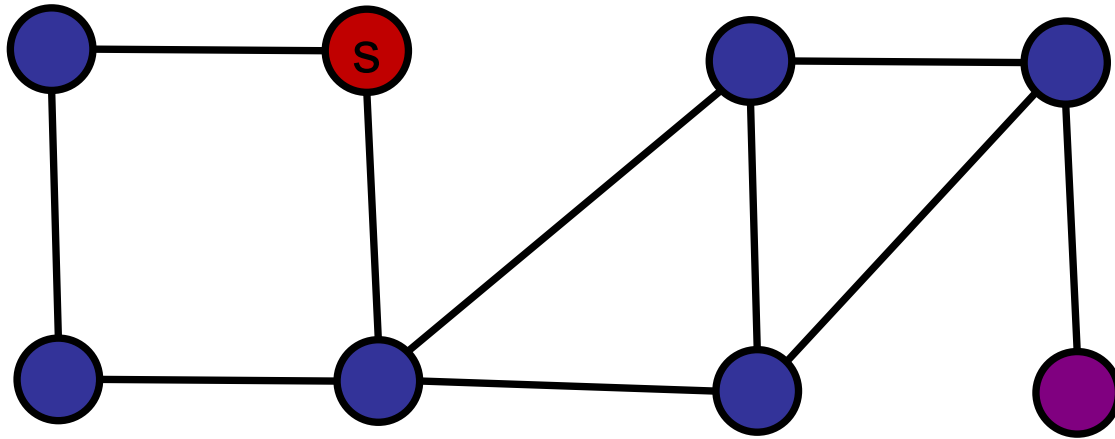
Why does this work?



# Bellman-Ford

---

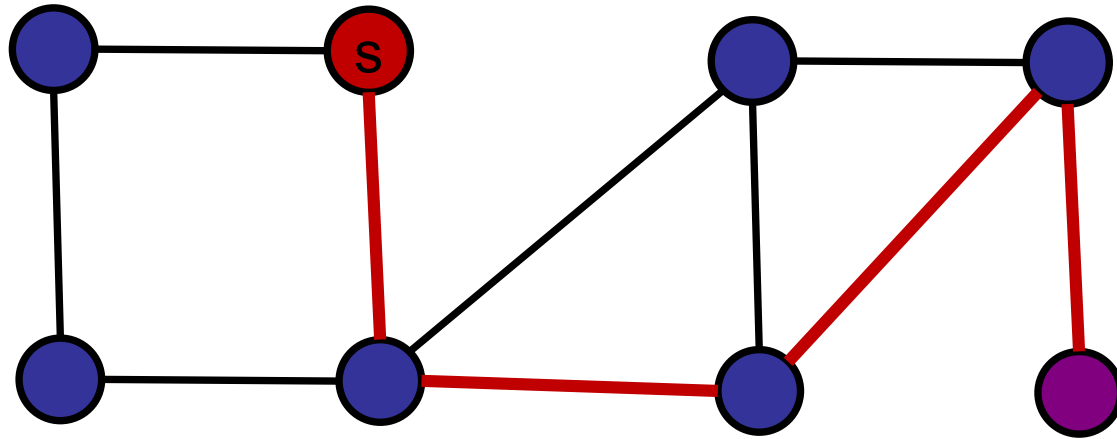
Why does this work?



# Bellman-Ford

---

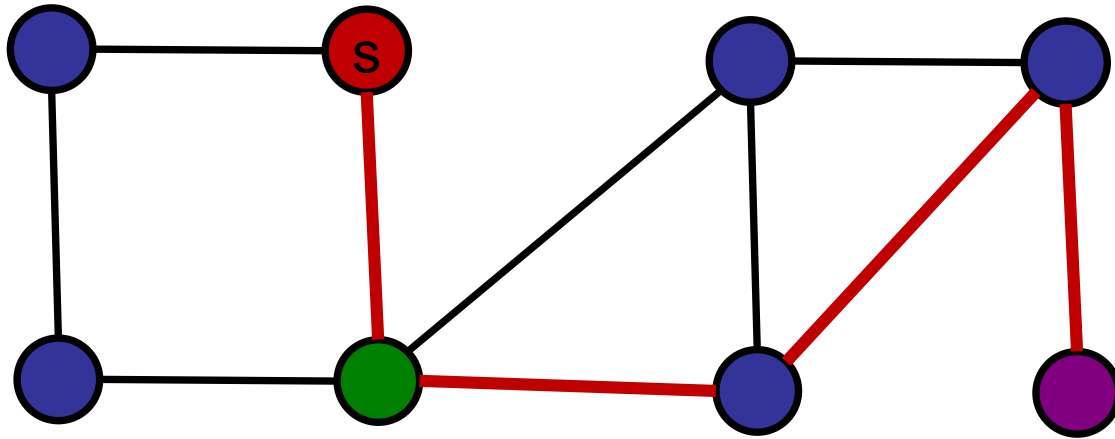
Why does this work?



Look at minimum weight path from S to D.  
(Path is simple: no loops.)

# Bellman-Ford

# Why does this work?

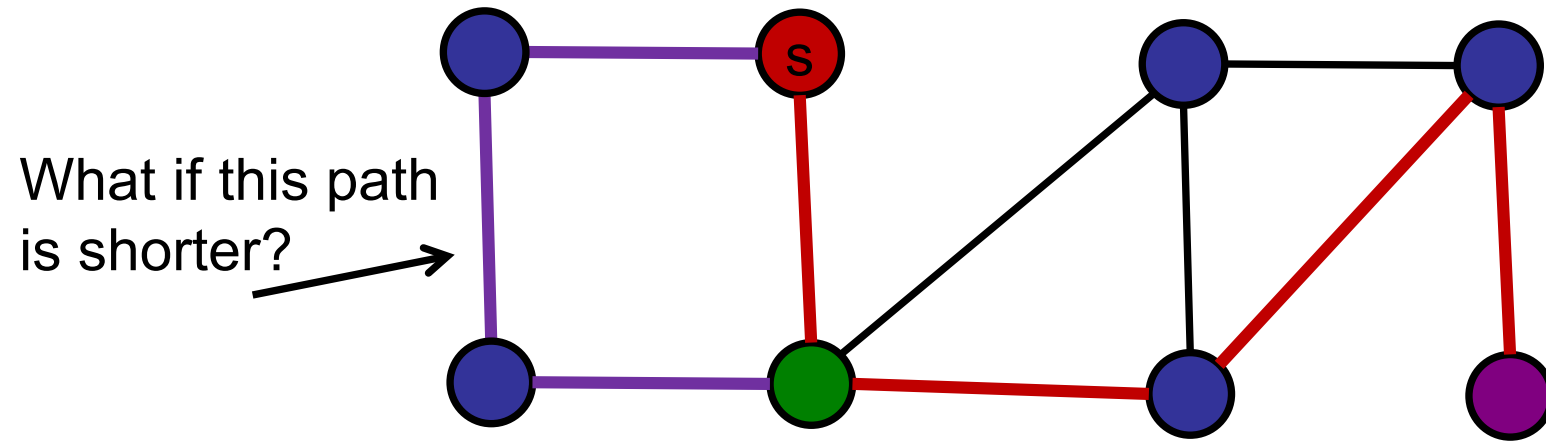


After 1 iteration, 1 hop estimate is correct.

# Bellman-Ford

---

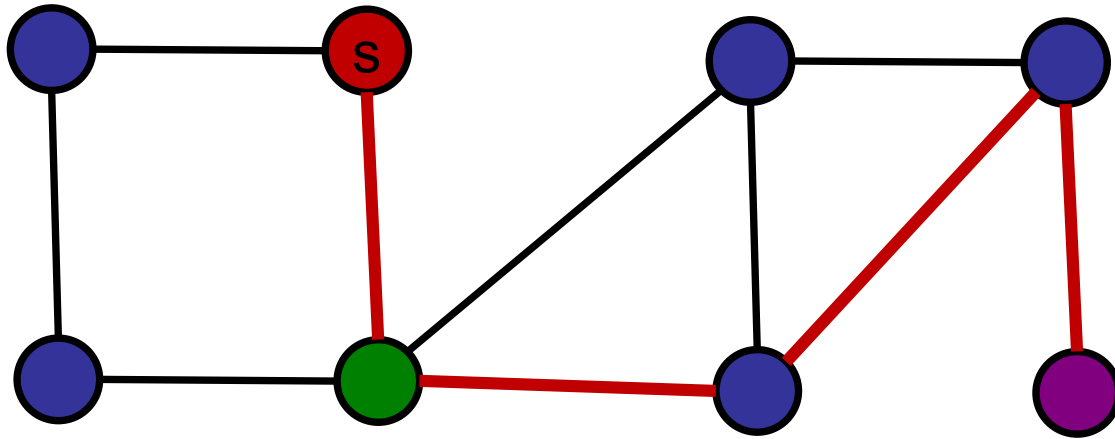
Why does this work?



After 1 iteration, 1 hop estimate is correct.

# Bellman-Ford

# Why does this work?

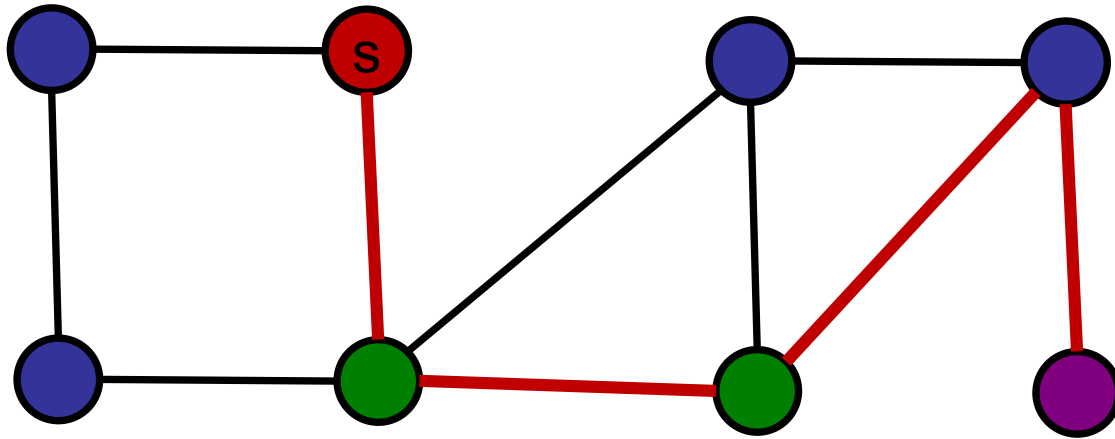


After 1 iteration, 1 hop estimate is correct.

# Bellman-Ford

---

Why does this work?

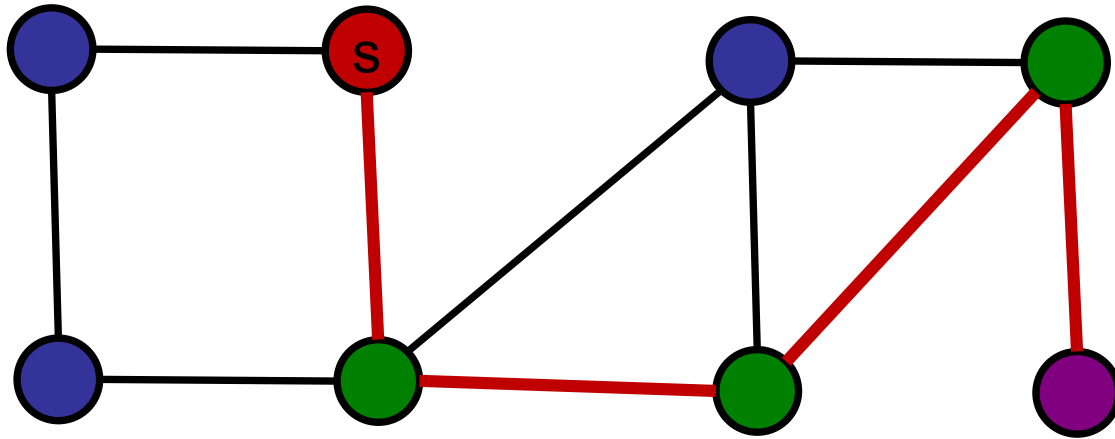


After 2 iterations, 2 hop estimate is correct.

# Bellman-Ford

---

Why does this work?

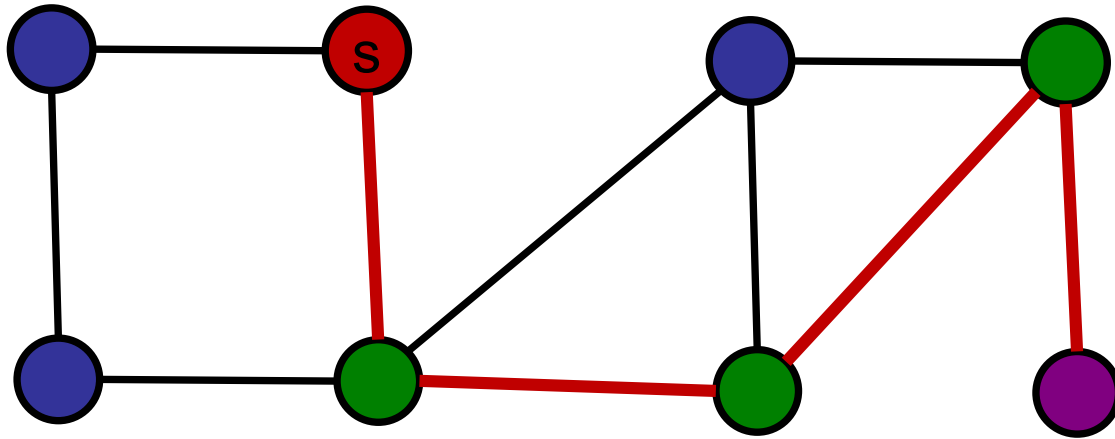


After 3 iterations, 3 hop estimate is correct.

# Bellman-Ford

---

Why does this work?



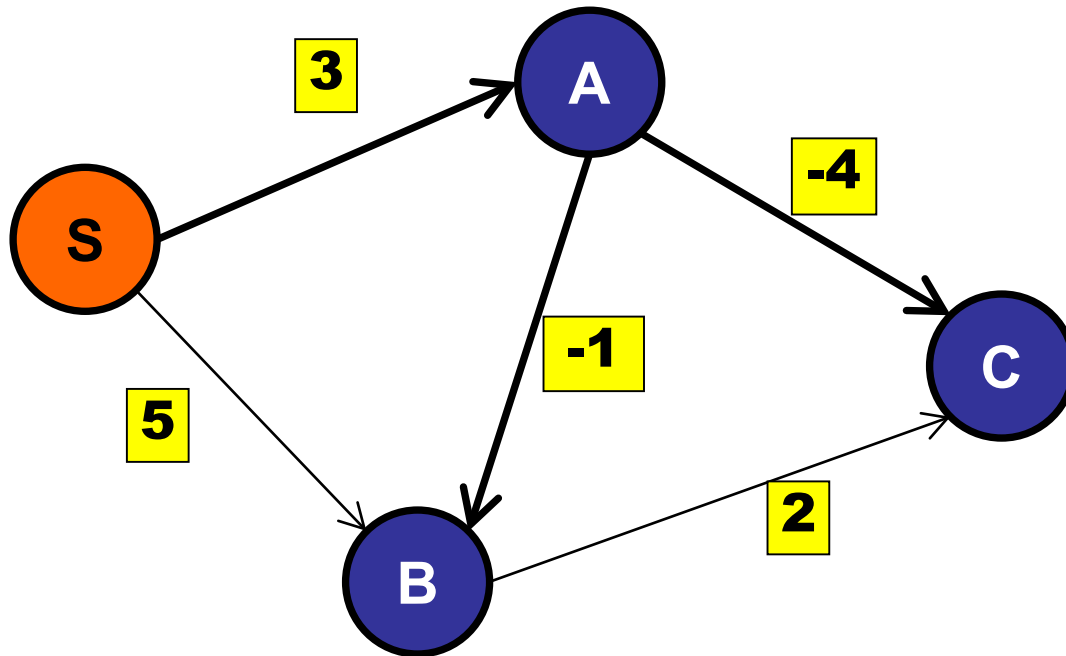
After 4 iterations, D estimate is correct.



# Bellman-Ford

---

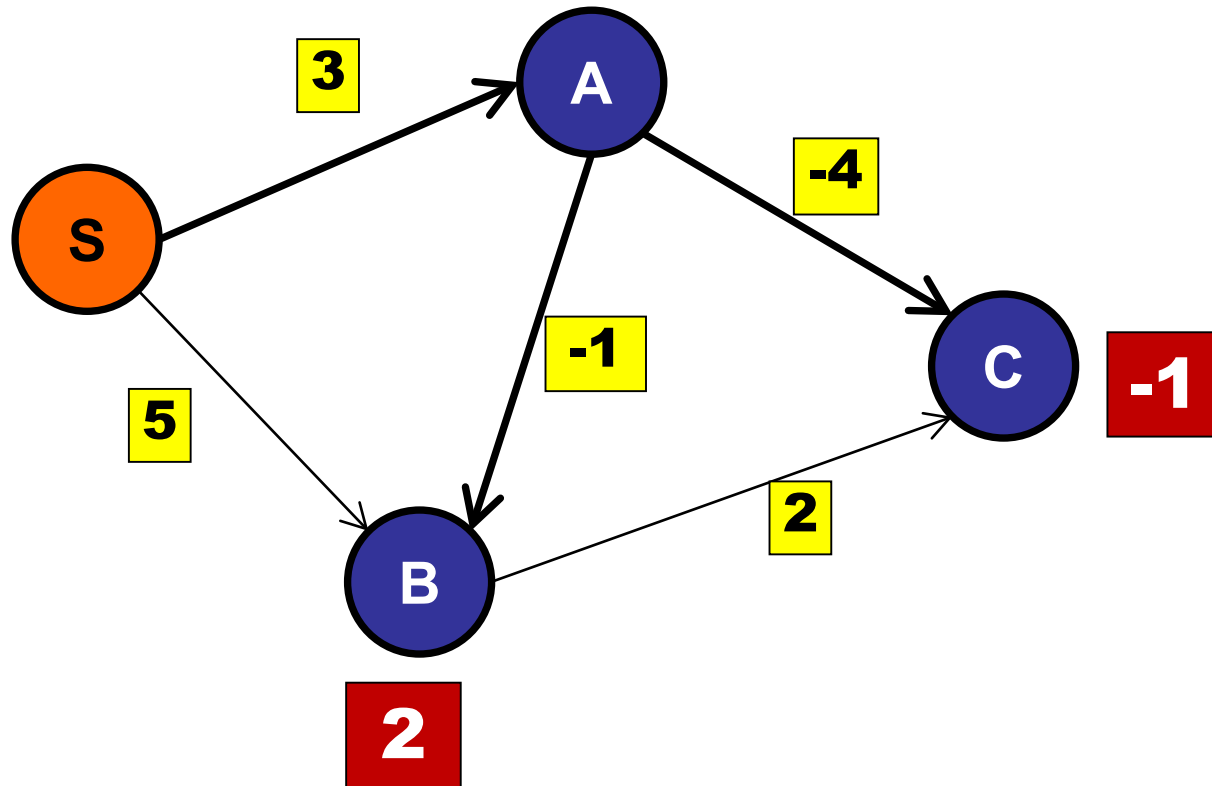
What if edges have negative weight?



# Bellman-Ford

---

What if edges have negative weight?

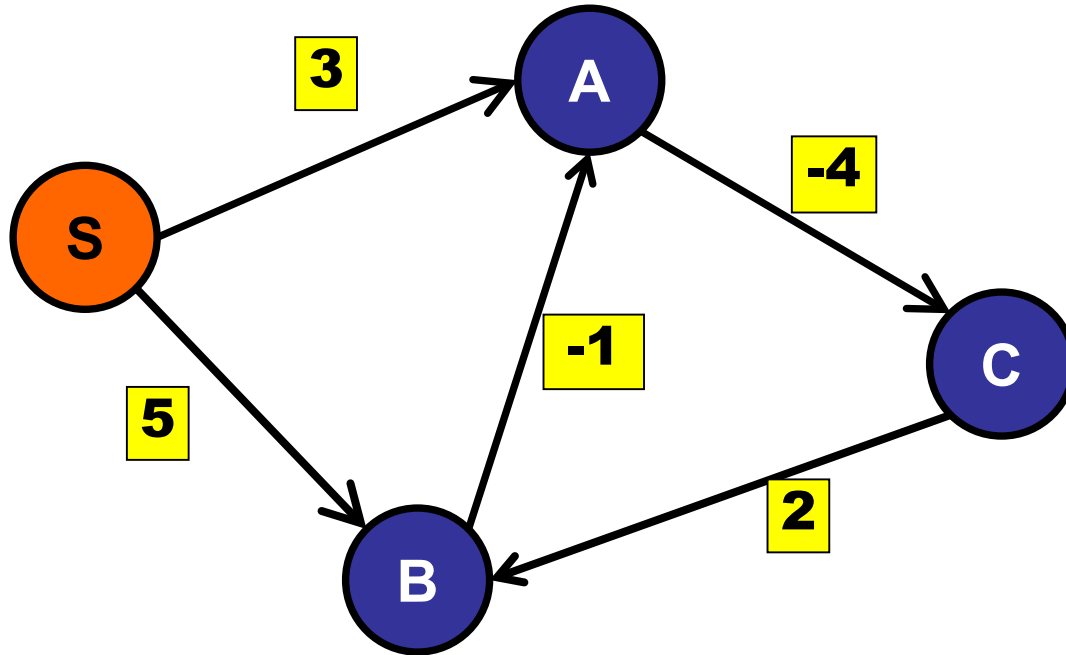


No problem!

# Bellman-Ford

---

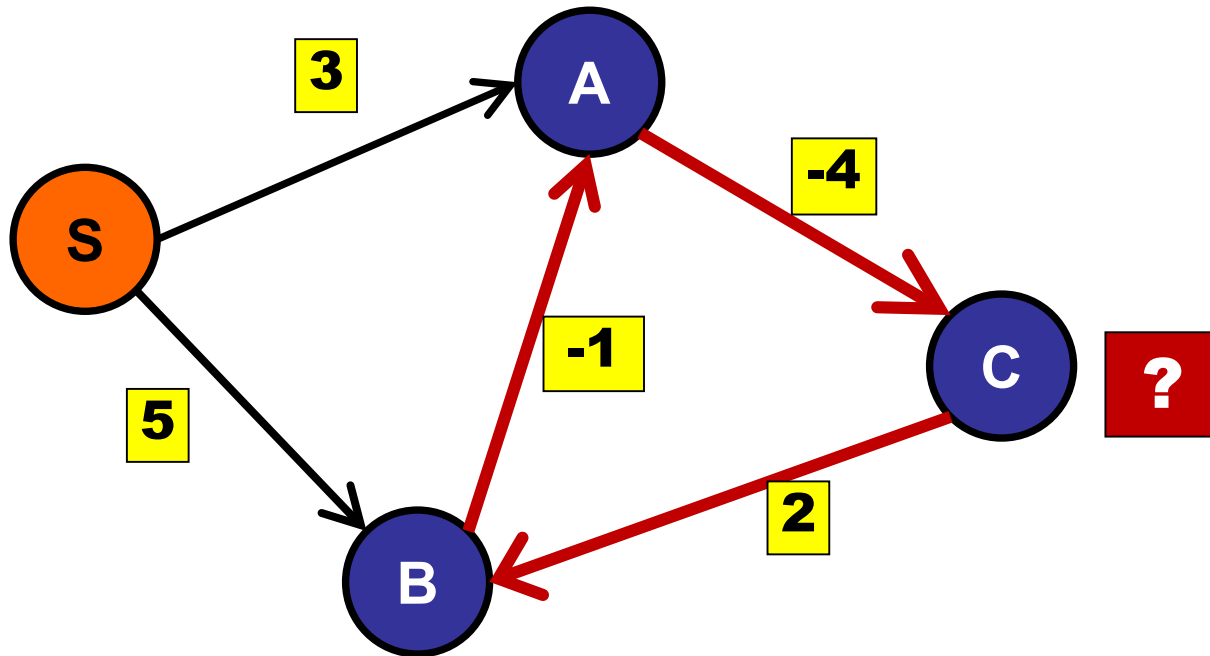
What if edges have negative weight?



# Bellman-Ford

---

What if edges have negative weight?

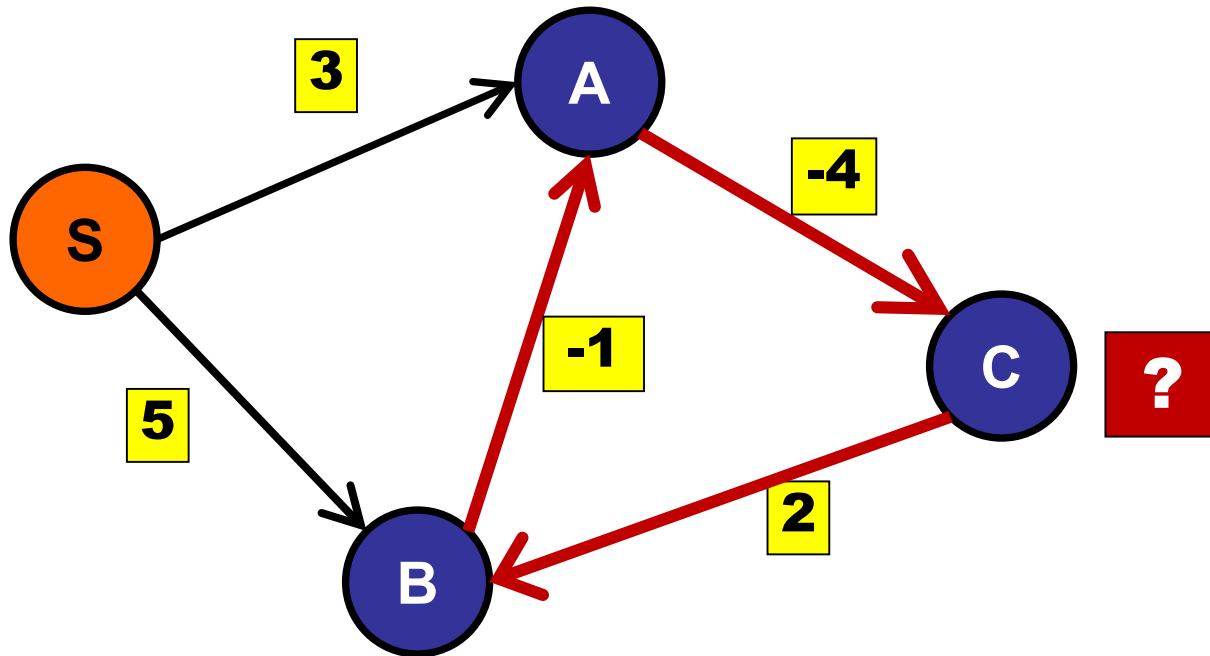


$d(S,C)$  is infinitely negative!

# Negative weight cycles

---

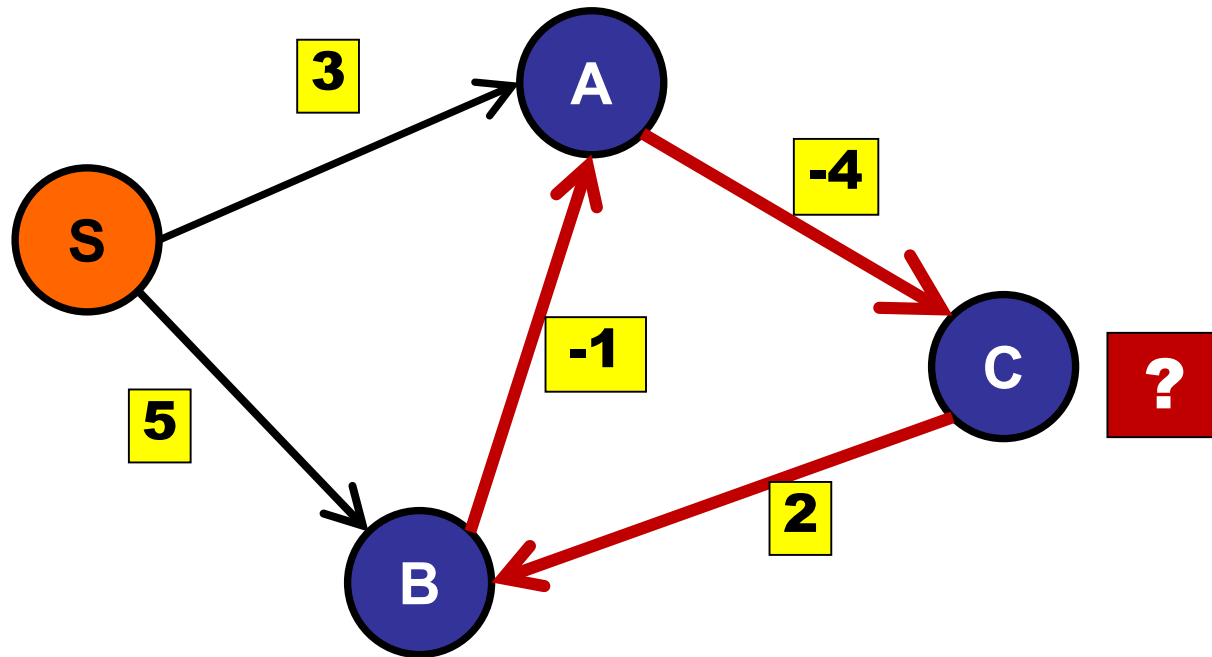
How to detect negative weight cycles?



# Negative weight cycles

---

How to detect negative weight cycles?



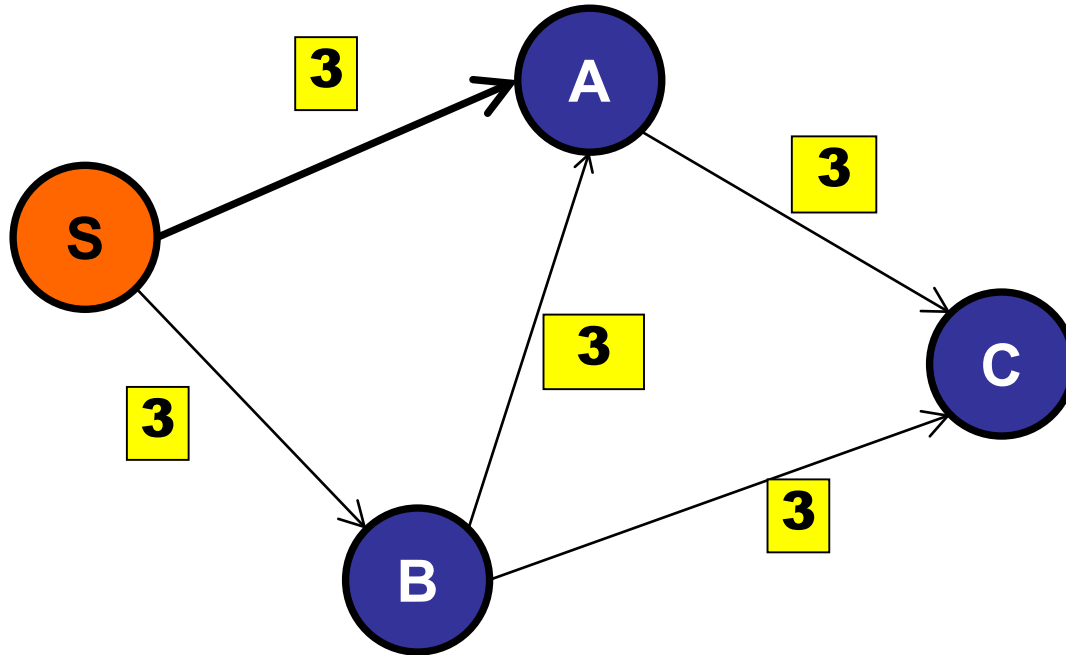
Run Bellman-Ford for  $|V|+1$  iterations.

If an estimate changes in the last iteration...  
then negative weight cycle.

# Bellman-Ford

---

Special case: all edges have the same weight.



Use regular Breadth-First Search.

# Bellman-Ford Summary

---

## Basic idea:

- Repeat  $|V|$  times: relax every edge
- Stop when “converges”.
- $O(VE)$  time.

## Special issues:

- If negative weight-cycle: impossible.
- Use Bellman-Ford to detect negative weight cycle.
- If all weights are the same, use BFS.