Shortest-Path Algorithms II

Shortest Path with Negative Costs

Given a weighted directed graph (weights can be negative), a start node **s**, and end node **e**, find the cost of the shortest path from **s** to **e** (or report that it does not exist)

Using Dijkstra’s Algorithm

Does Dijkstra’s Algorithm work for this problem?

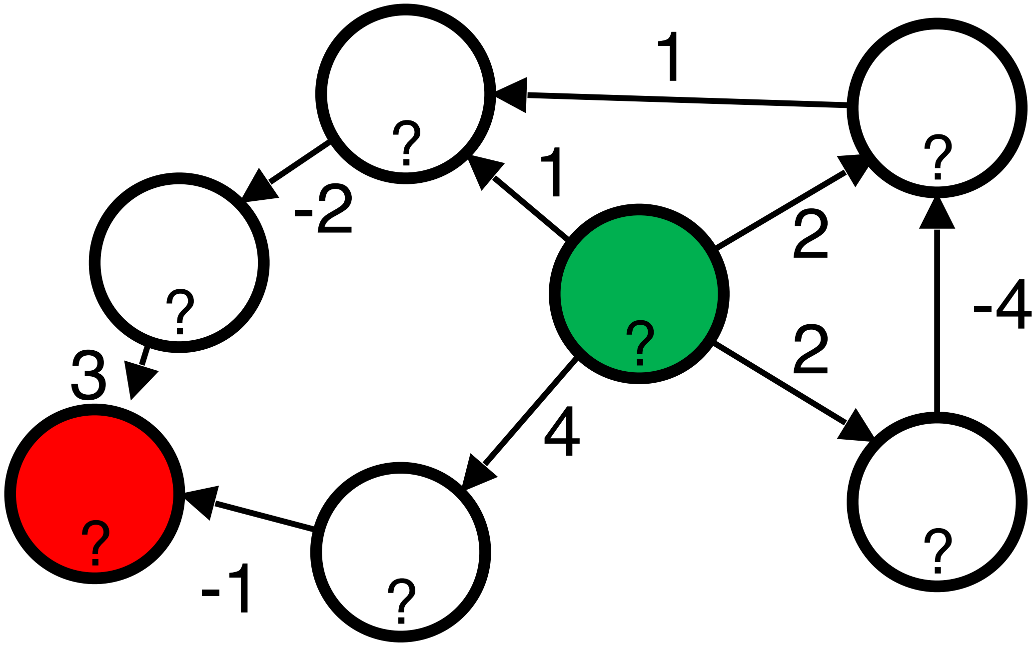
Using Dijkstra’s Algorithm

Does Dijkstra’s Algorithm work for this problem?

Breaks invariants

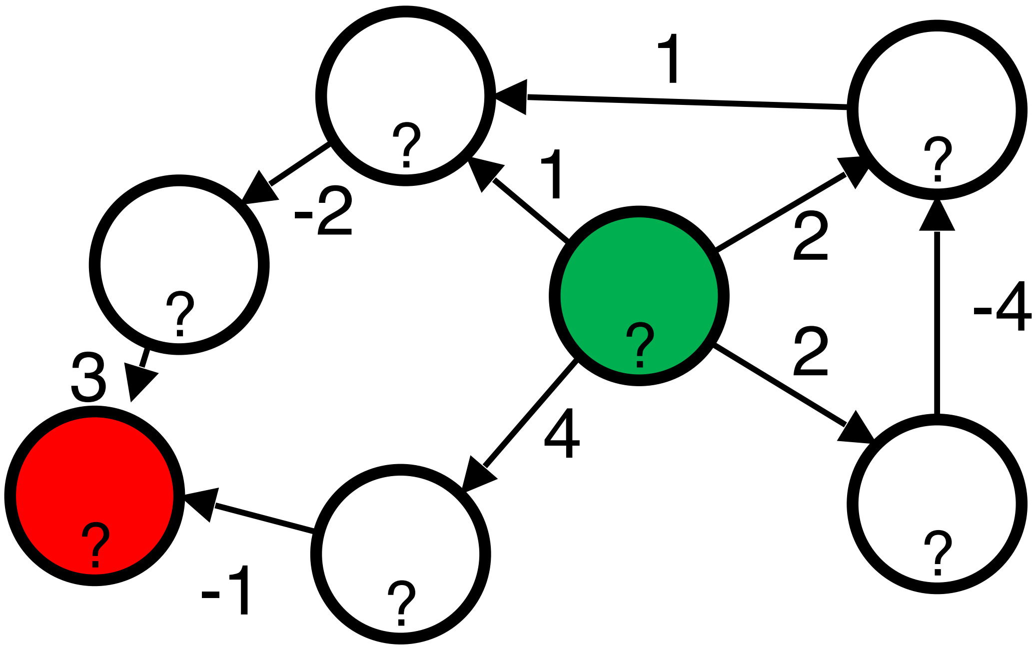
Negative Costs

start state
goal states



start state
goal states

Negative Costs



Is there anything we **know**?

Adjustment: Adding Constant

Proposed Idea: What if we add a fixed constant to each edge such that all negative edges become positive?

Adjustment: Adding Constant

Proposed Idea: What if we add a fixed constant to each edge such that all negative edges become positive?

Doesn’t work because this penalizes paths with more edges

Edge Relaxation

```
for each edge e:
  if(e.dest.cost > e.src.cost + e.cost)
    e.dest.cost = e.src.cost + e.cost;
```

Edge Relaxation

```
for each edge e:
  if(e.dest.cost > e.src.cost + e.cost)
    e.dest.cost = e.src.cost + e.cost;
```

Questions:

1. Will this process terminate?
2. How many iterations are needed?

Key Observations

If the shortest path to a node involves L edges, it will be found in L iterations.

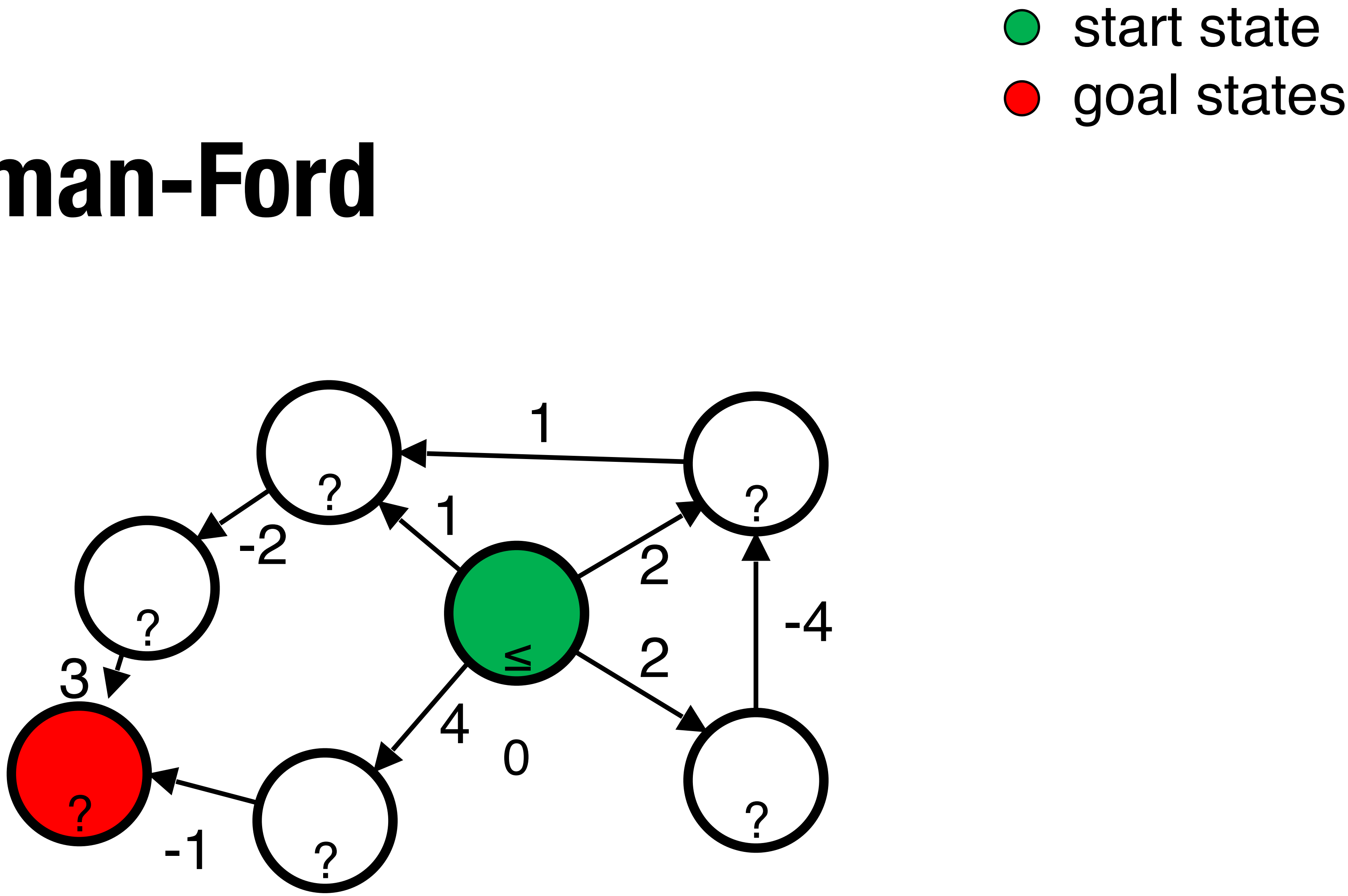
Key Observations

If the shortest path to a node involves L edges, it will be found in L iterations.

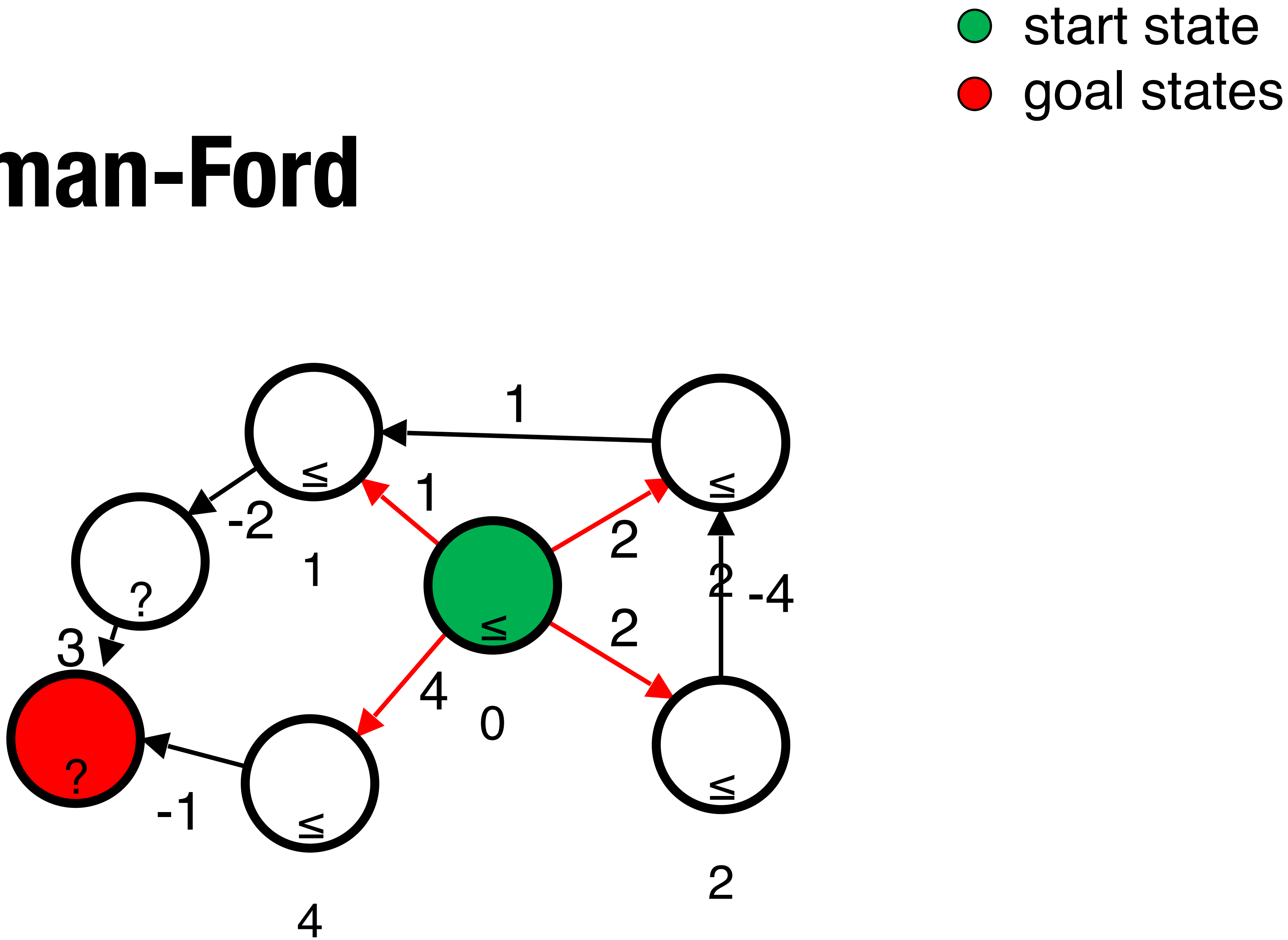
The shortest path to any node is either:

1) <= |V|
2) infinite

Bellman-Ford

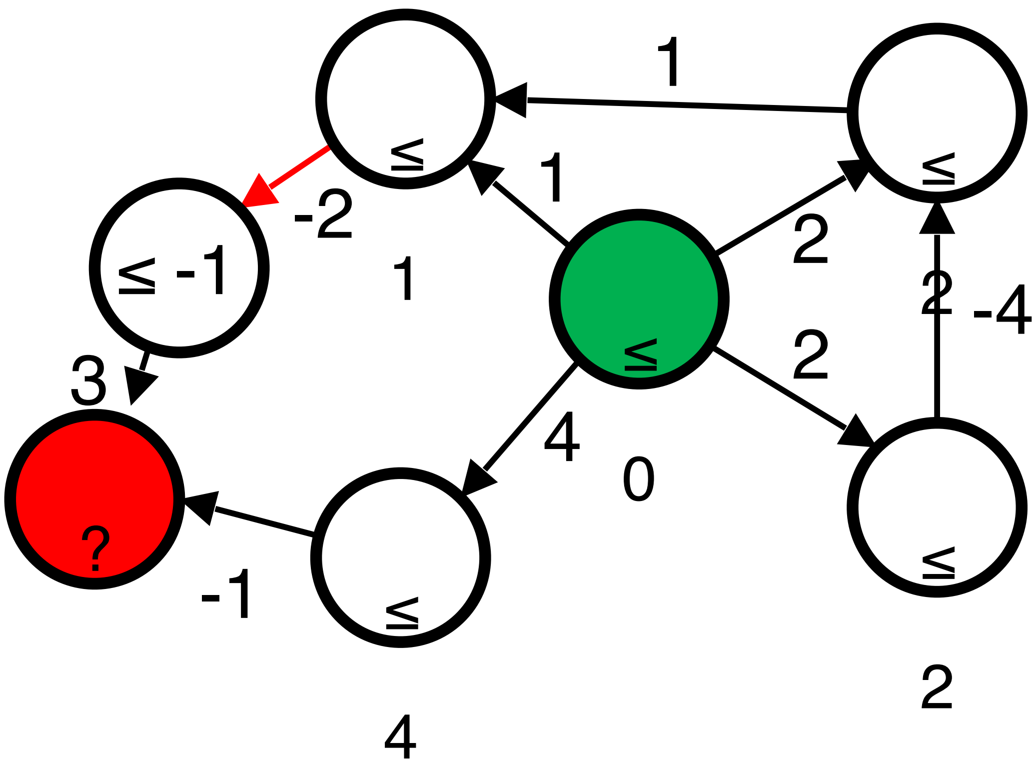


Bellman-Ford

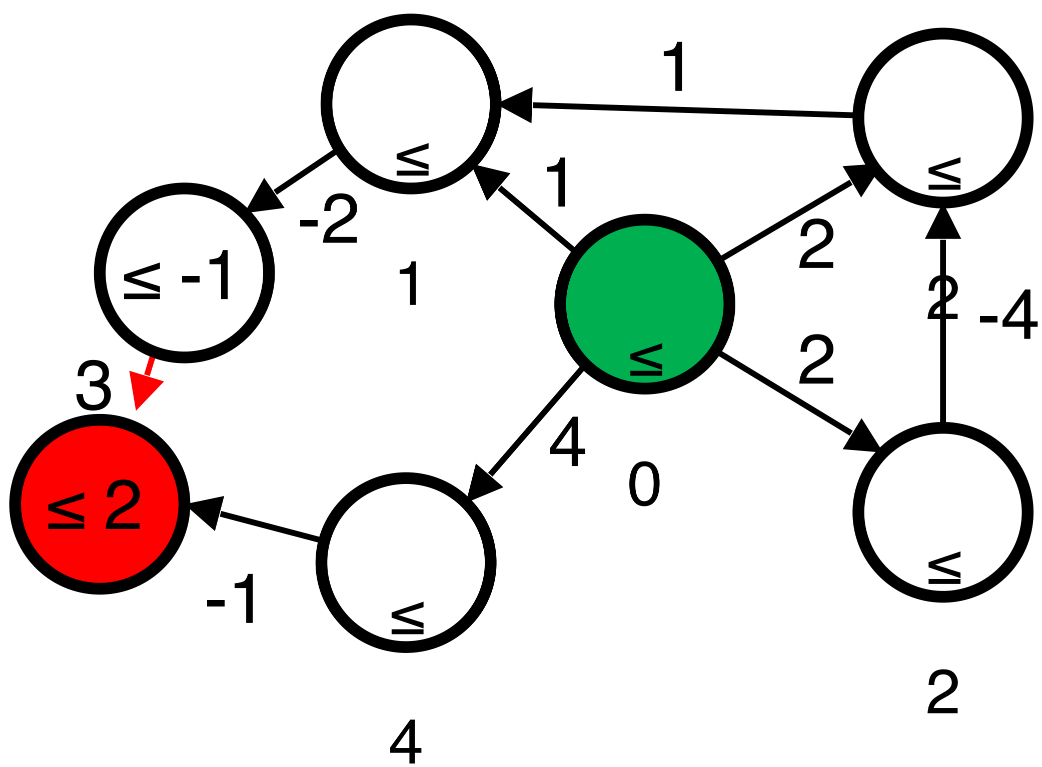


start state
goal states

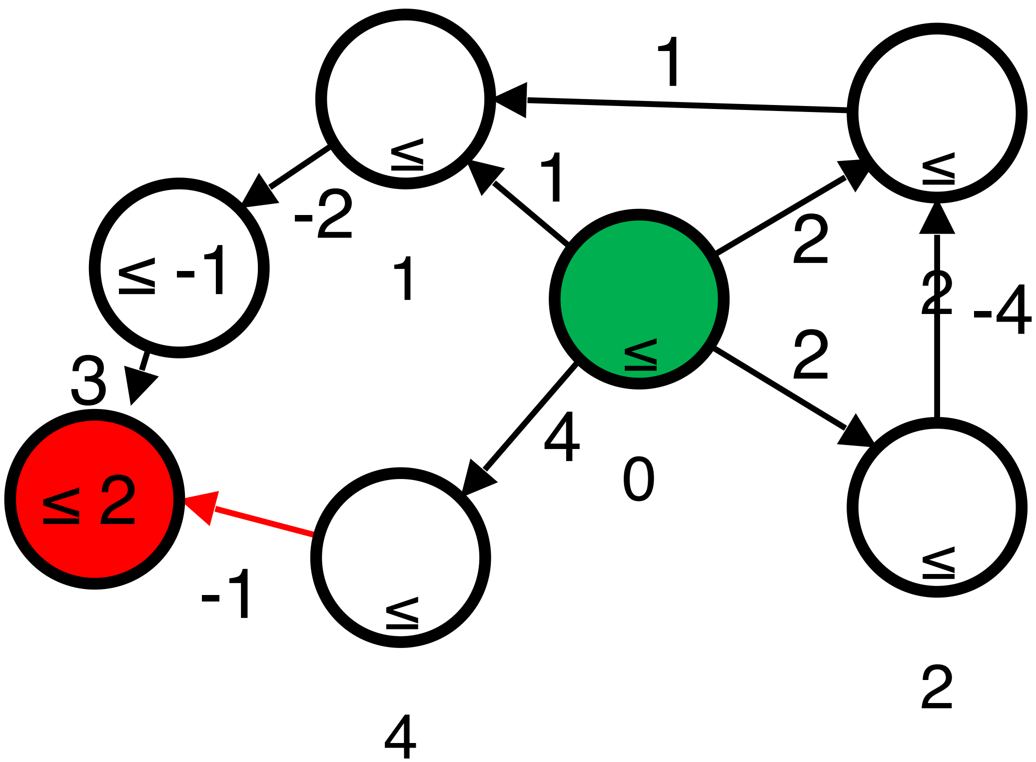
Bellman-Ford



Bellman-Ford

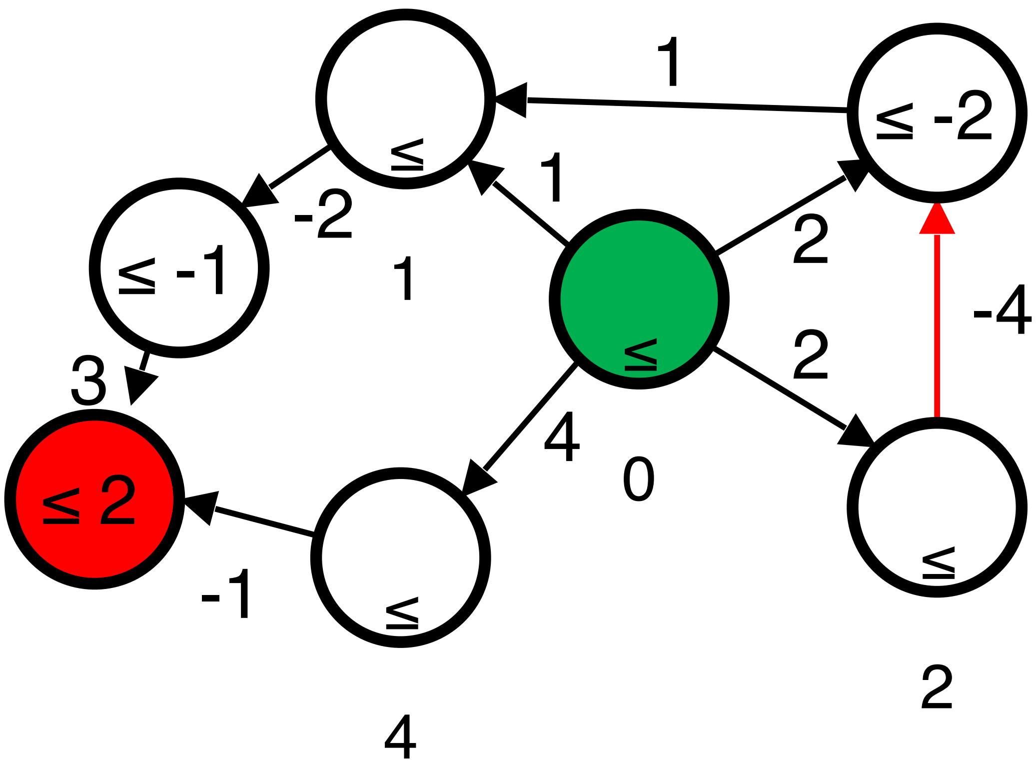


Bellman-Ford

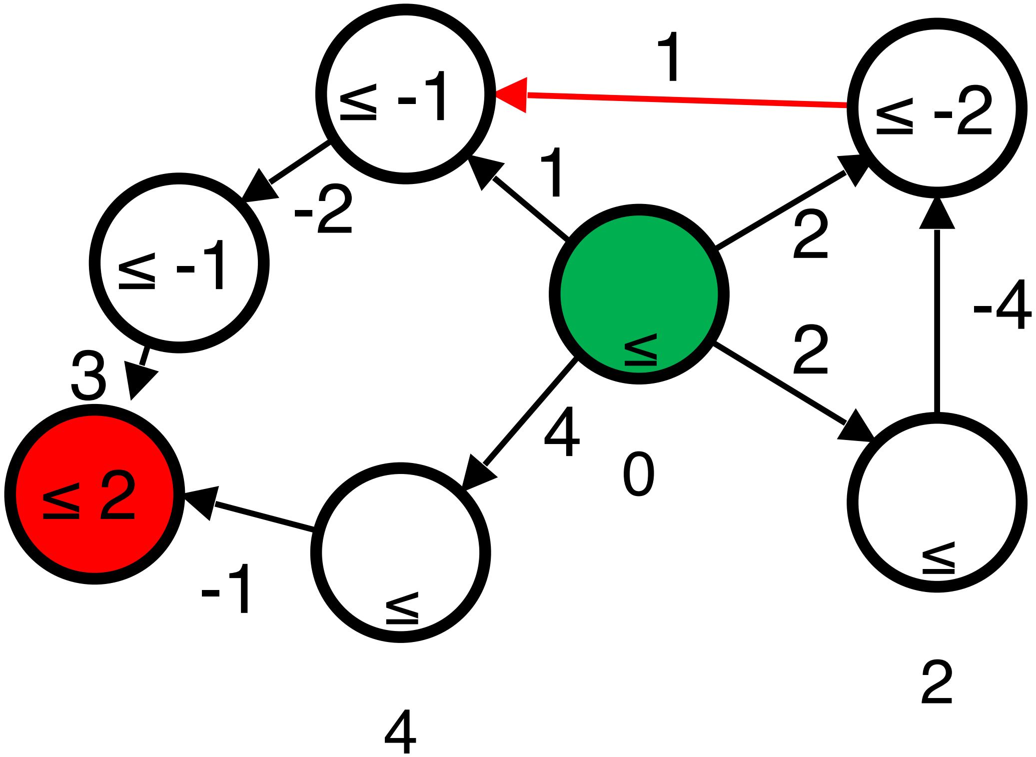


start state
goal states

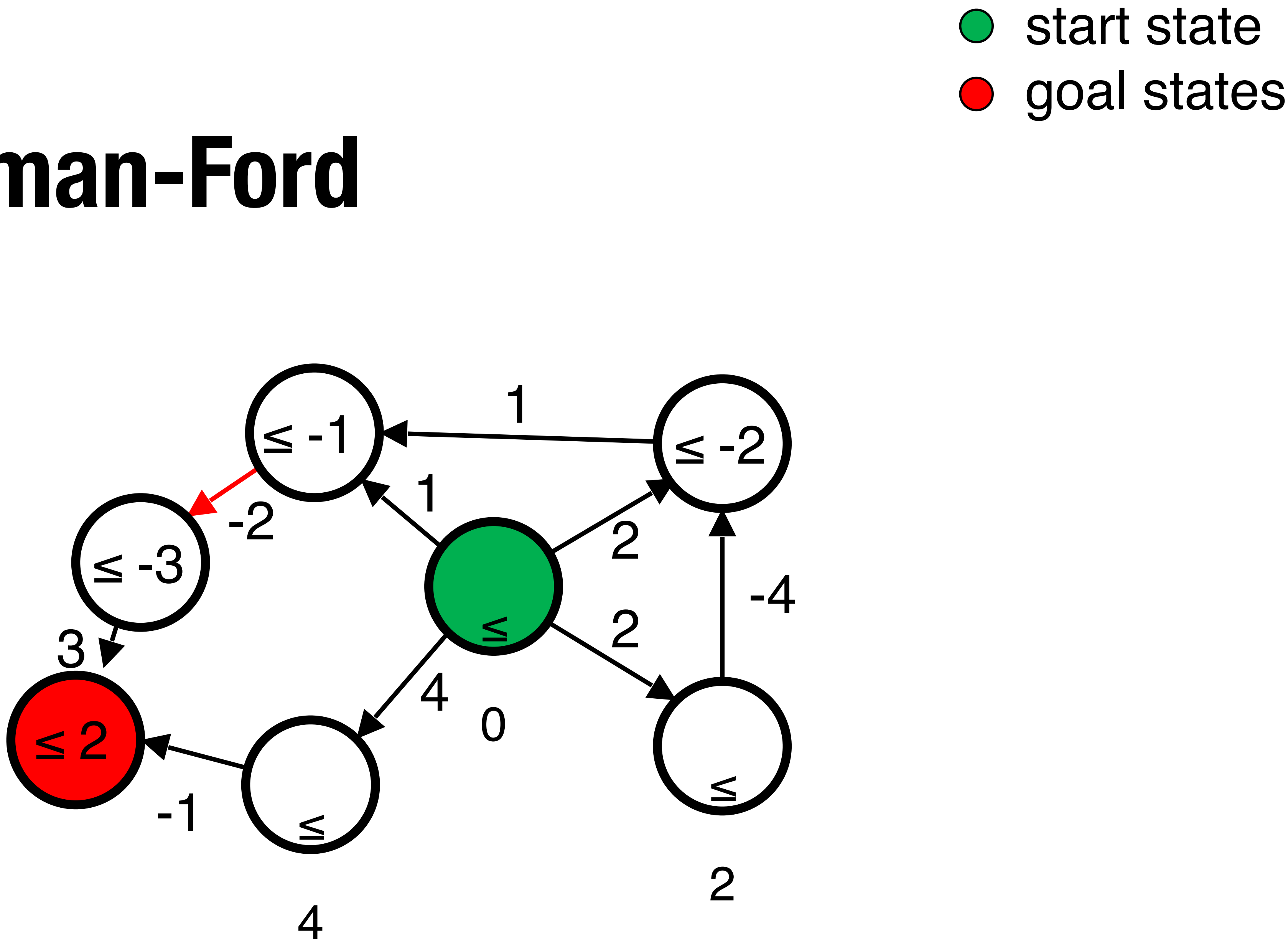
Bellman-Ford



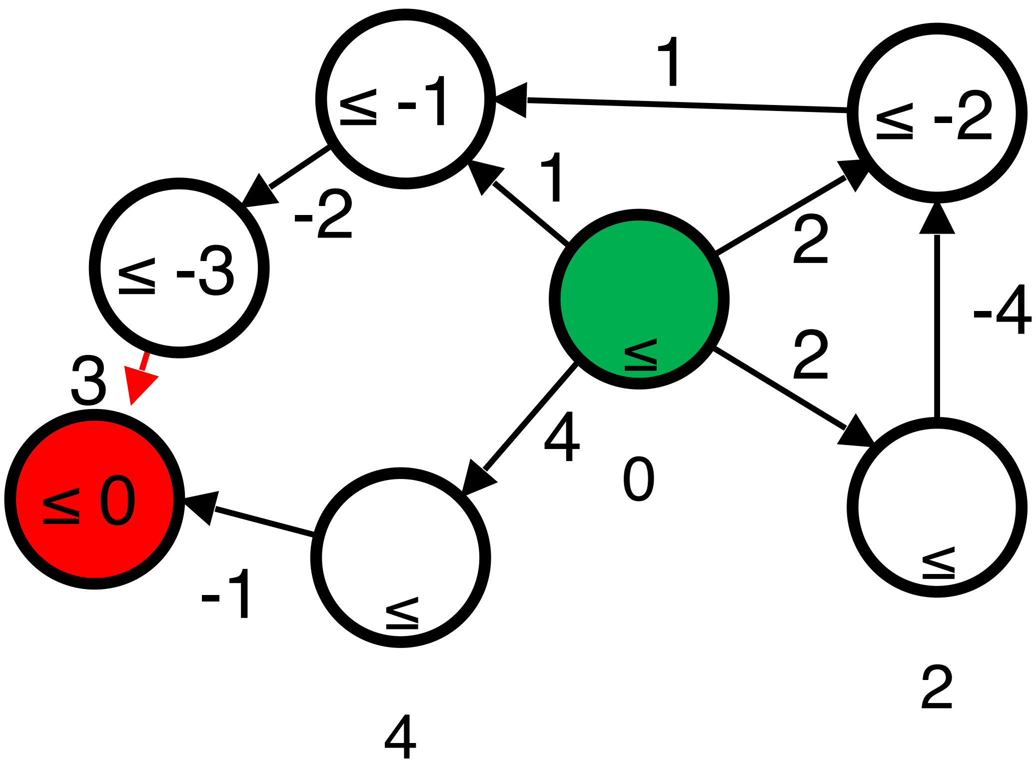
Bellman-Ford



Bellman-Ford

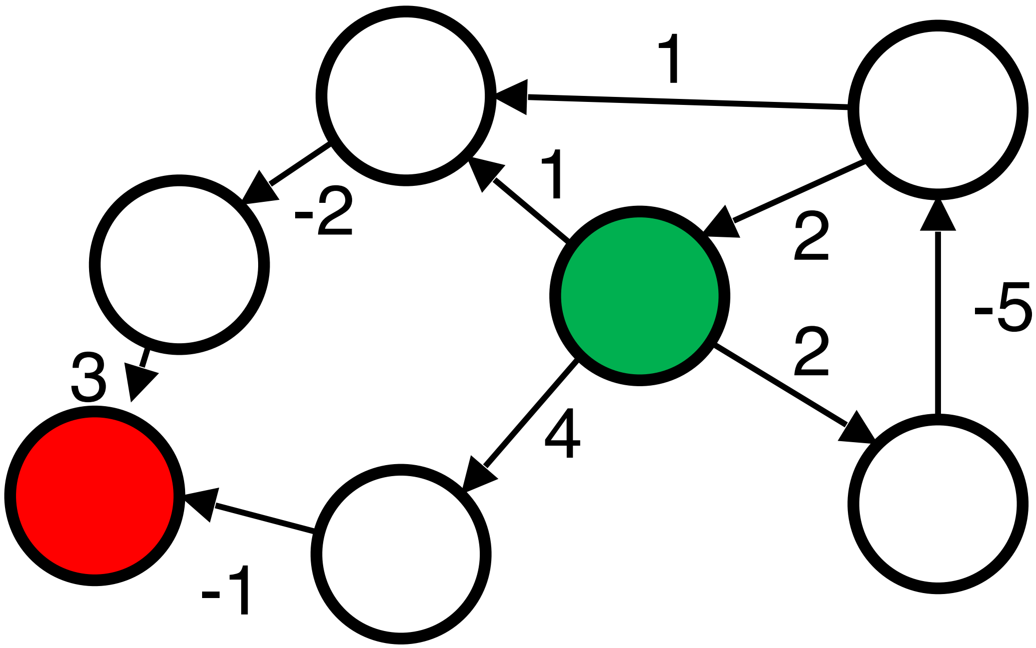


Bellman-Ford

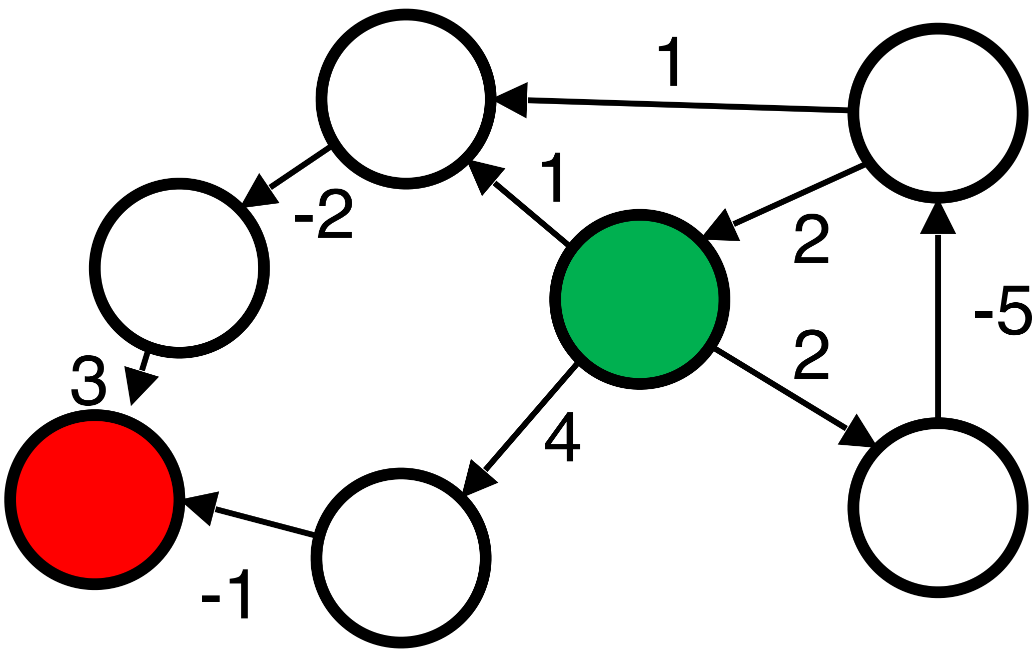


start state
goal states

Termination?

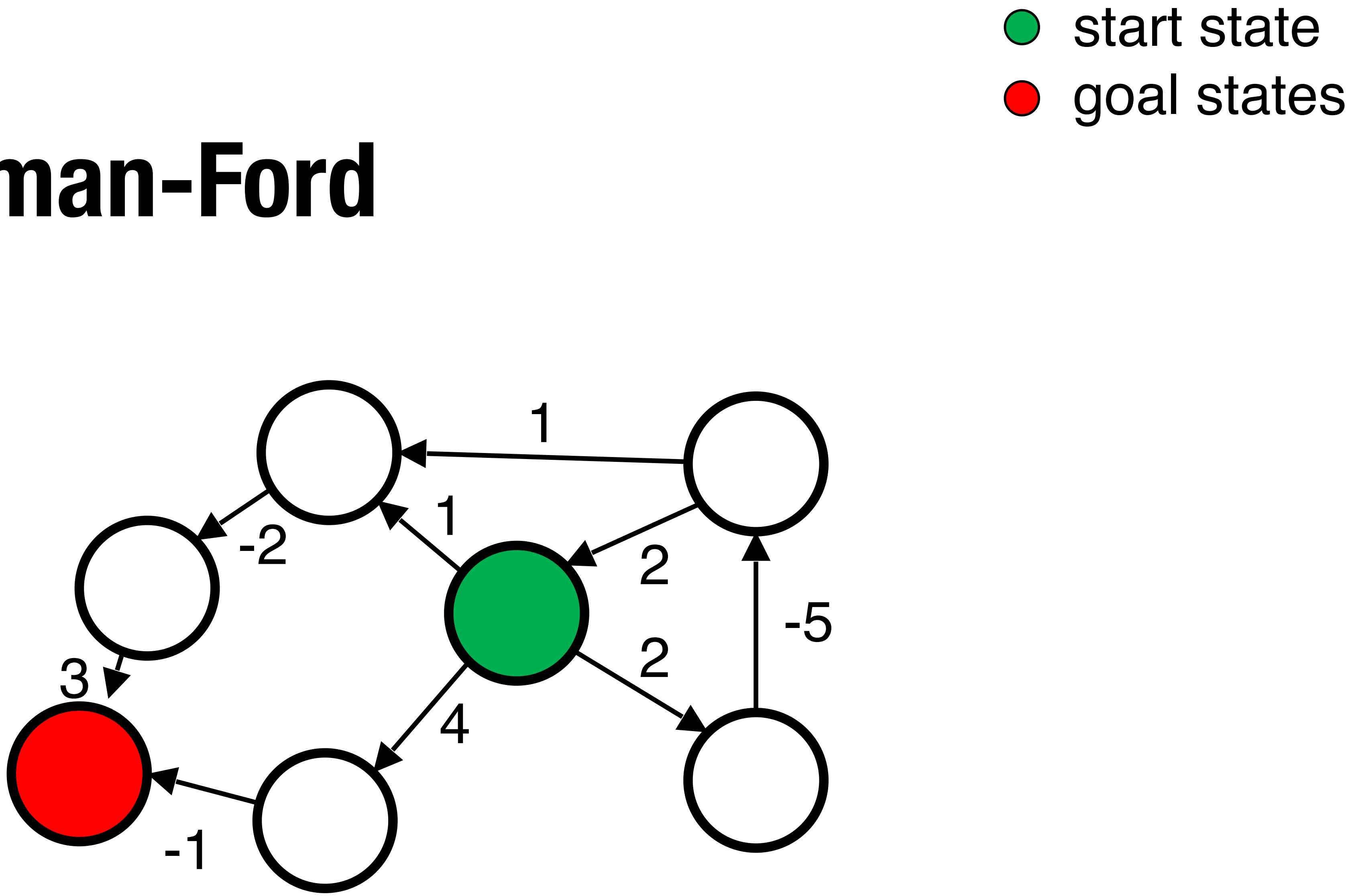


Termination?



changes on the V+1st iteration => negative cycle

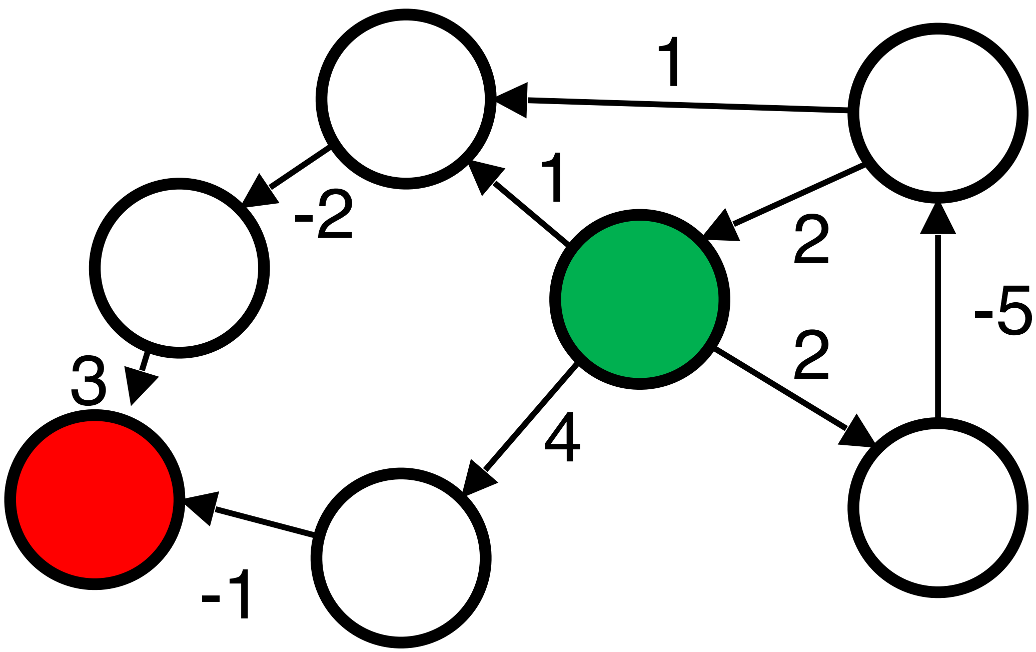
Bellman-Ford



What’s the run time?

start state
goal states

Bellman-Ford



Changes past iter $|V| \Rightarrow$ cycle

What's the run time? O(|V||E|)

Bellman-Ford Pseudocode

for **i** from 1 to |V| - 1:

    for edge (**u, v**) with weight **w**:

        if(dist(**v**) > dist(**u**) + **w**)

            dist(**v**) = dist(**u**) + **w**;

Time complexity O(|V||E|)

All-Pairs Shortest Path

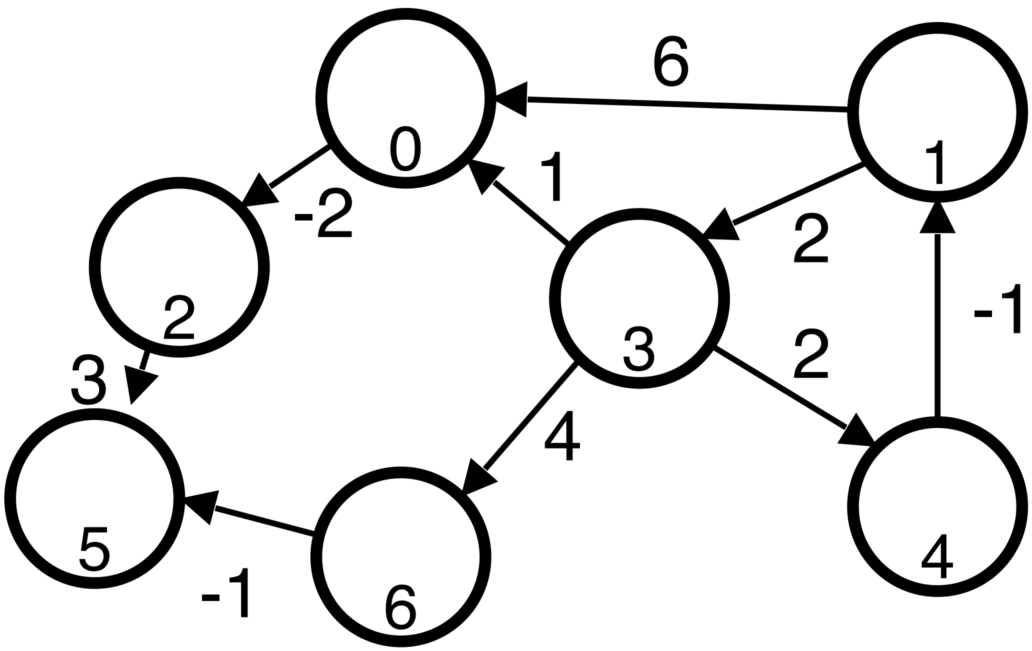
Given a weighted, directed graph (negative weights possible). For each pairs of vertices (**v**, **w**), return length of shortest path from **v** to **w**.

All-Pairs Shortest Path

Given a weighted, directed graph (negative weights possible). For each pairs of vertices (**v**, **w**), return length of shortest path from **v** to **w**.

Run Bellman-Ford for each **v**: $O(|V|^2|E|)$

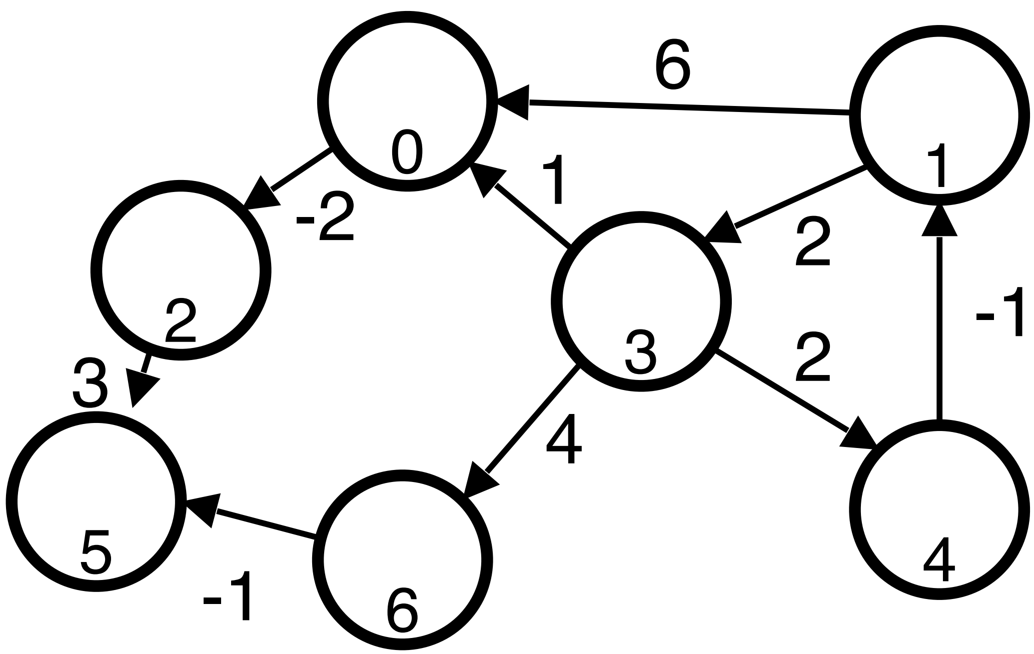
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & ? & ? & ? & ? & ? \\ ? & 0 & ? & ? & ? & ? & ? \\ ? & ? & 0 & ? & ? & ? & ? \\ ? & ? & ? & 0 & ? & ? & ? \\ ? & ? & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & ? & 0 \end{bmatrix}$$

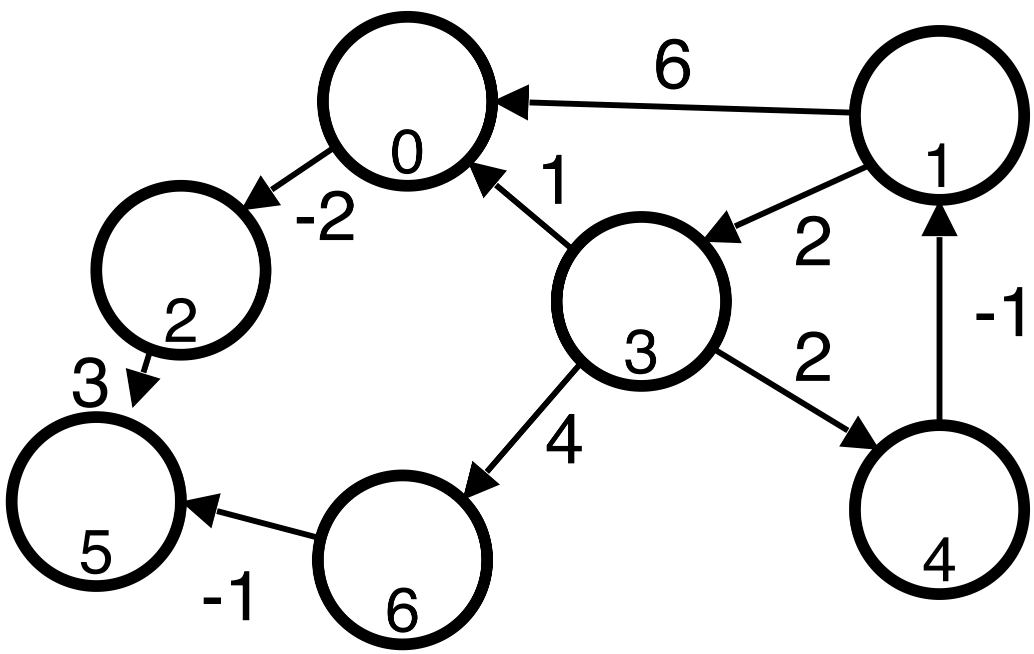
Vertex costs no longer useful; create **distance matrix**

All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & -2 & ? & ? & ? & ? \\ 6 & 0 & ? & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & ? & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

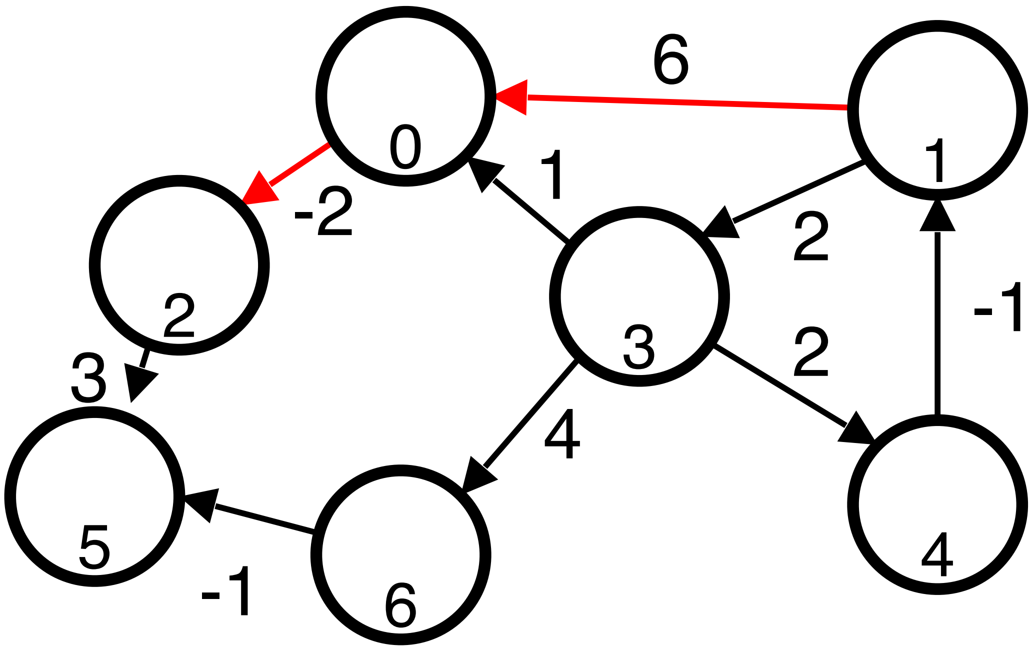
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & -2 & ? & ? & ? & ? \\ 6 & 0 & ? & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & ? & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 0?

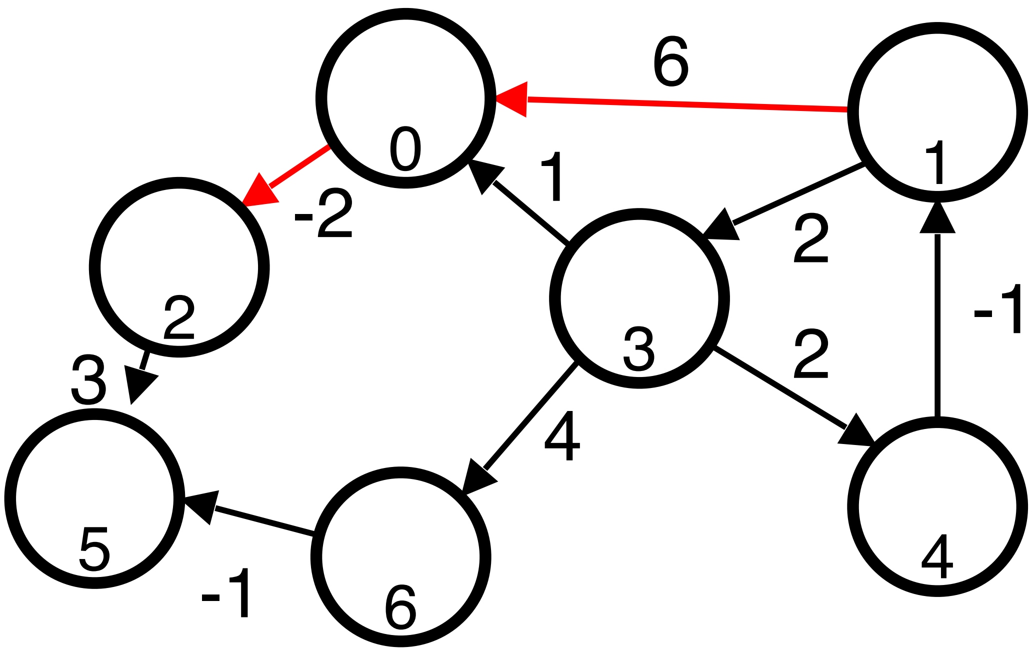
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & \boxed{-2} & ? & ? & ? & ? \\ \boxed{6} & 0 & \boxed{?} & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & ? & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 0?

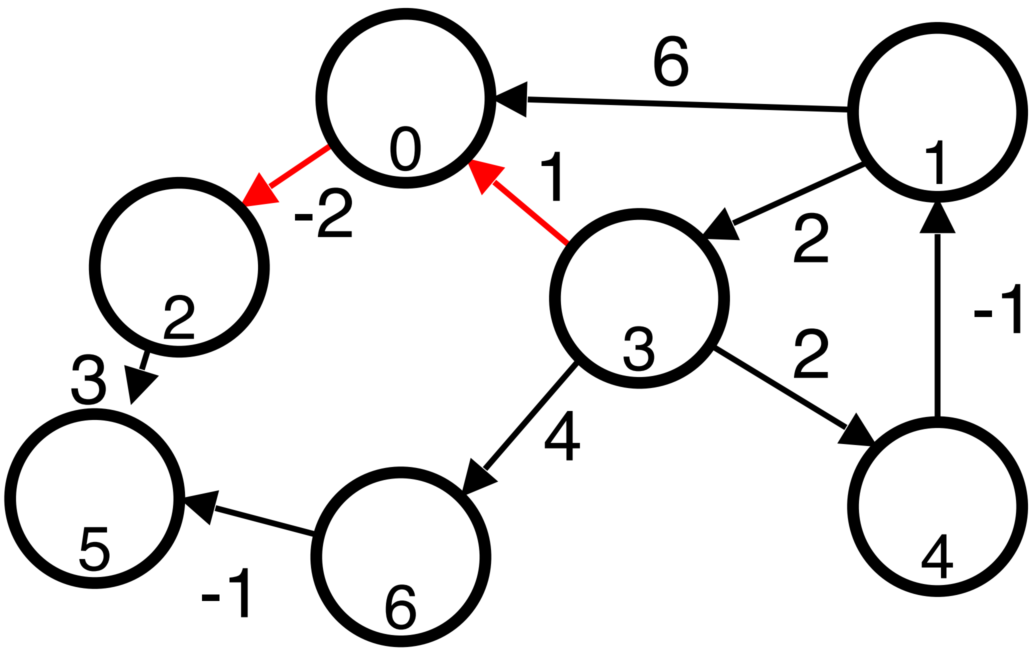
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & \boxed{-2} & ? & ? & ? & ? \\ \boxed{6} & 0 & \boxed{4} & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & ? & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 0?

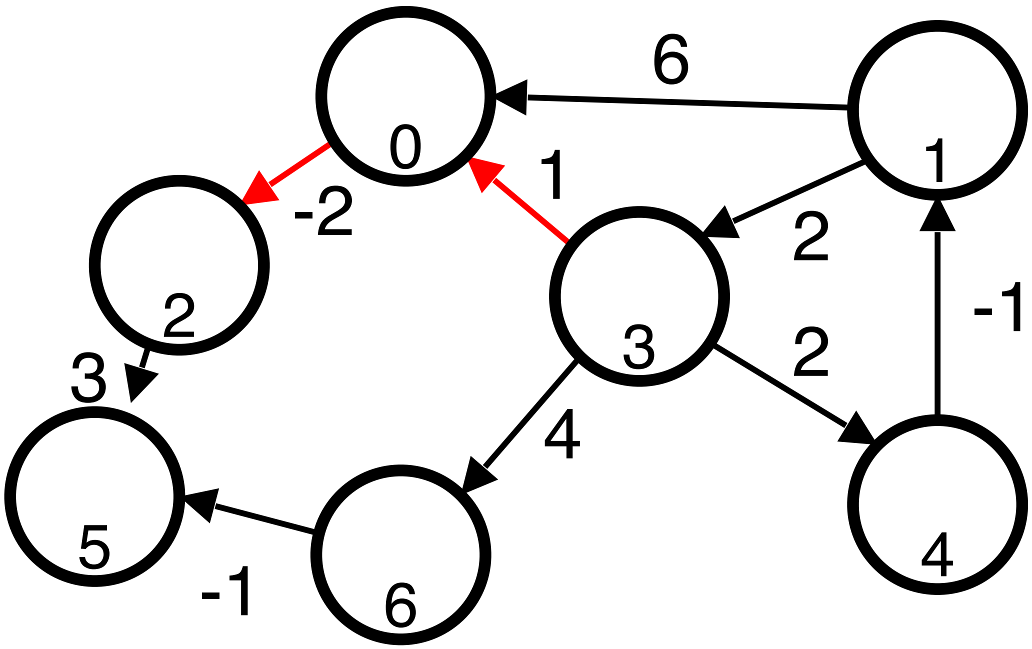
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & \boxed{-2} & ? & ? & ? & ? \\ 6 & 0 & 4 & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ \boxed{1} & ? & \boxed{?} & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 0?

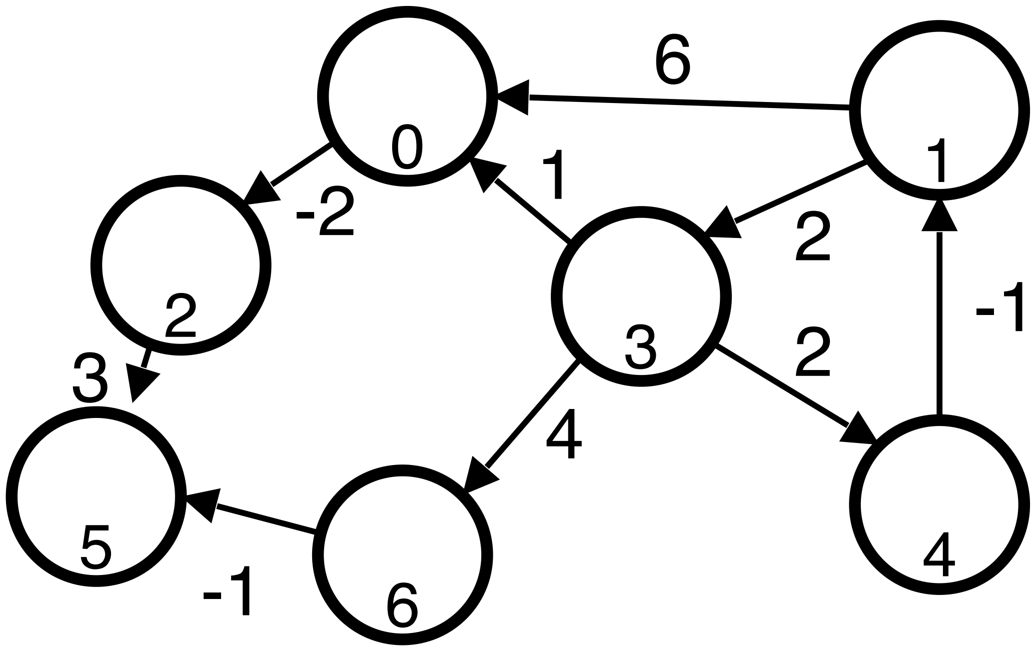
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & \boxed{-2} & ? & ? & ? & ? \\ 6 & 0 & 4 & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ \boxed{1} & ? & \boxed{-1} & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 0?

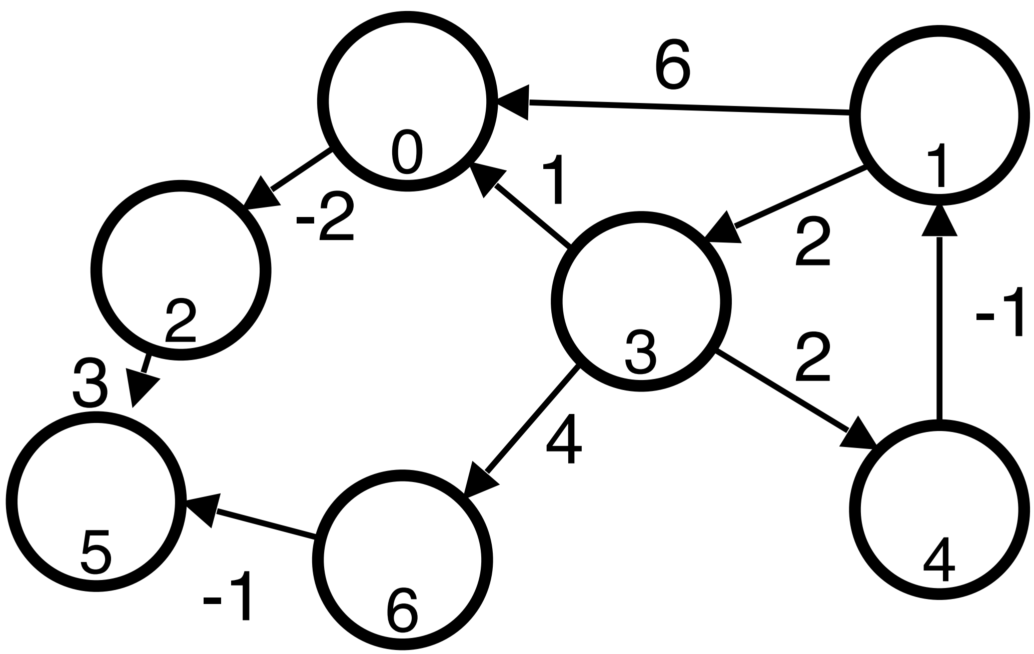
All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & -2 & ? & ? & ? & ? \\ 6 & 0 & 4 & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & -1 & 0 & 2 & ? & 4 \\ ? & -1 & ? & ? & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 1?

All-Pairs Shortest Path



$$\begin{bmatrix} 0 & ? & -2 & ? & ? & ? & ? \\ 6 & 0 & 4 & 2 & ? & ? & ? \\ ? & ? & 0 & ? & ? & 3 & ? \\ 1 & ? & -1 & 0 & 2 & ? & 4 \\ 5 & -1 & 3 & 1 & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? \\ ? & ? & ? & ? & ? & -1 & 0 \end{bmatrix}$$

Which paths can improve by passing through node 1?

Floyd-Warshall

```
for each vertex k:
  for each vertex v:
    for each vertex w:
      if( dist(v,w) > dist(v,k) + dist(k,w) )
        dist(v,w) = dist(v,k) + dist(k,w);
```

Floyd-Warshall

for each vertex **k**:
  for each vertex **v**:
    for each vertex **w**:
      if(dist(**v**,**w**) > dist(**v**,**k**) + dist(**k**,**w**))
        dist(**v**,**w**) = dist(**v**,**k**) + dist(**k**,**w**);

Claim: if no negative cycles, will find all shortest paths

Floyd-Warshall

```
for each vertex k:
  for each vertex v:
    for each vertex w:
      if( dist(v,w) > dist(v,k) + dist(k,w) )
        dist(v,w) = dist(v,k) + dist(k,w);
```

How can we tell if there are negative cycles?

Floyd-Warshall

```
for each vertex k:
  for each vertex v:
    for each vertex w:
      if( dist(v,w) > dist(v,k) + dist(k,w) )
        dist(v,w) = dist(v,k) + dist(k,w);
```

How can we tell if there are negative cycles?

Check for dist(**v,v**) < 0

Floyd-Warshall

for each vertex **k**:
  for each vertex **v**:
    for each vertex **w**:
      if(dist(**v**,**w**) > dist(**v**,**k**) + dist(**k**,**w**))
        dist(**v**,**w**) = dist(**v**,**k**) + dist(**k**,**w**);

Danger: watch for overflow is using **MAXINT** to denote missing edges

Floyd-Warshall

```
for each vertex k:
  for each vertex v:
    for each vertex w:
      if( dist(v,w) > dist(v,k) + dist(k,w) )
        dist(v,w) = dist(v,k) + dist(k,w);
```

Time complexity $O(|V|^3)$