Competitive Algorithm Design and Practice
Shortest Path
2014/03/26

Guan Yu, Chen (kevinx6000)

kevinx6000@gmail.com

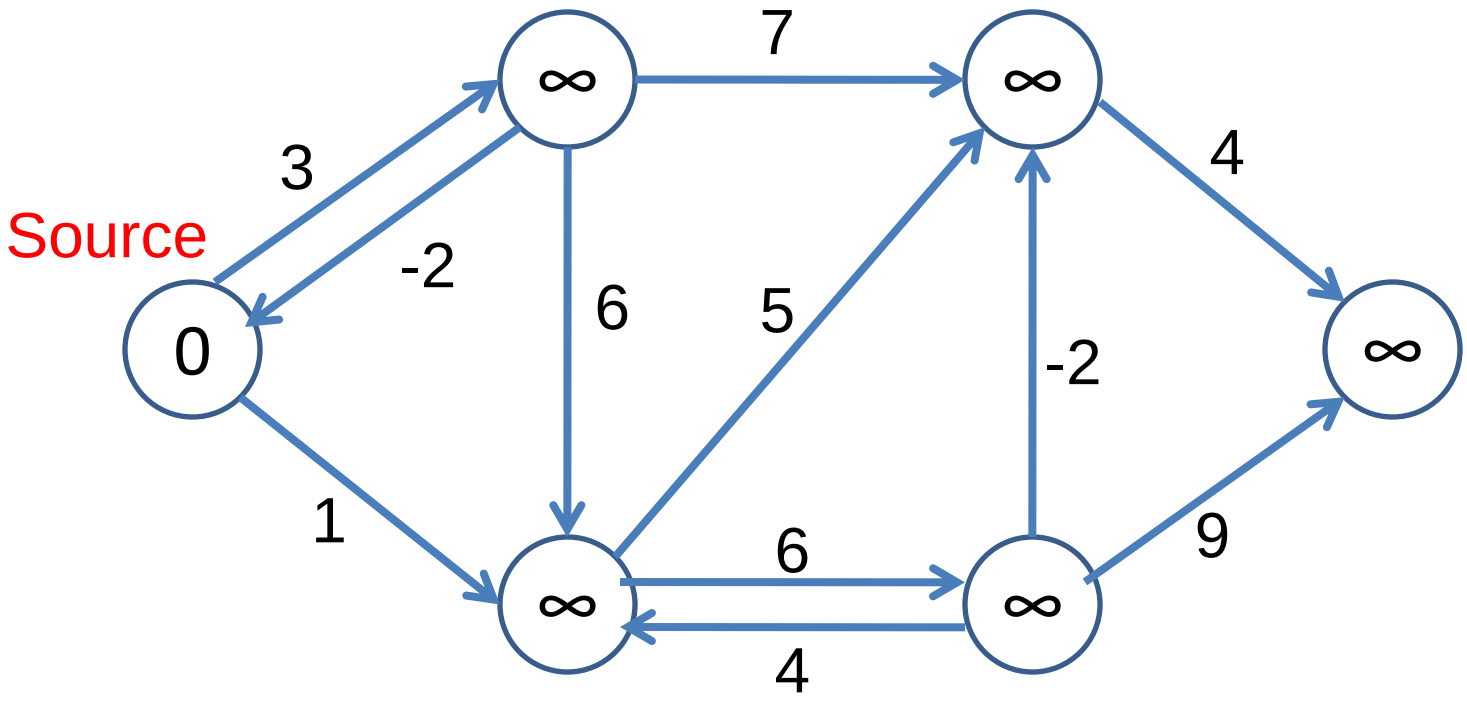
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

Outline

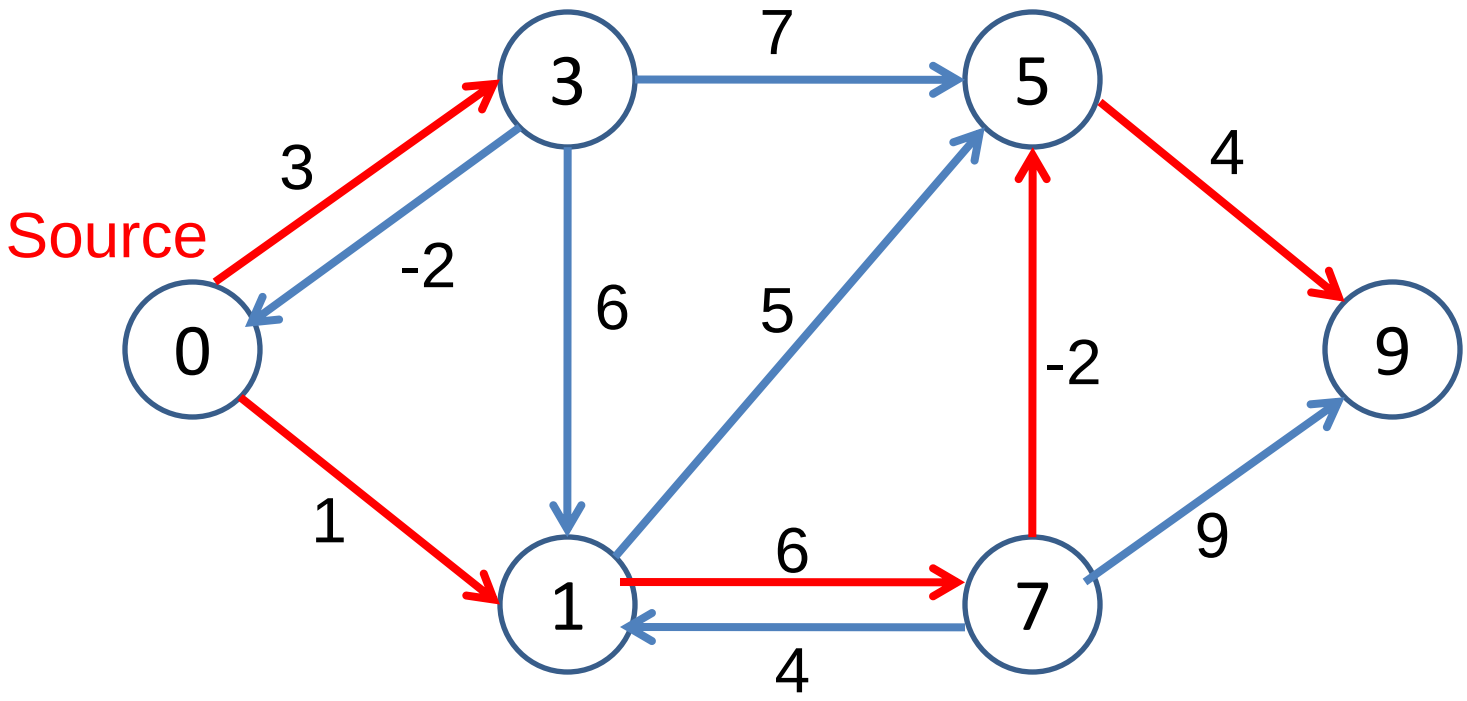
- Single Source Shortest Path
  - Relaxation
  - Bellman Ford
  - SPFA
- All Pair Shortest Path
  - Floyd

Singe Source Shortest Path

SSSP



SSSP



SSSP

- How?
  - Greedy?
  - BFS?
  - Backtracking?

SSSP

- How?
  - Greedy? – WA if not greedy properly..
  - BFS? – WA, only for un-weighted shortest path
  - Backtracking? – TLE

- See another slide for more details

SSSP - Algorithm

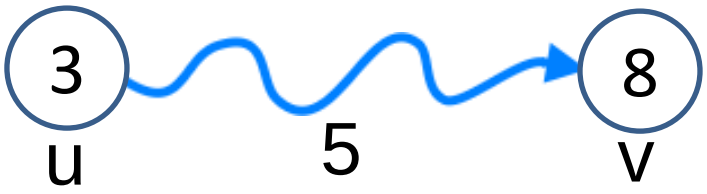
- Bellman Ford
- SPFA
- Dijkstra
- And more…


acm International Collegiate Programming Contest
IBM event sponsor

Relaxation

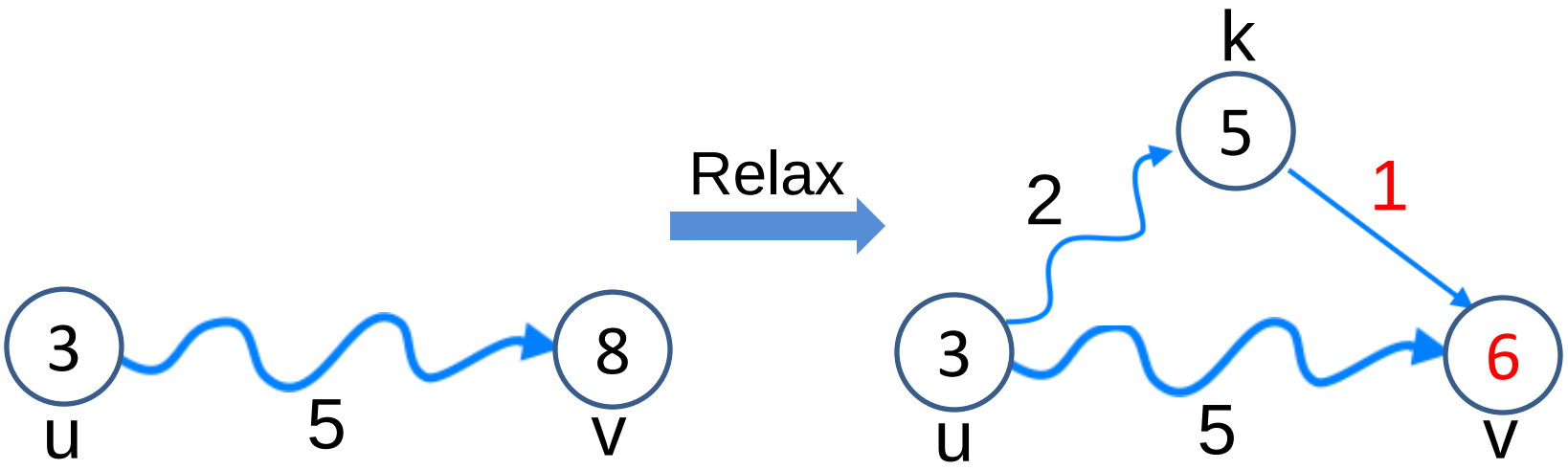
Relaxation

- Triangle Inequality



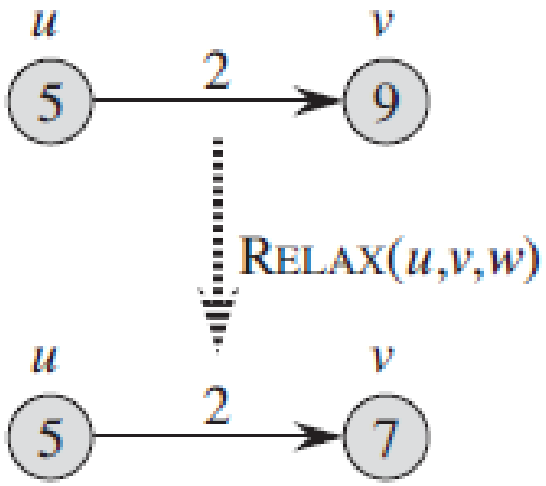
Relaxation

- Triangle Inequality

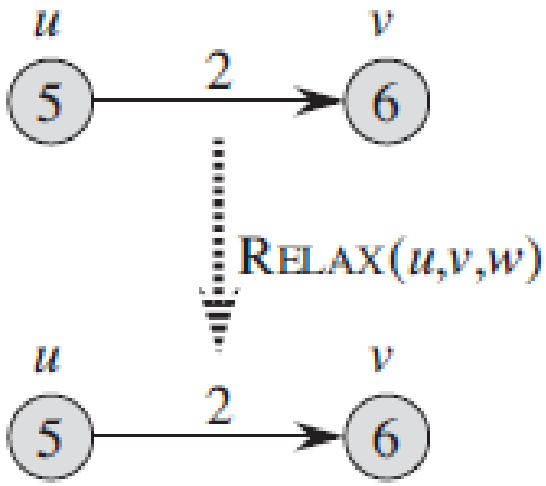


Relaxation

- Examples



Shorter than before



Remain unchanged

Relaxation

- Pseudo code

```
1  Relax(u,v,w){
2      if(dis[u]+w(u,v)<dis[v])
3          dis[v]=dis[u]+w(u,v);
4  }
```

Bellman Ford

Bellman Ford

- Relax all edges in graph
  - Totally n-1 times

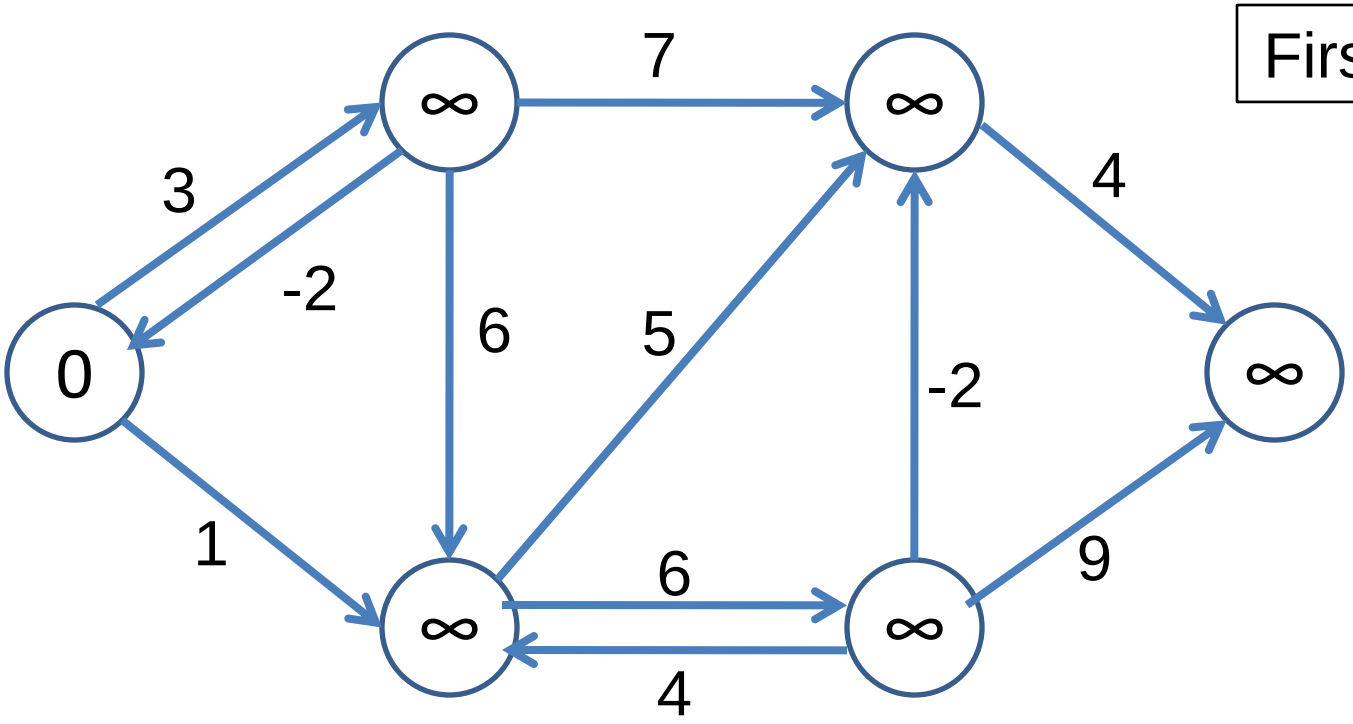
Bellman Ford

- Relax all edges in graph
  - Totally n-1 times
- Always n-1 times?
  - Stop when all edges stop relaxing

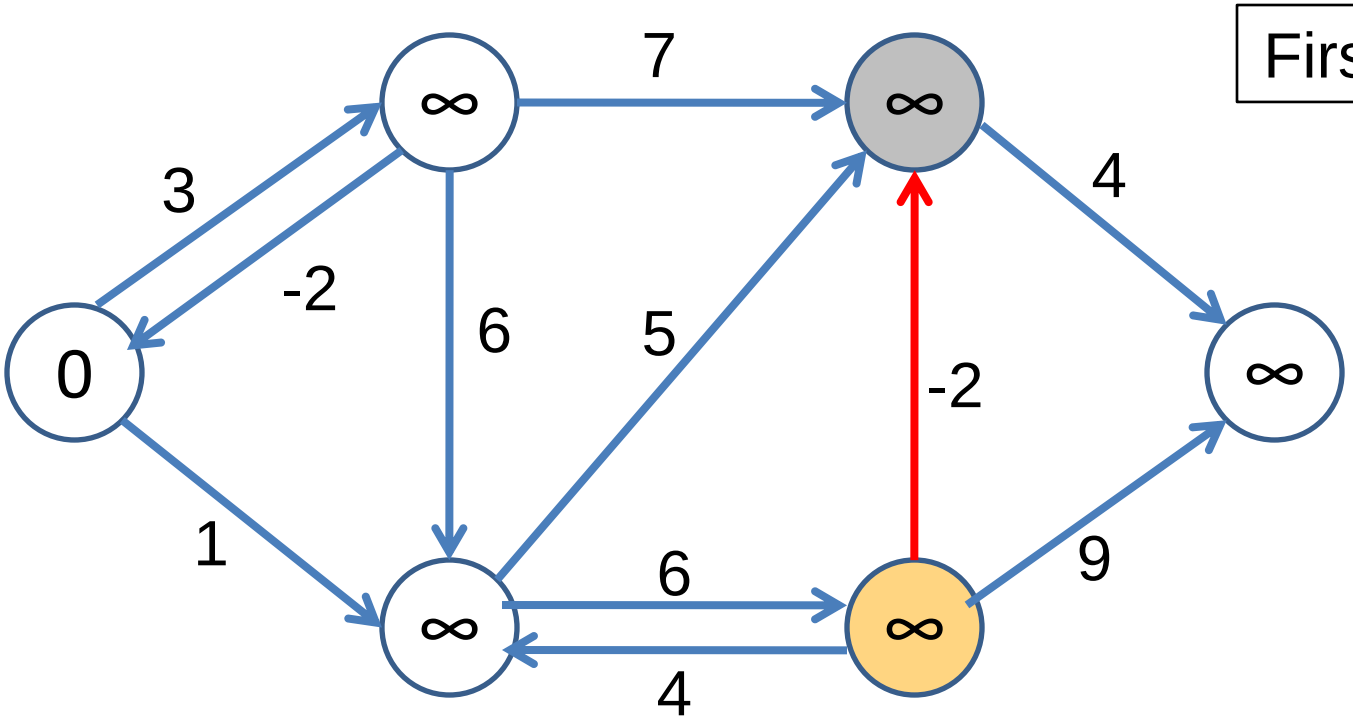
Bellman Ford

- Relax all edges in graph
  - Totally n-1 times
- Always n-1 times?
  - Stop when all edges stop relaxing
- Complexity
  - O(VE)

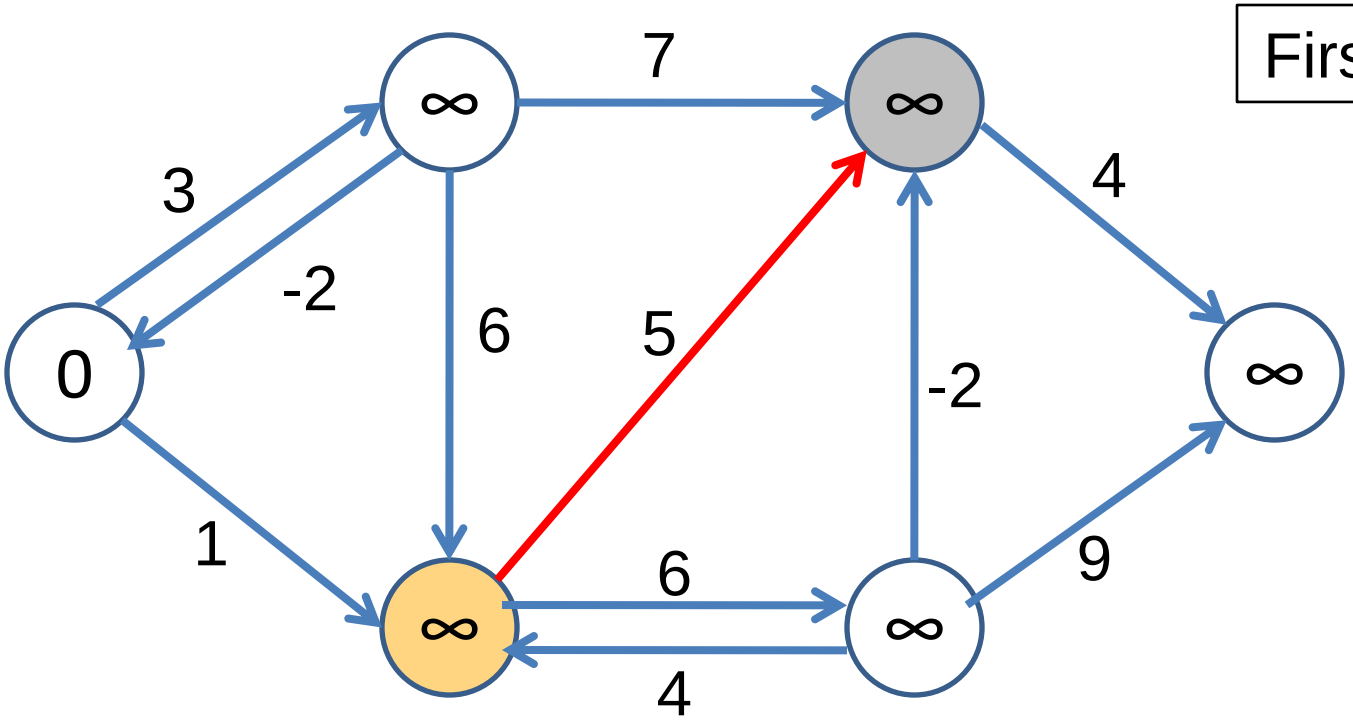
Bellman Ford



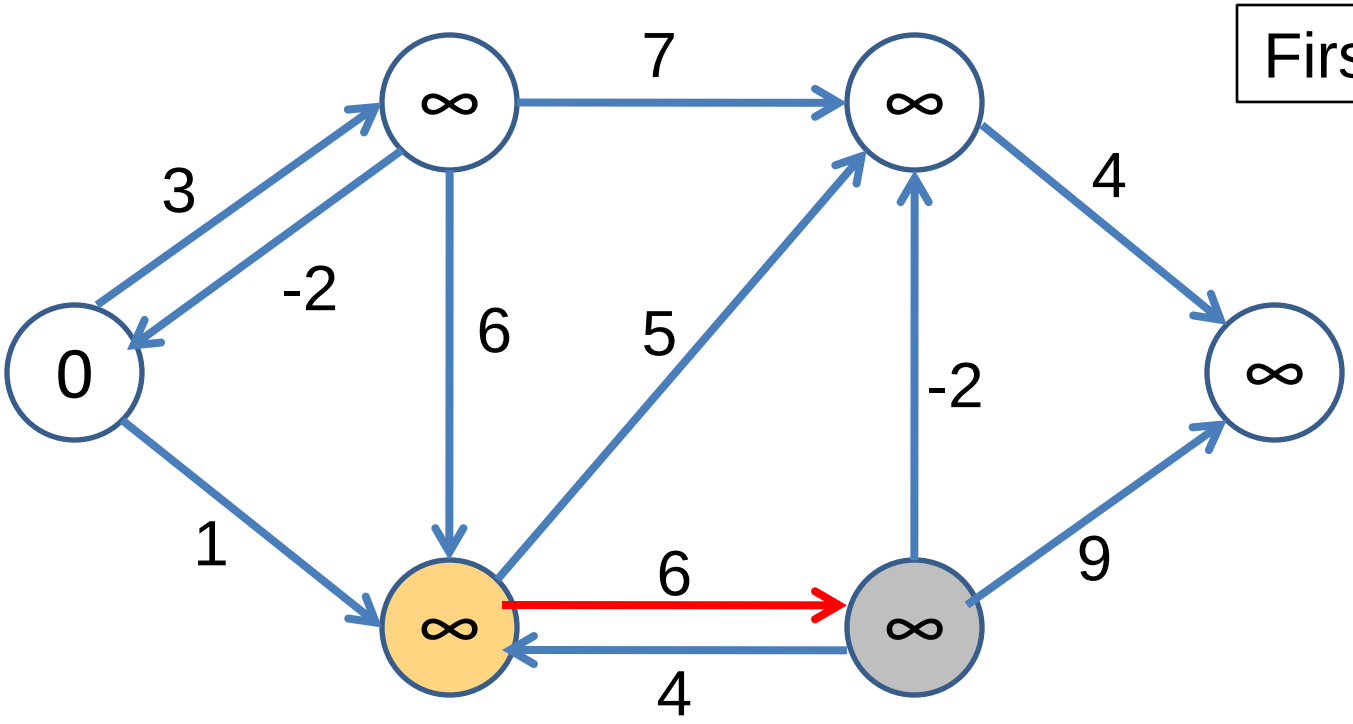
Bellman Ford



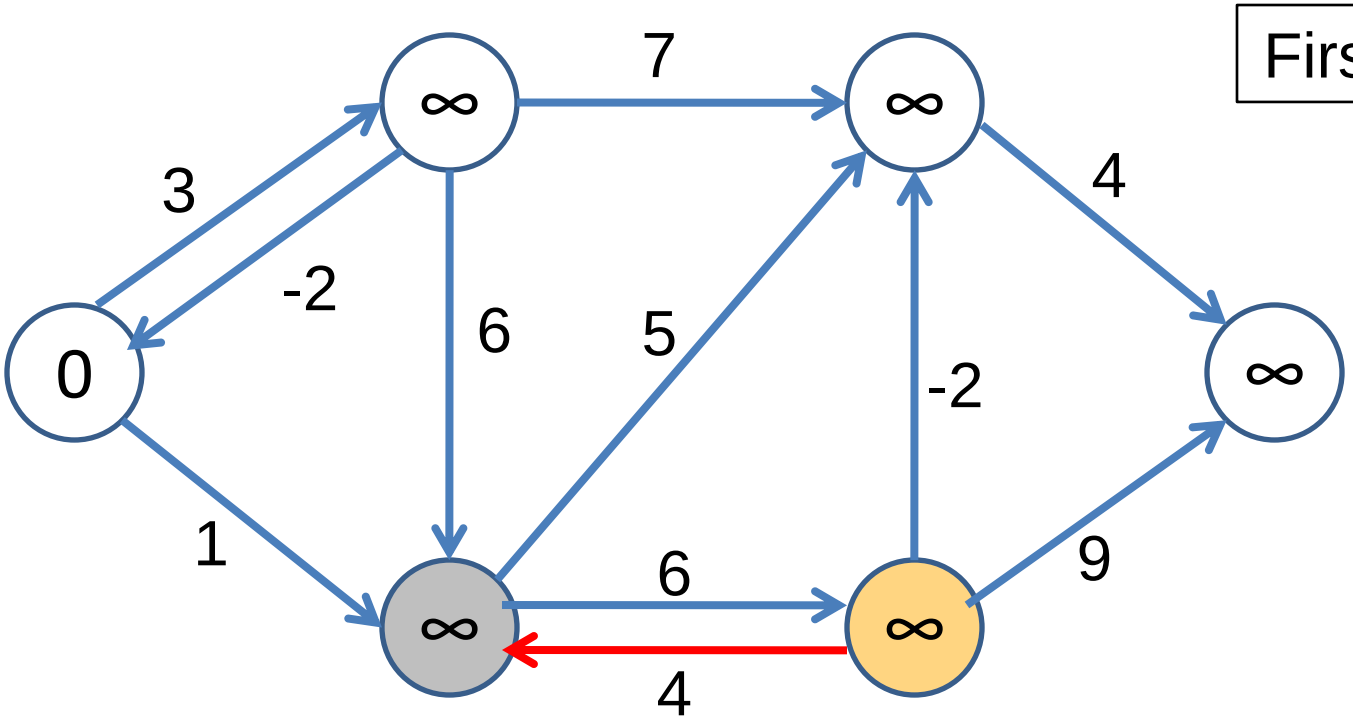
Bellman Ford



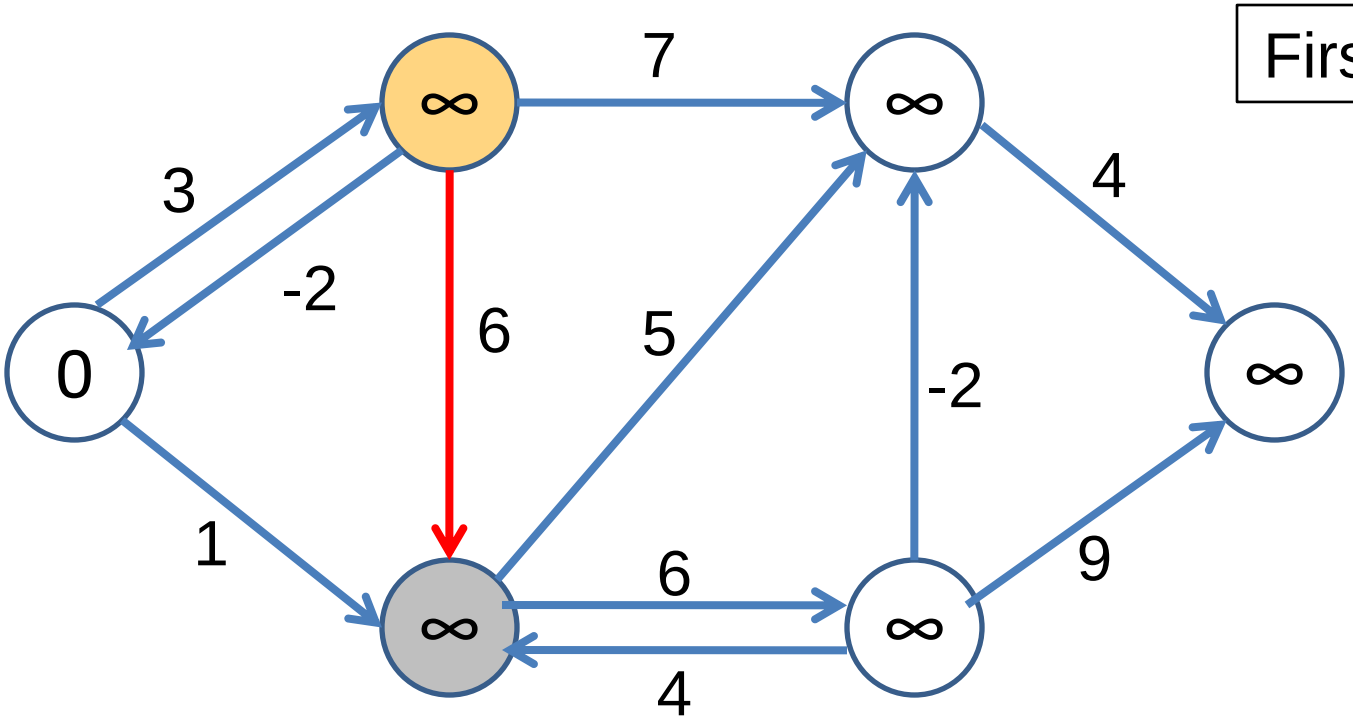
Bellman Ford



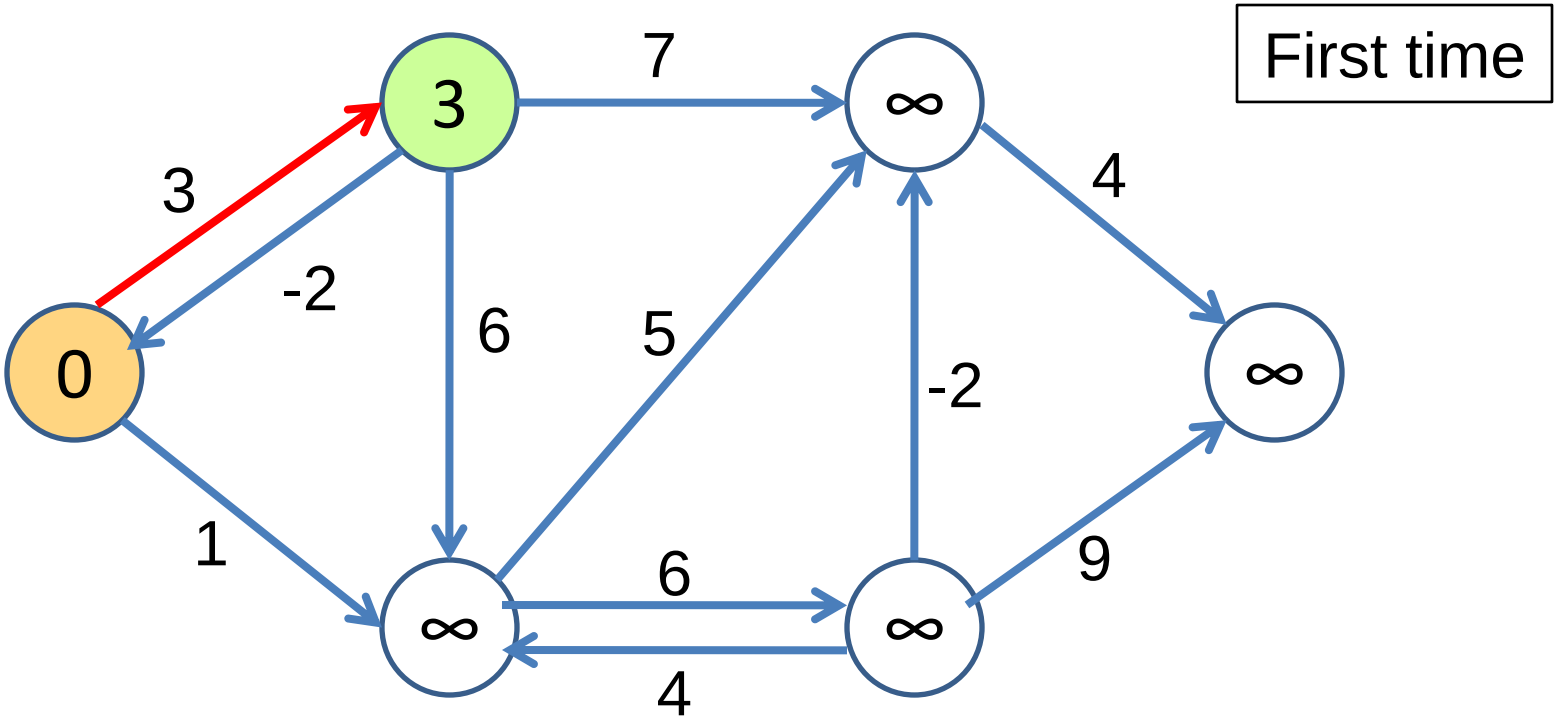
Bellman Ford



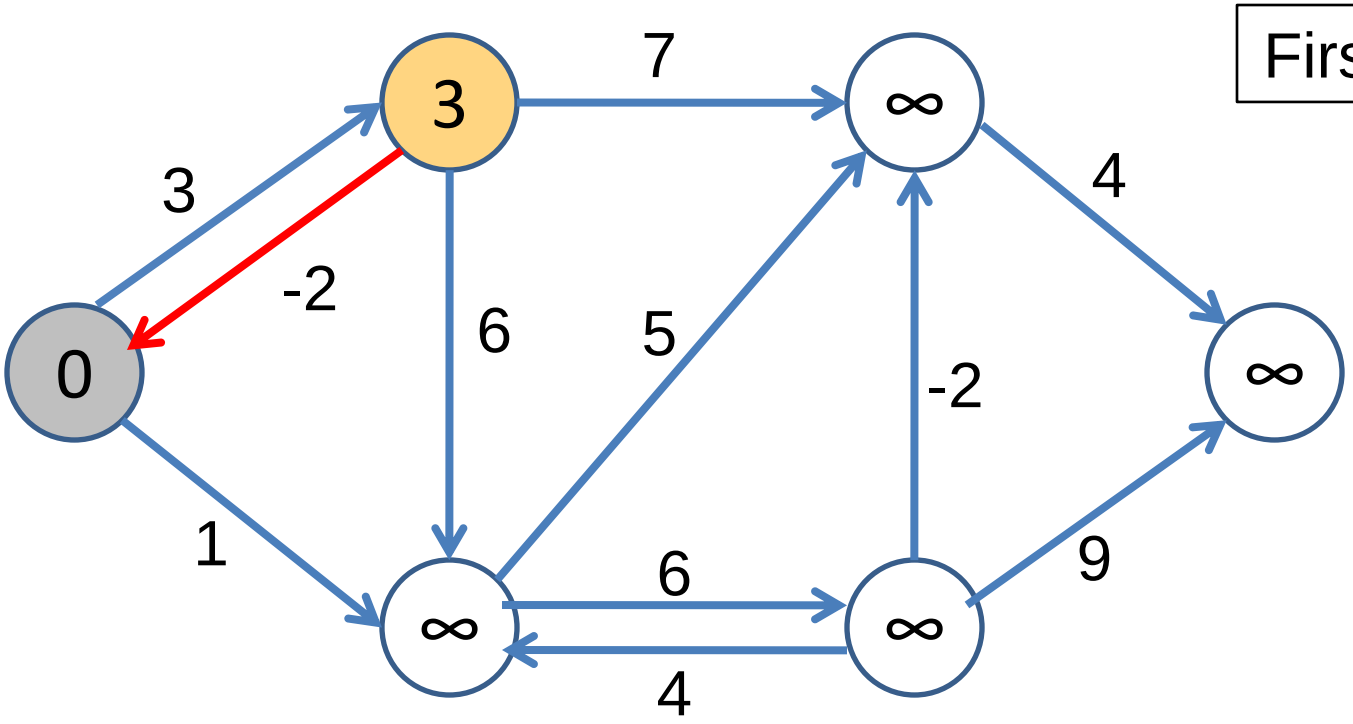
Bellman Ford



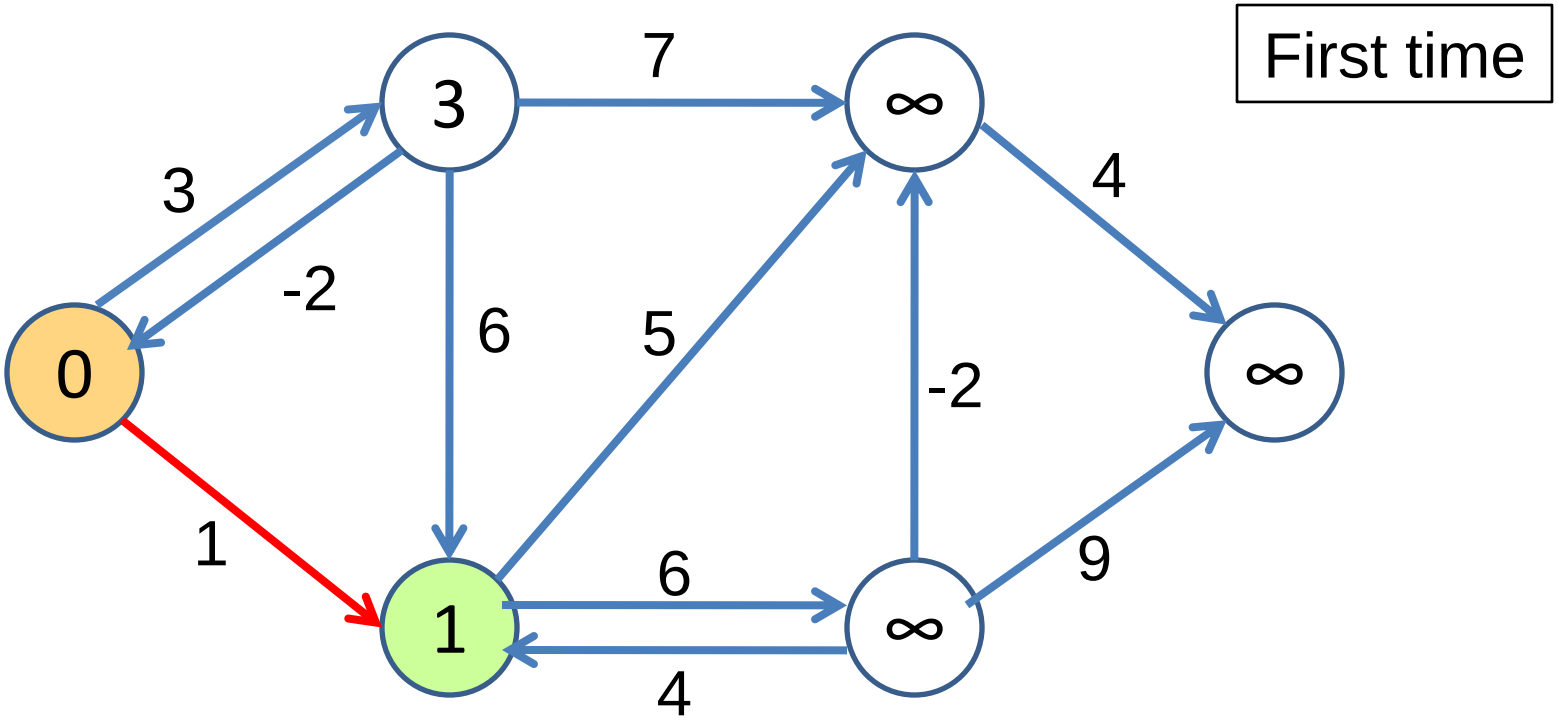
Bellman Ford



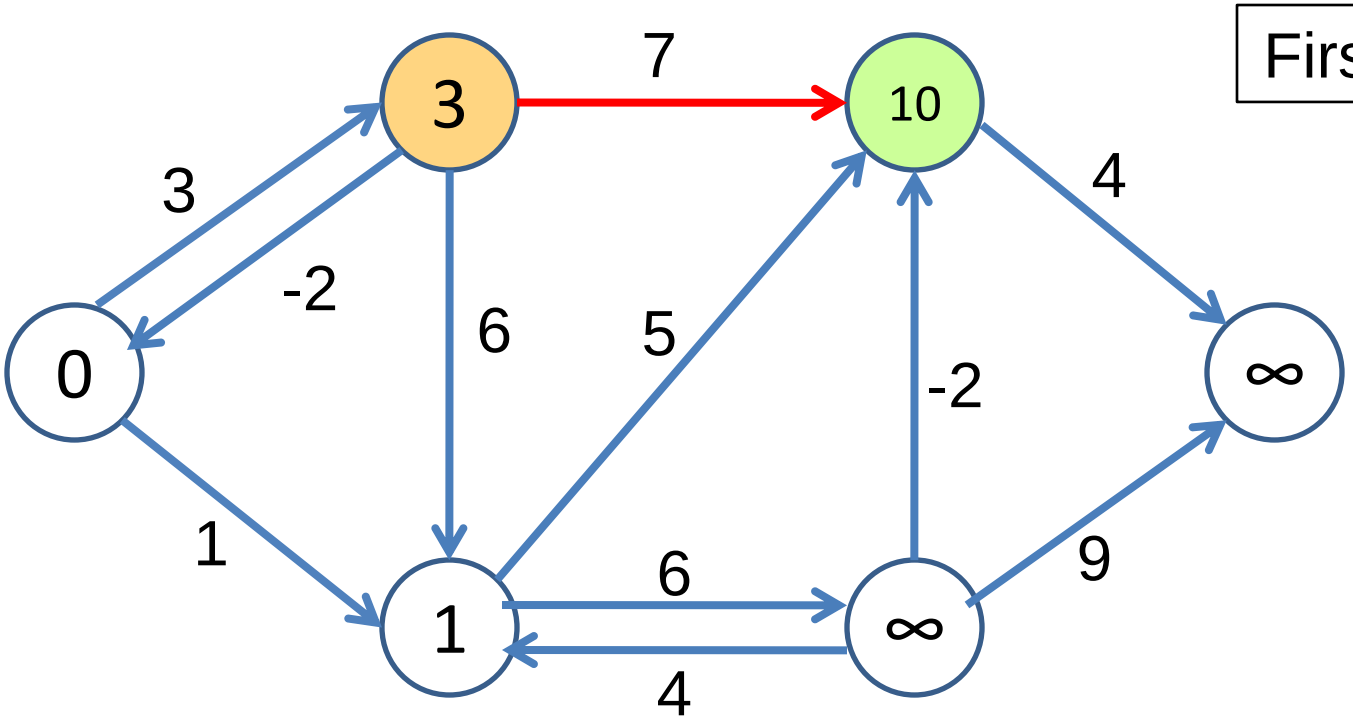
Bellman Ford



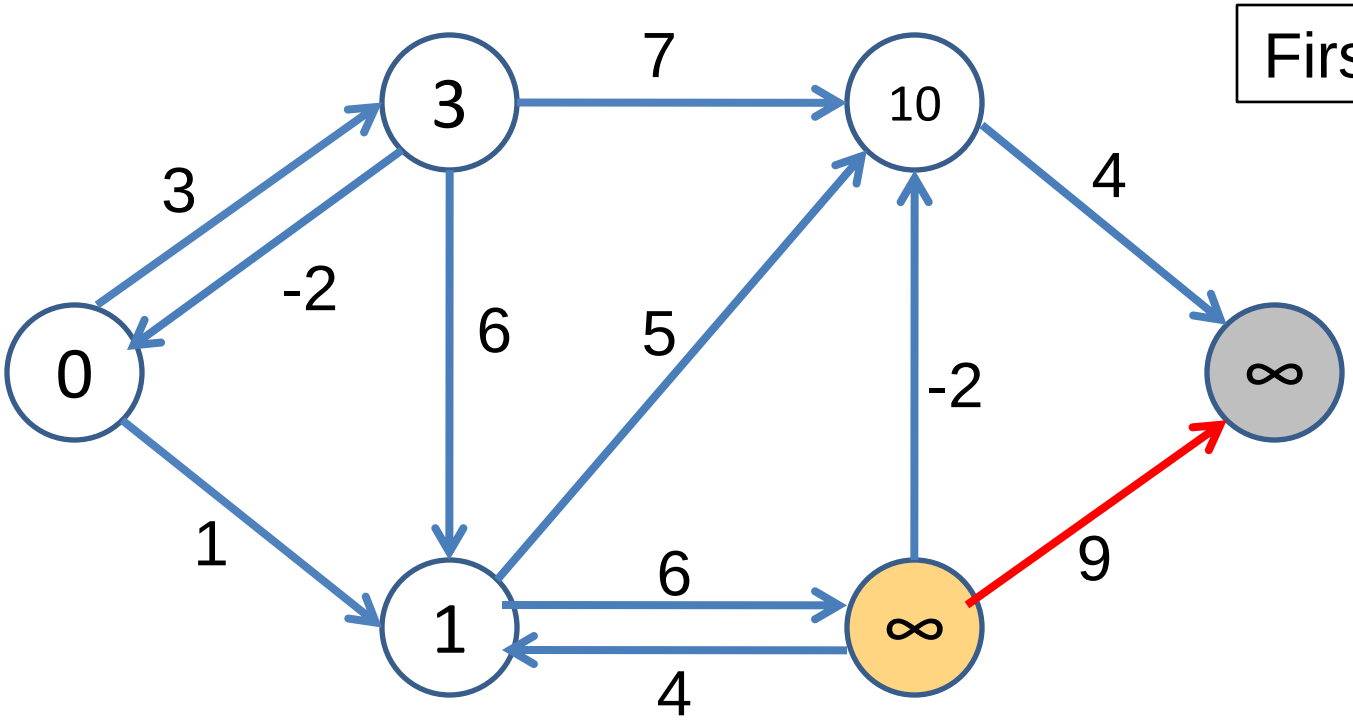
Bellman Ford



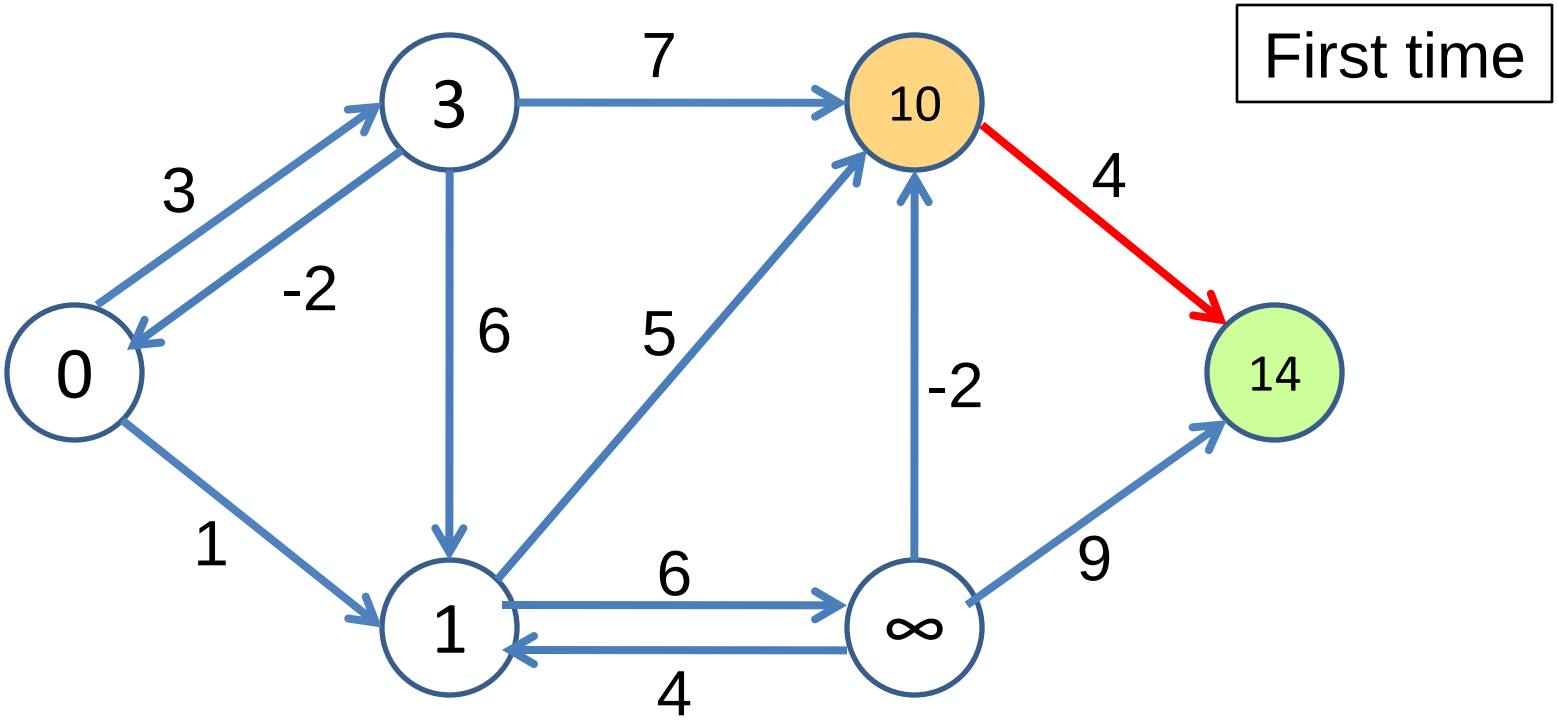
Bellman Ford



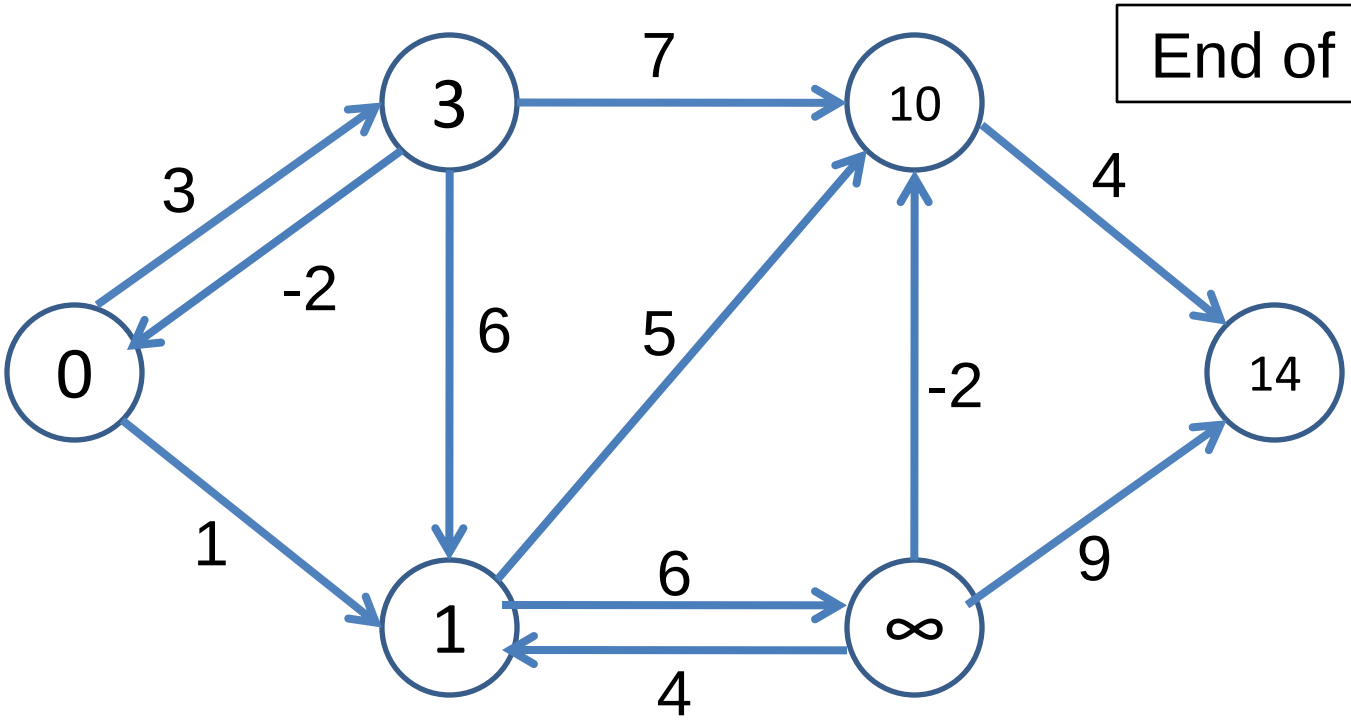
Bellman Ford



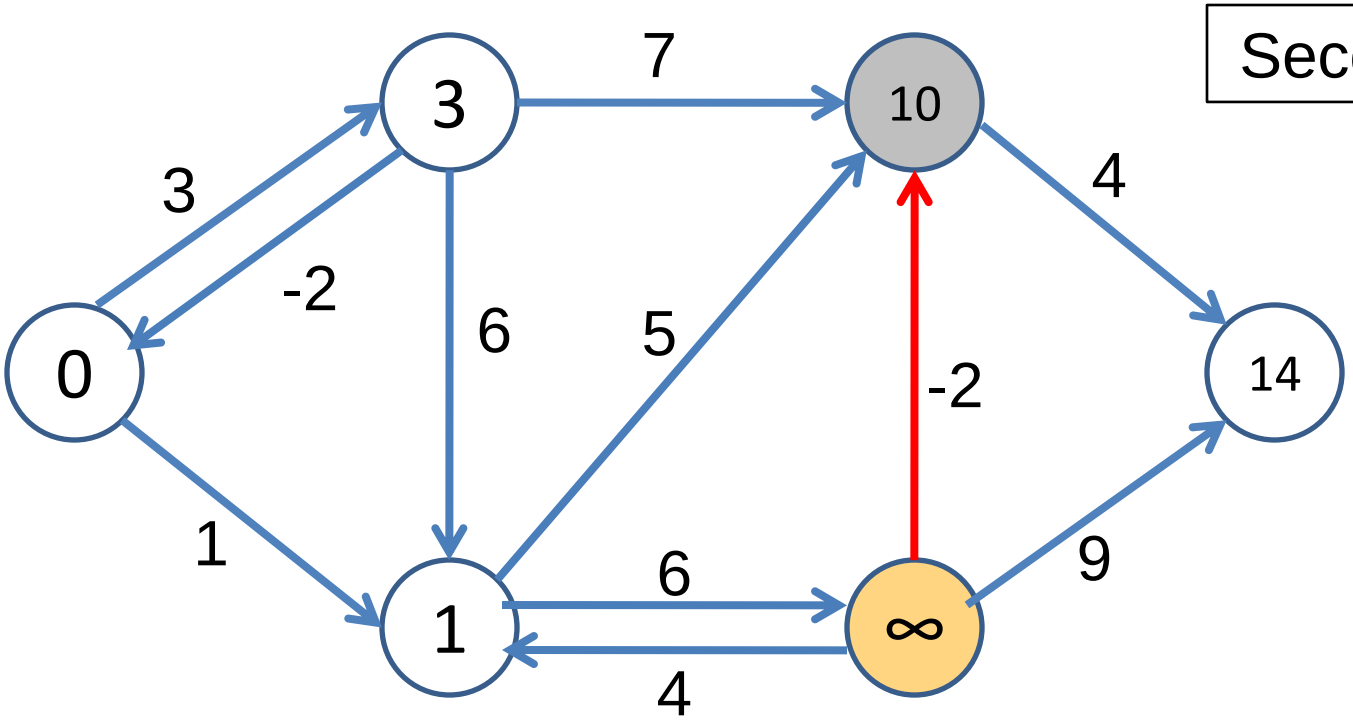
Bellman Ford



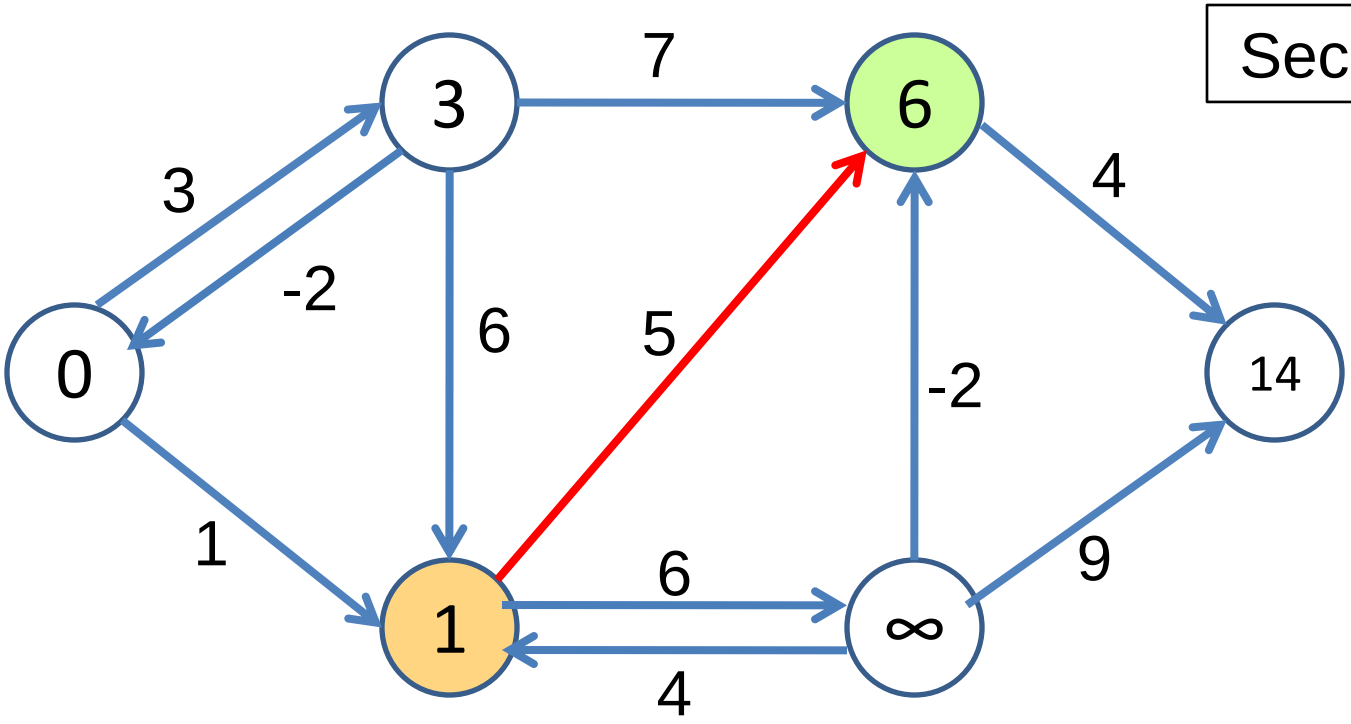
Bellman Ford



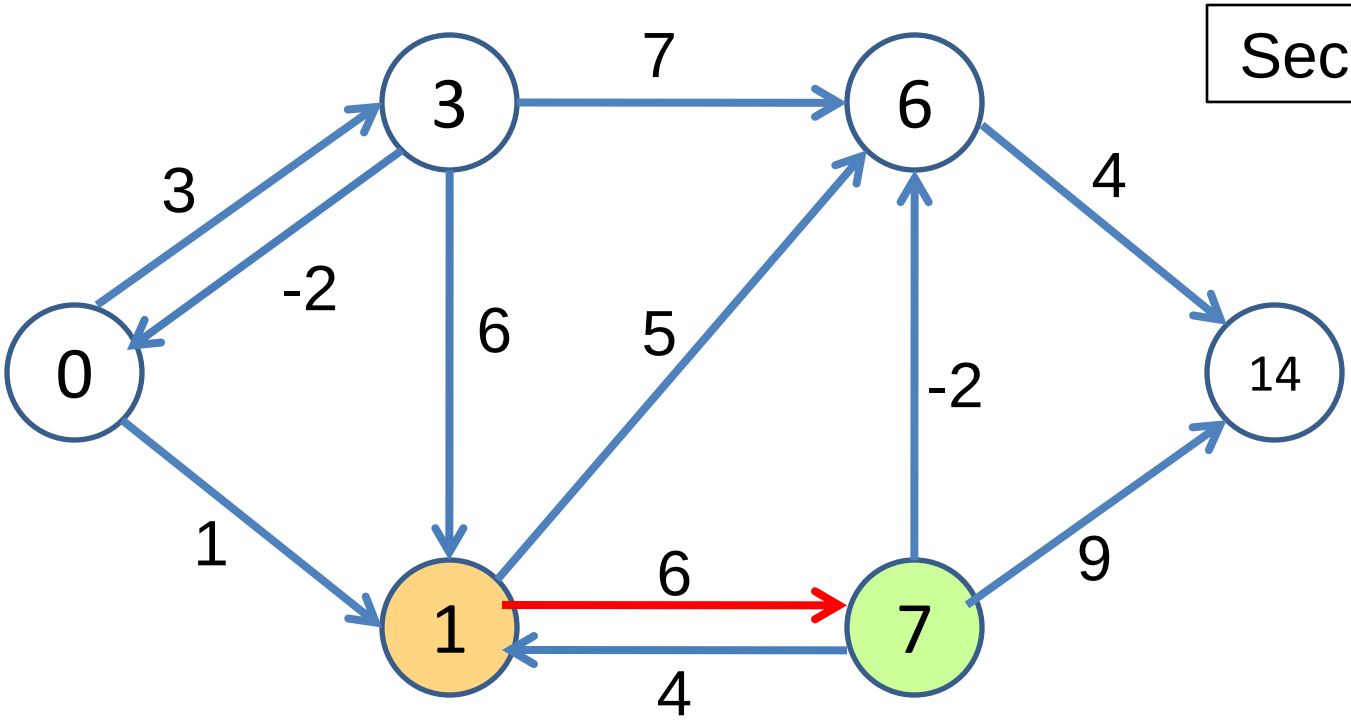
Bellman Ford



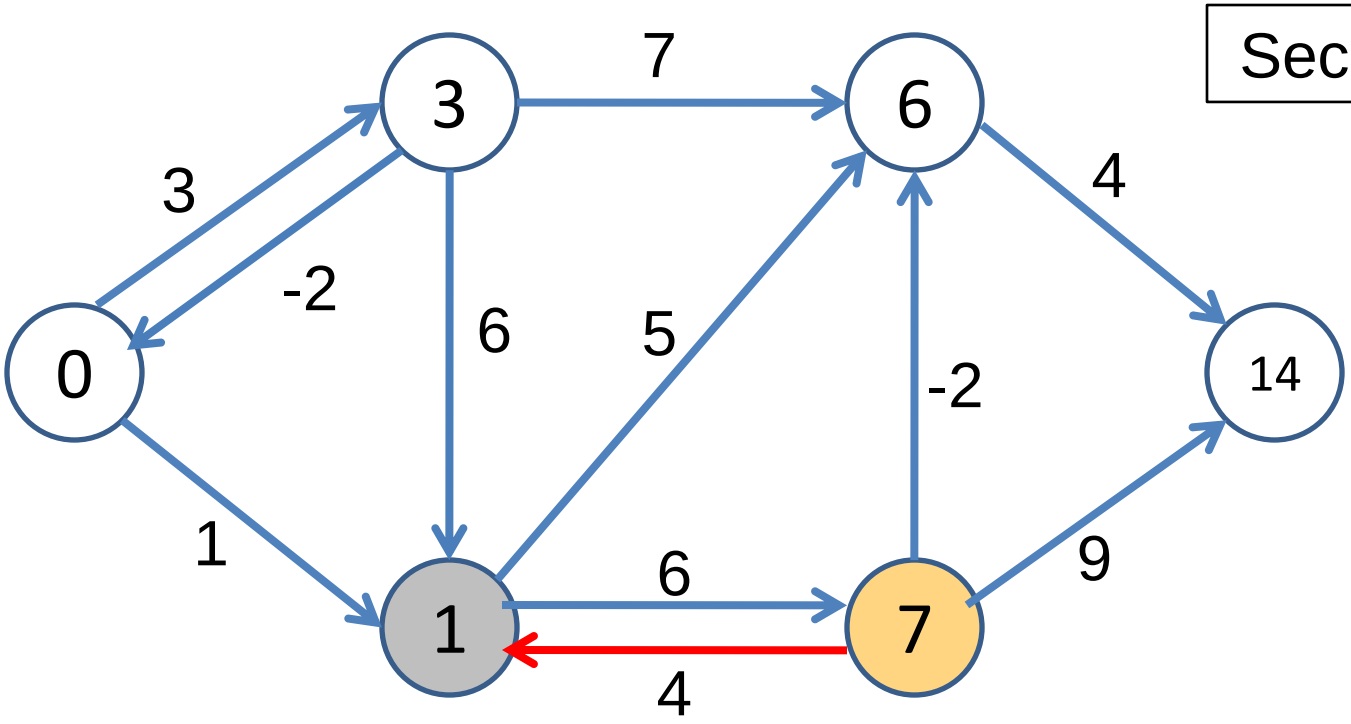
Bellman Ford



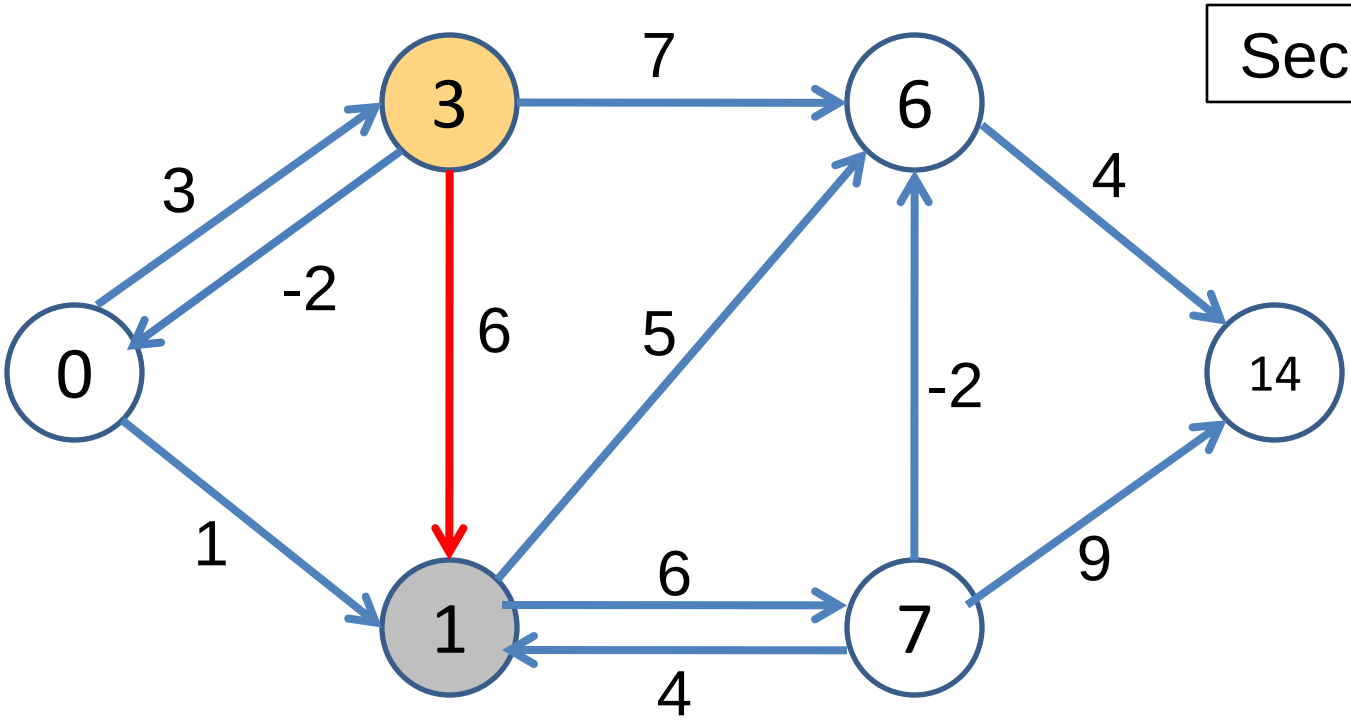
Bellman Ford



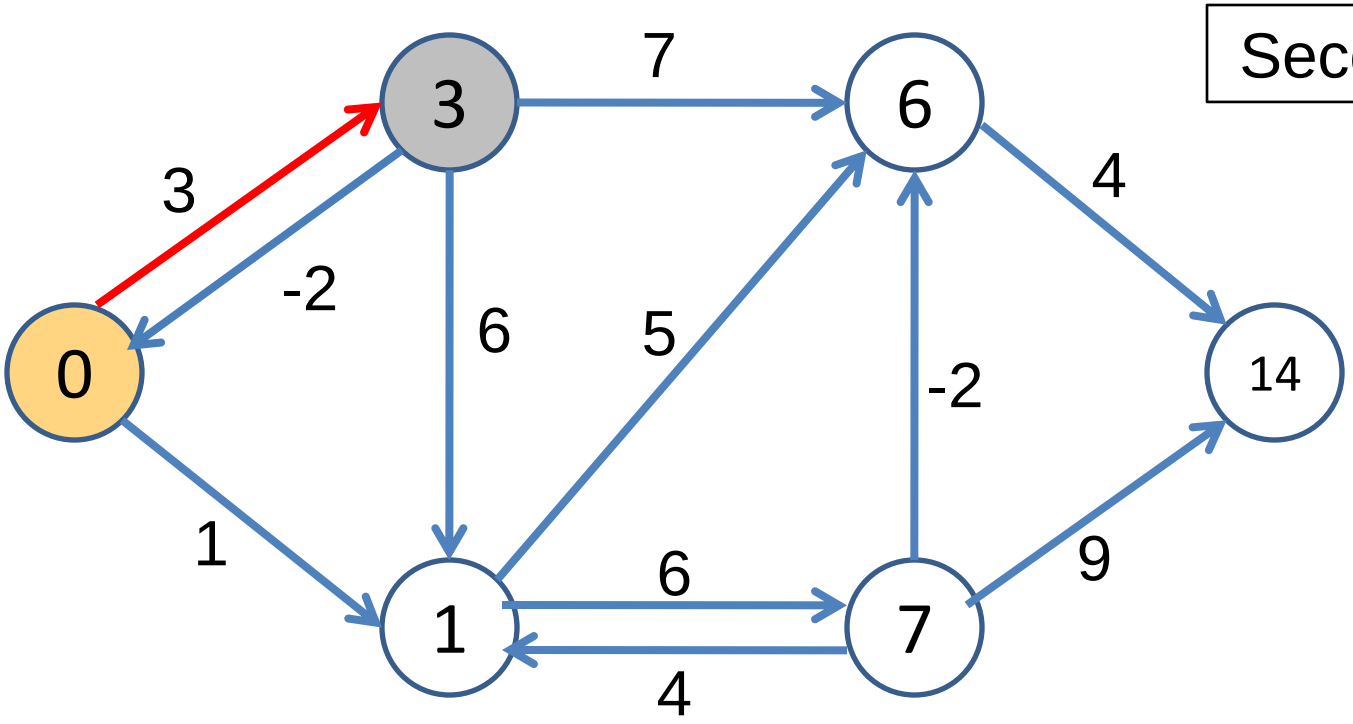
Bellman Ford



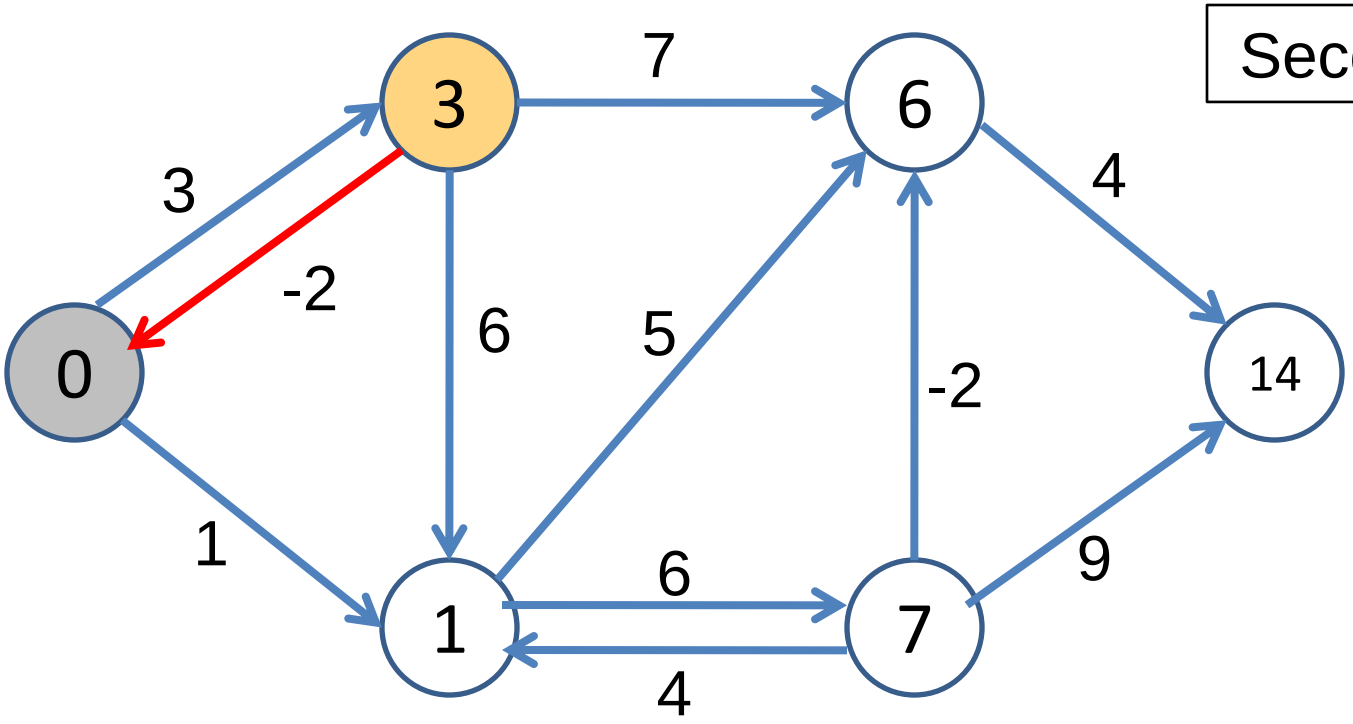
Bellman Ford



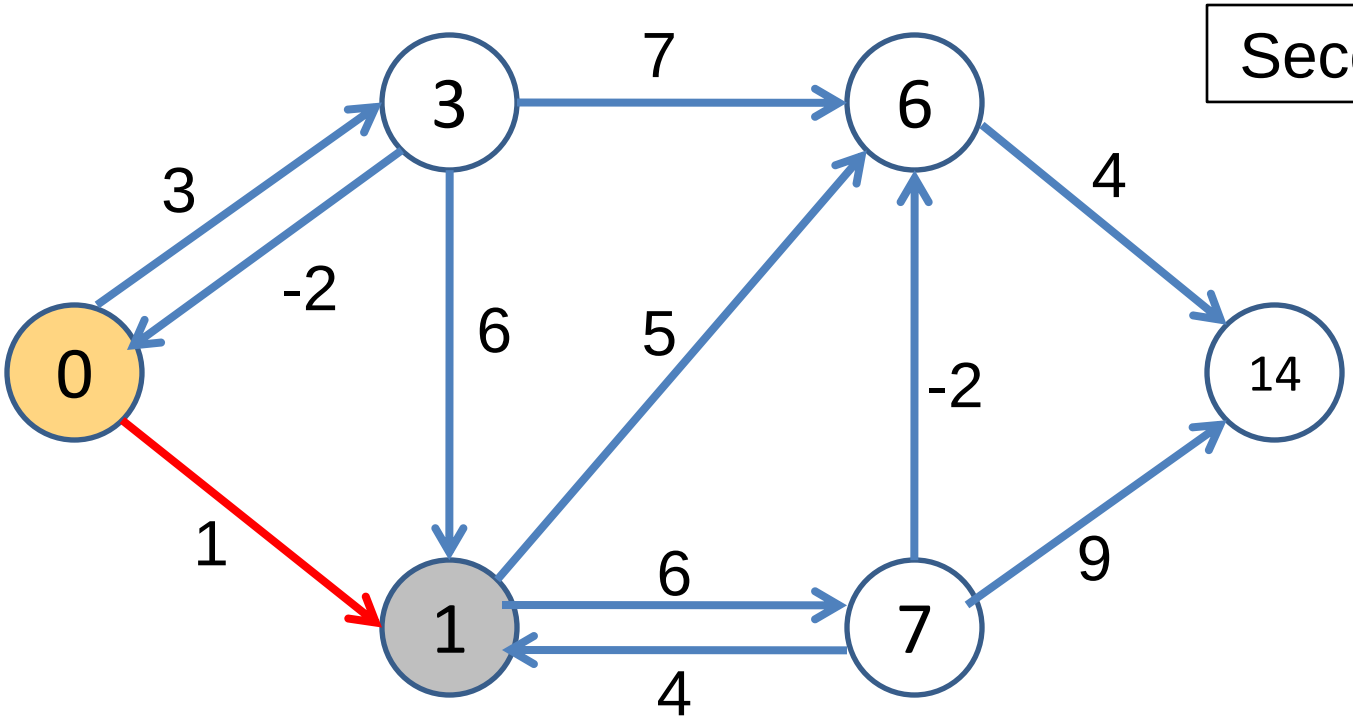
Bellman Ford



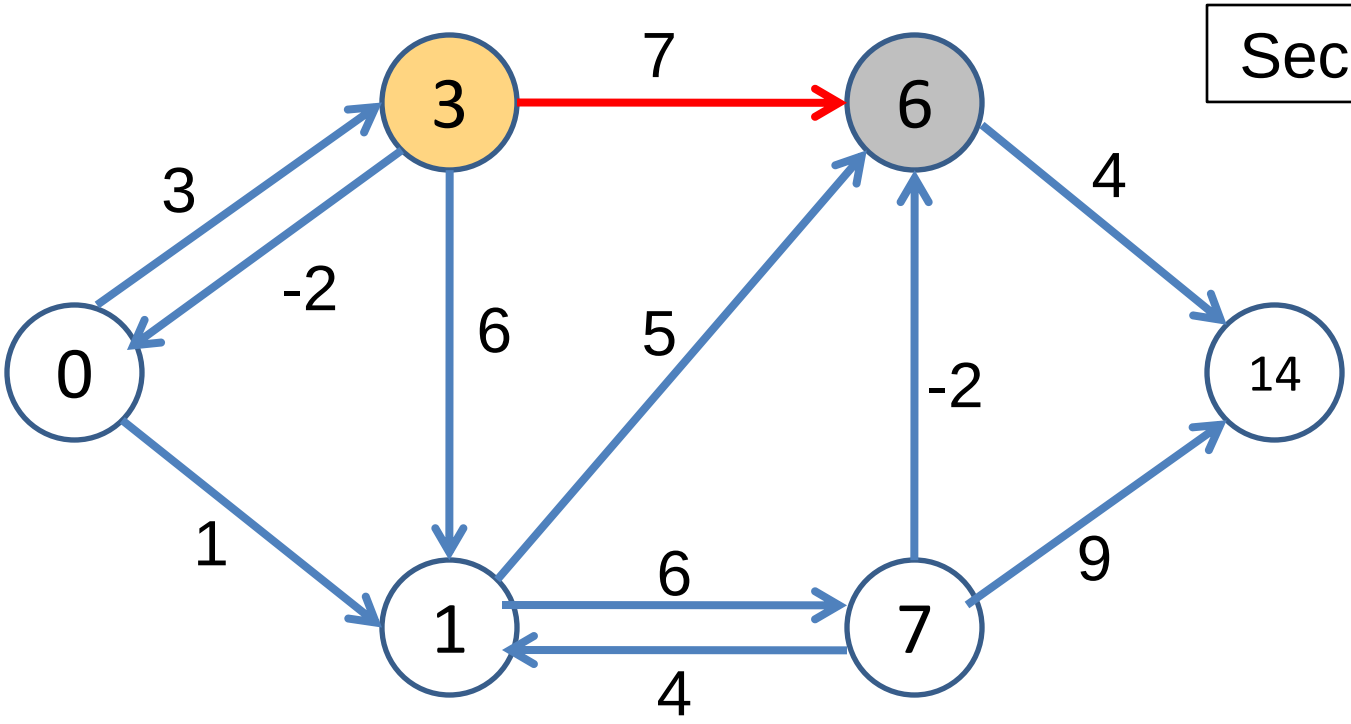
Bellman Ford



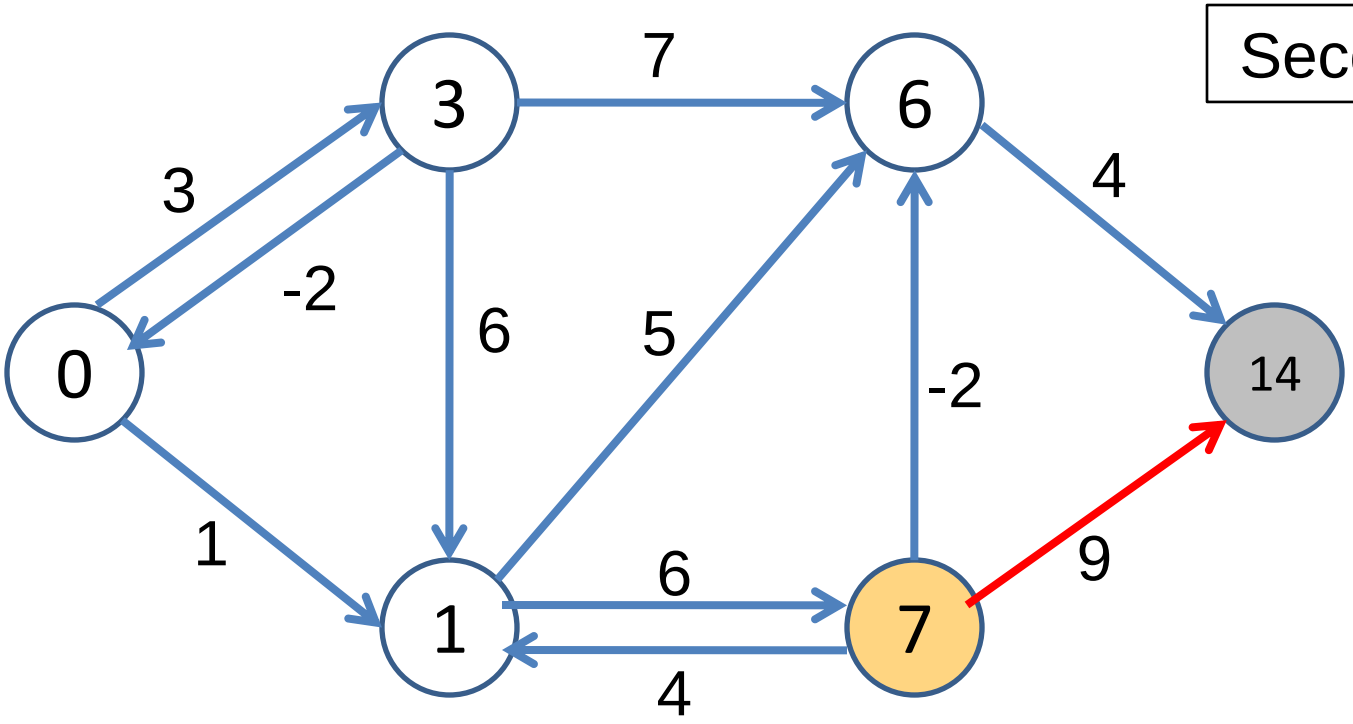
Bellman Ford



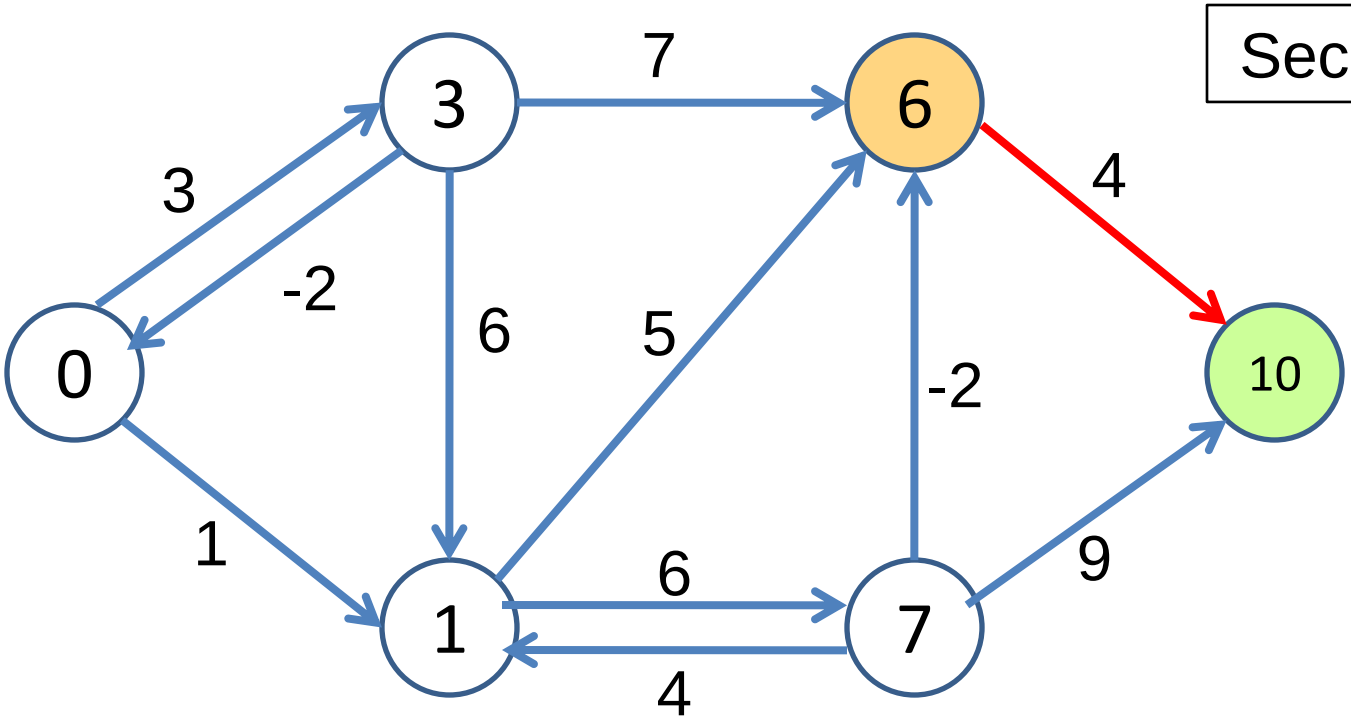
Bellman Ford



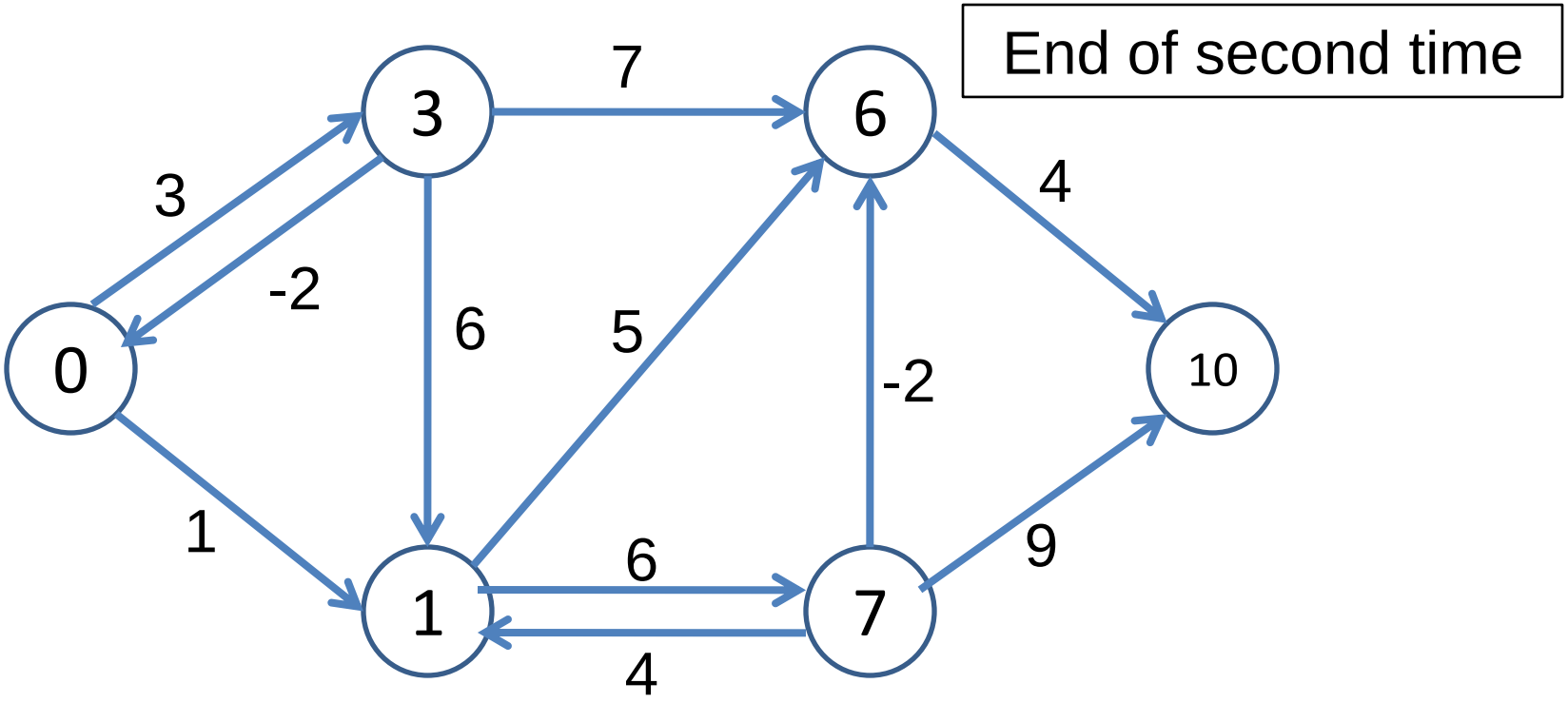
Bellman Ford



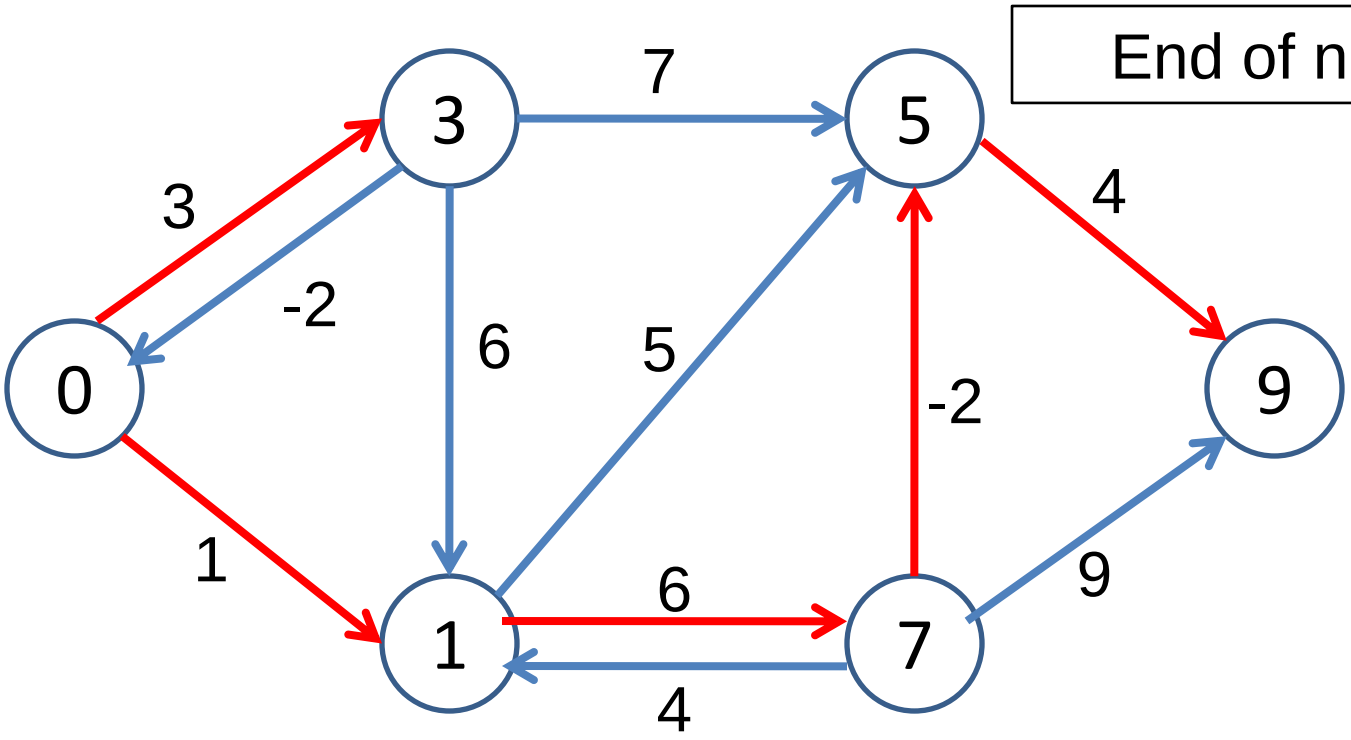
Bellman Ford



Bellman Ford



Bellman Ford



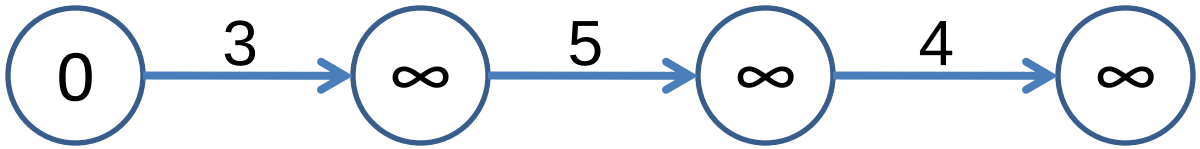
Bellman Ford

- Why n-1 time(s)?

Bellman Ford

- Why n-1 time(s)?

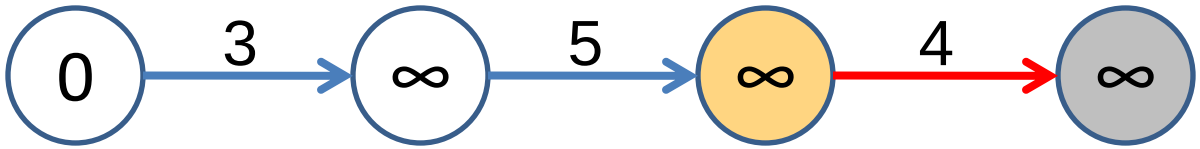
1st time



Bellman Ford

- Why n-1 time(s)?

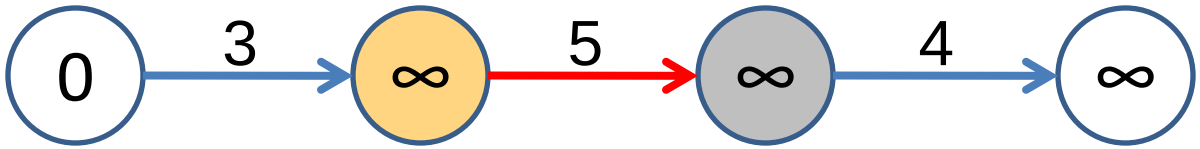
1st time



Bellman Ford

- Why n-1 time(s)?

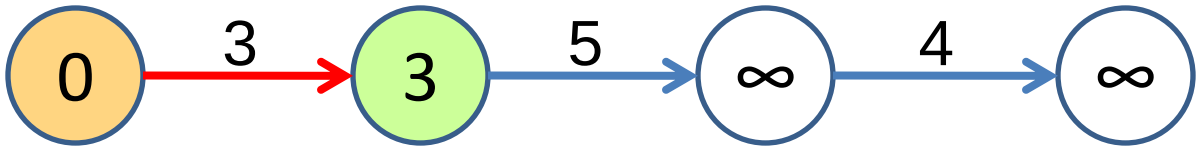
1st time



Bellman Ford

- Why n-1 time(s)?

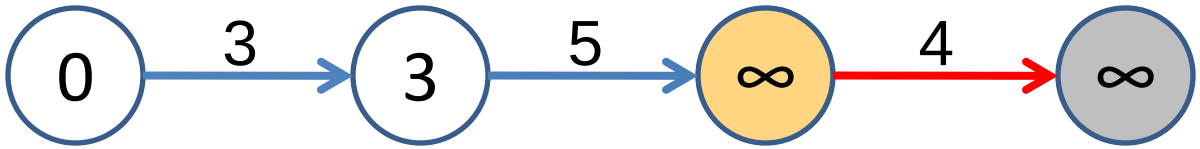
1st time



Bellman Ford

- Why n-1 time(s)?

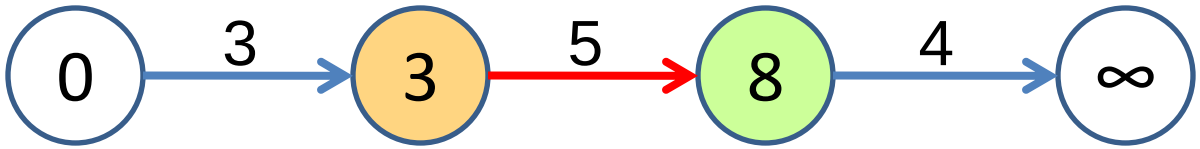
2nd time



Bellman Ford

- Why n-1 time(s)?

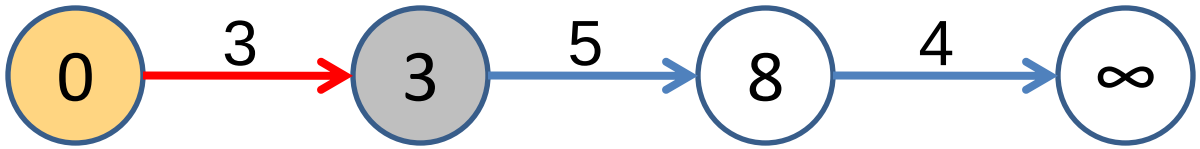
2nd time



Bellman Ford

- Why n-1 time(s)?

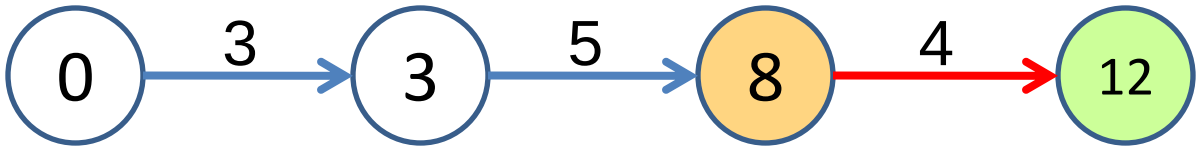
2nd time



Bellman Ford

- Why n-1 time(s)?

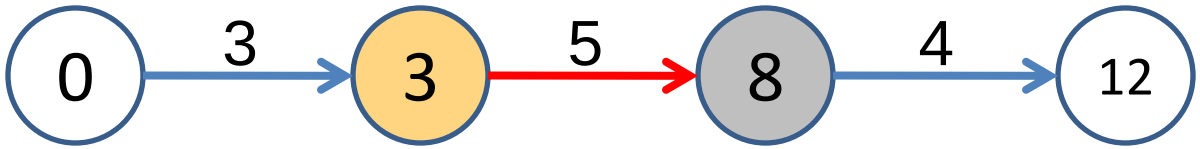
3rd time



Bellman Ford

- Why n-1 time(s)?

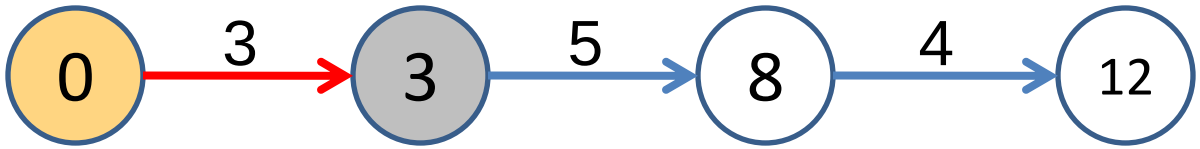
3rd time



Bellman Ford

- Why n-1 time(s)?

3rd time



Bellman Ford

- Pseudo code

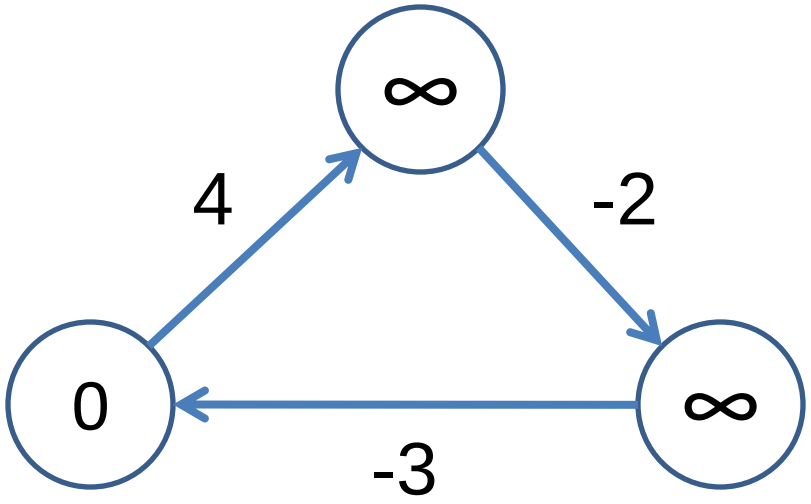
```
1  BellmanFord(){
2      // Initialize
3      dis[source]=0;
4      dis[i]=INF, for all i!=source
5
6      // n-1 times
7      for(i=0;i<n-1;i++)
8          for each edge w(u,v) in G
9              Relax(u,v,w);
10 }
```

Practice1

- POJ 2387 - Til the Cows Come Home

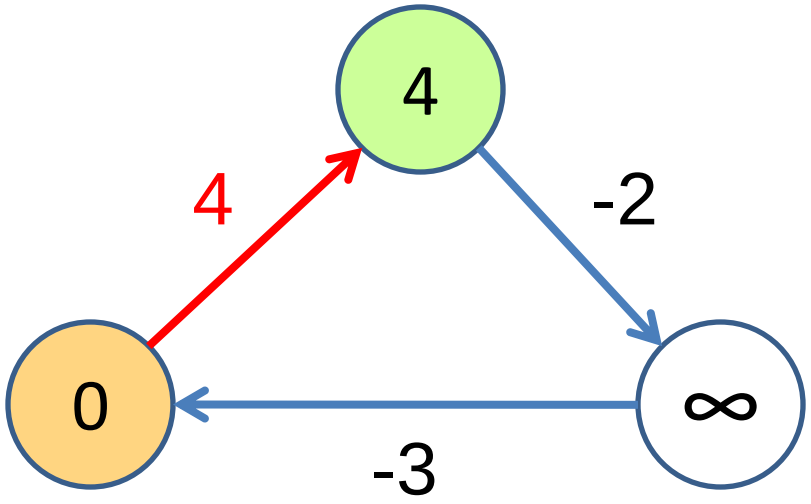
Bellman Ford

- Negative Cycle?



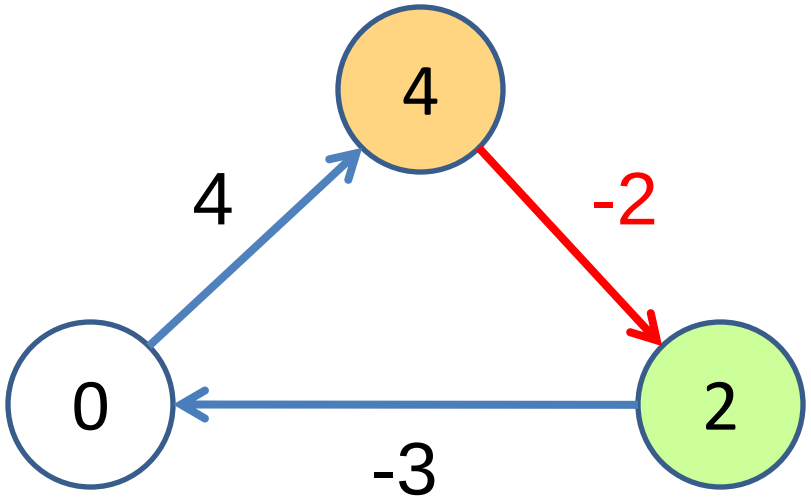
Bellman Ford

- Negative Cycle?



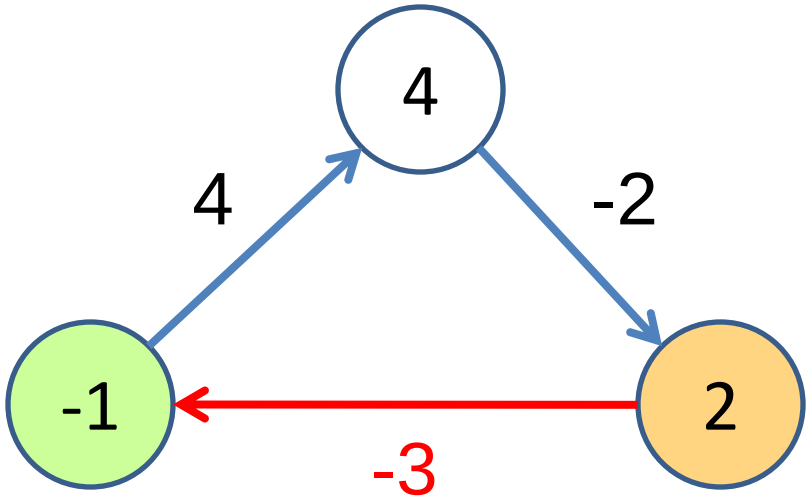
Bellman Ford

- Negative Cycle?



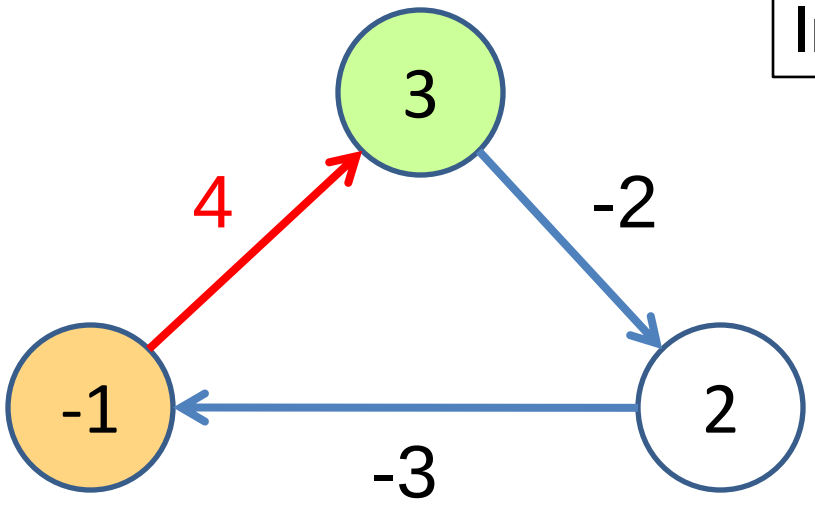
Bellman Ford

- Negative Cycle?



Bellman Ford

- Negative Cycle?



Infinite loop….

Bellman Ford

- Relax 1 more time after n-1 times
  - If relax successfully, negative cycle exists.

- 若找到負環，是位於整張圖上的某處
  - Source不見得可以走的到該負環

Bellman Ford

- Pseudo code

```
1  BellmanFord(){
2      // Initialize
3      dis[source]=0;
4      dis[i]=INF, for all i!=source
5
6      // n-1 times
7      for(i=0;i<n-1;i++)
8          for each edge w(u,v) in G
9              Relax(u,v,w);
10
11     // Negative Cycle
12     for each edge w(u,v) in G
13         if(dis[u]+w(u,v)<dis[v])
14             return true;
15     return false;
16 }
```

Practice2

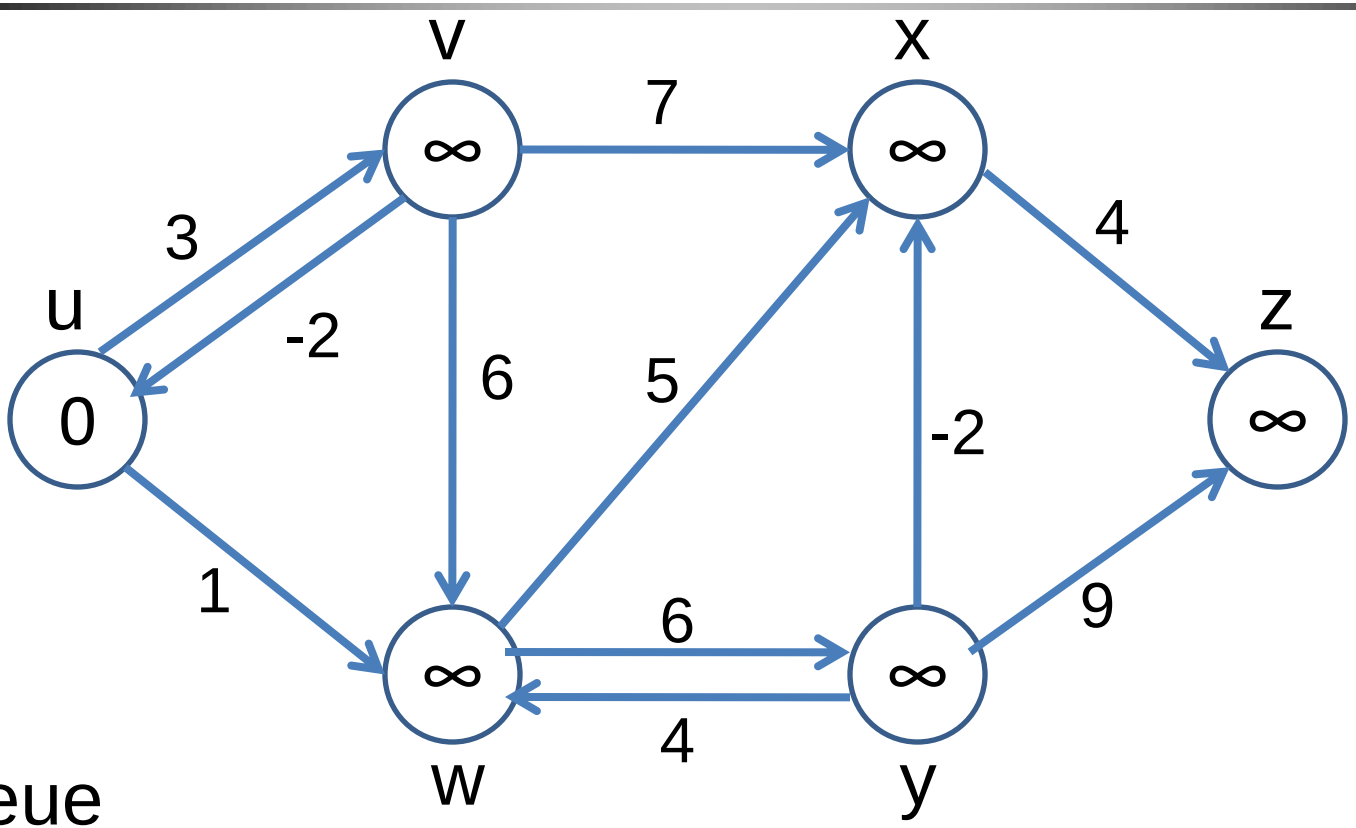
- POJ 3259 - Wormhole

SPFA

SPFA

- Shortest Path Faster Algorithm
- In Bellman Ford, relax n-1 times
  - Do we really need n-1 times…?
  - Only relax the one whose cost changed!
- Queue

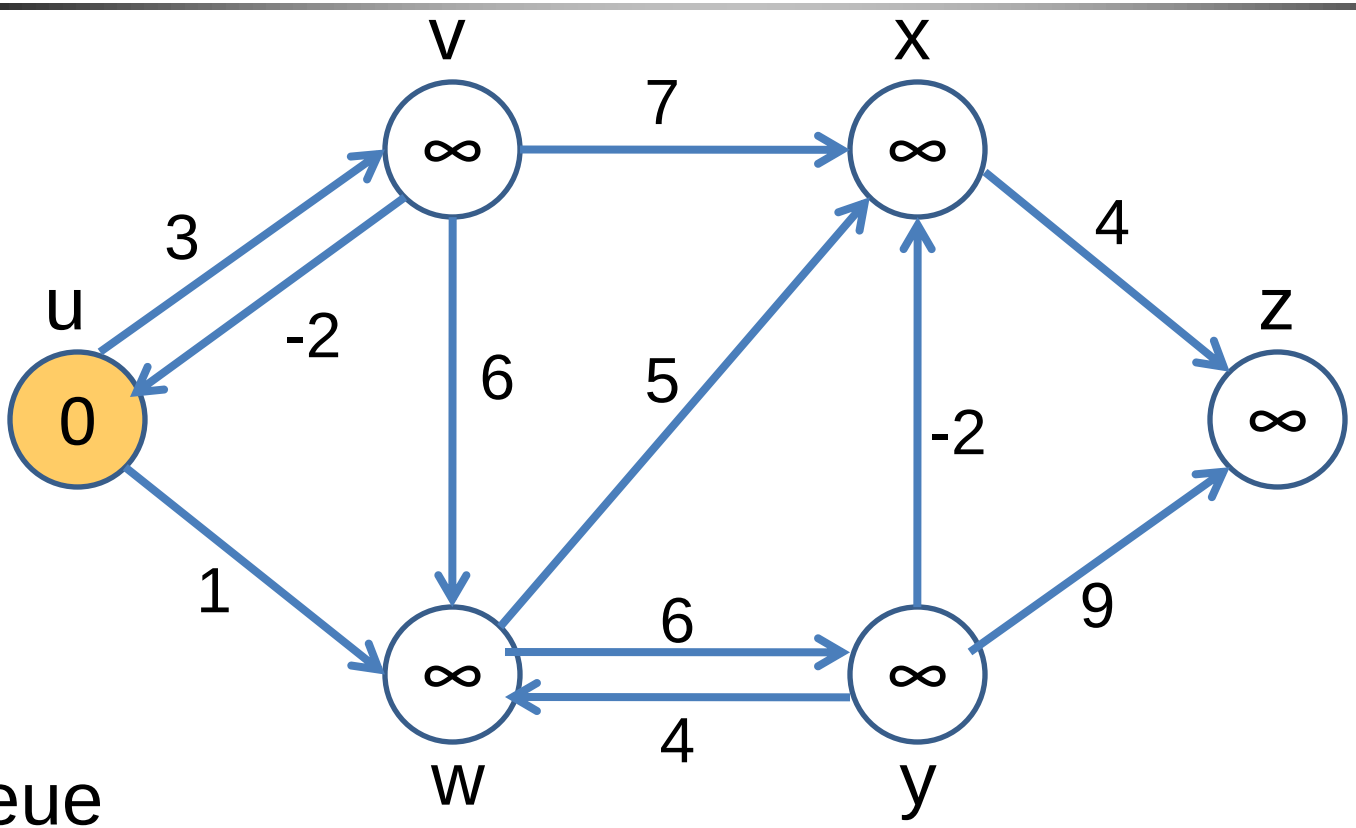
SPFA



Queue

Now:

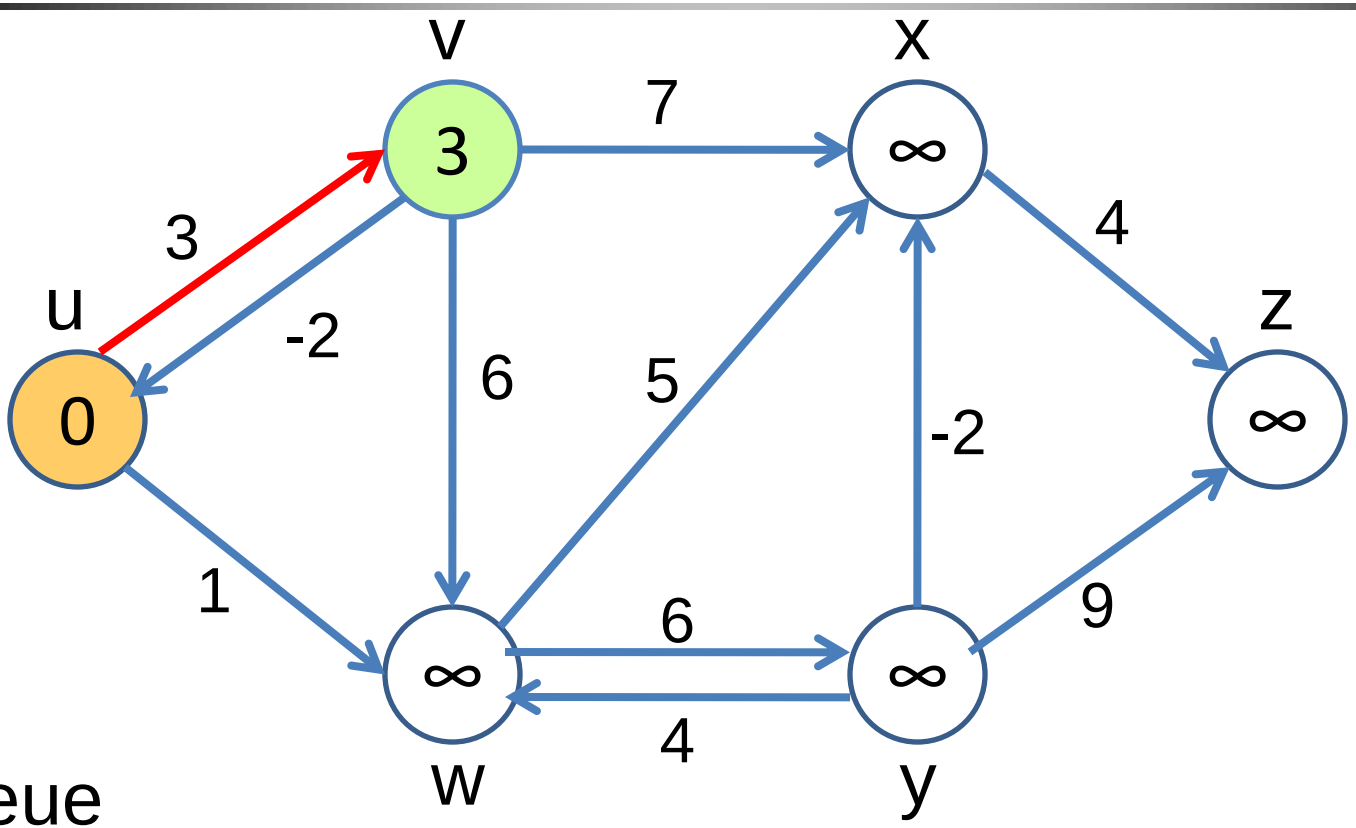
SPFA



Queue

Now: u

SPFA

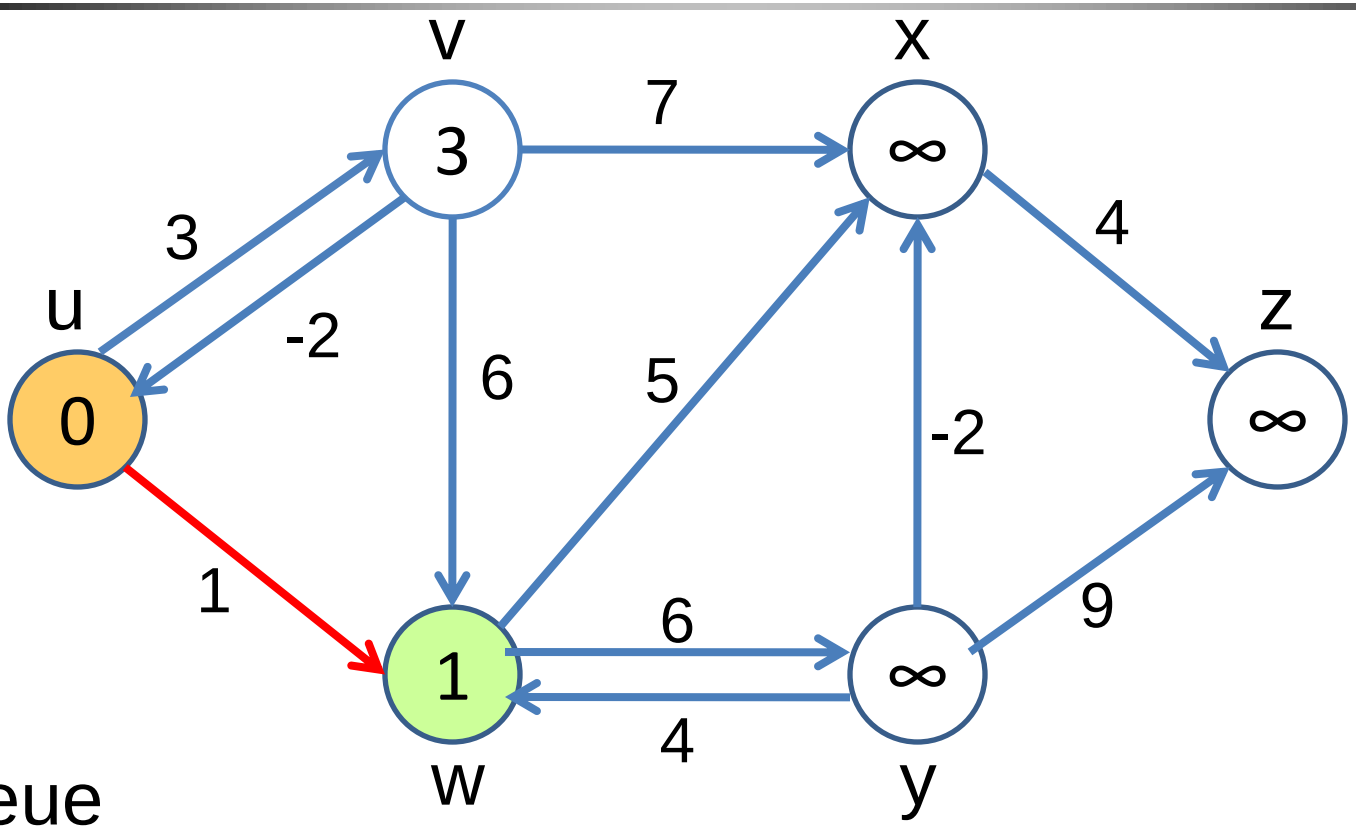


Queue

v	

Now: u v has changed

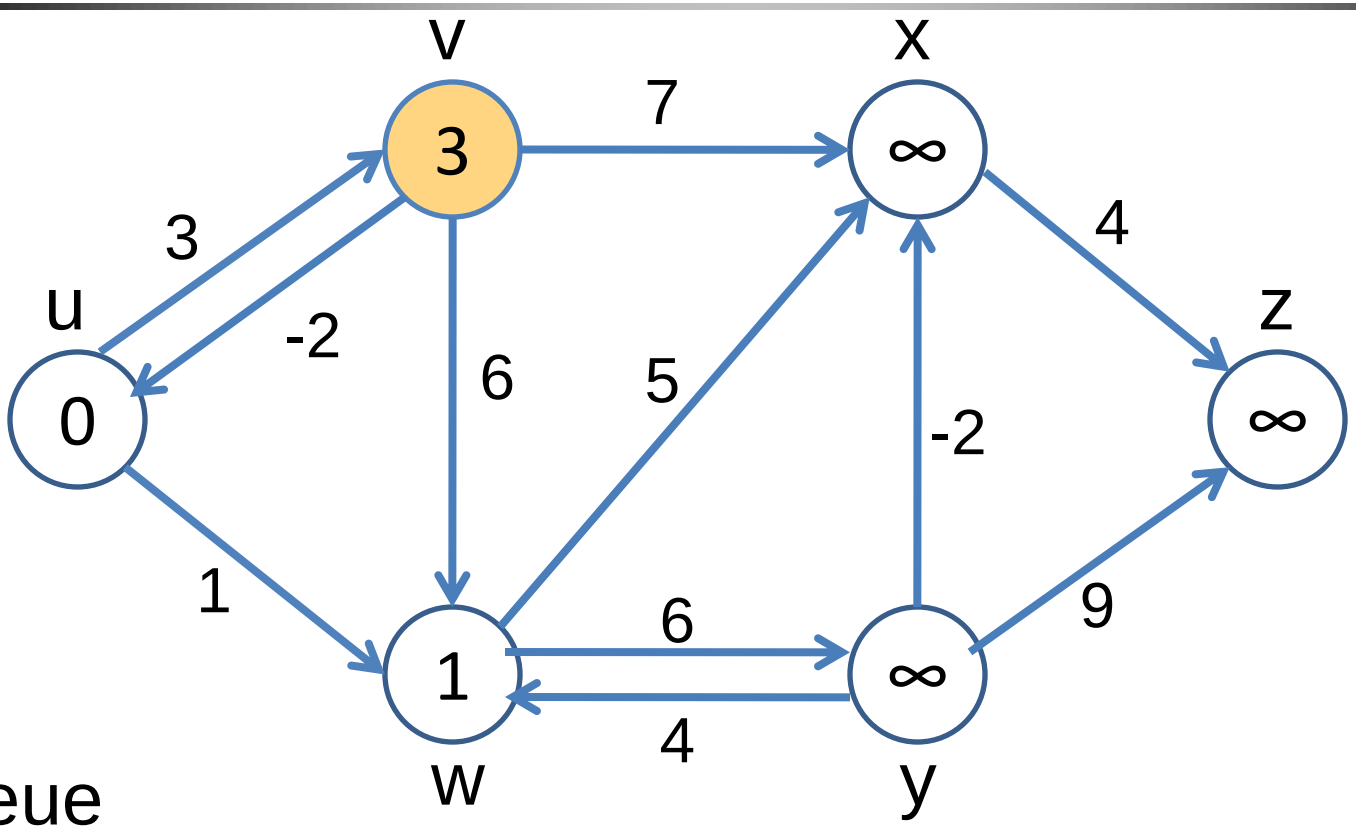
SPFA



Queue

Now: u w has changed

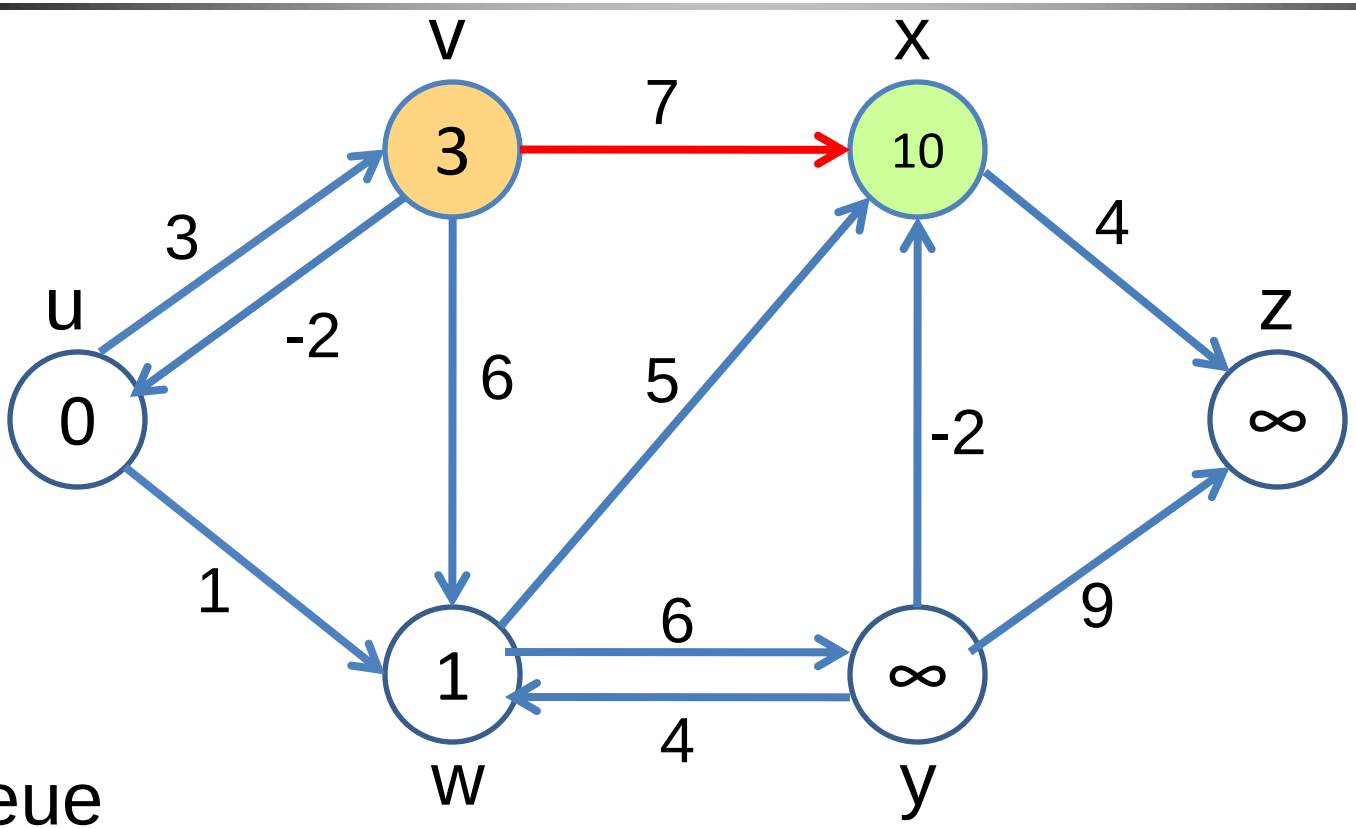
SPFA



Queue

Now: v

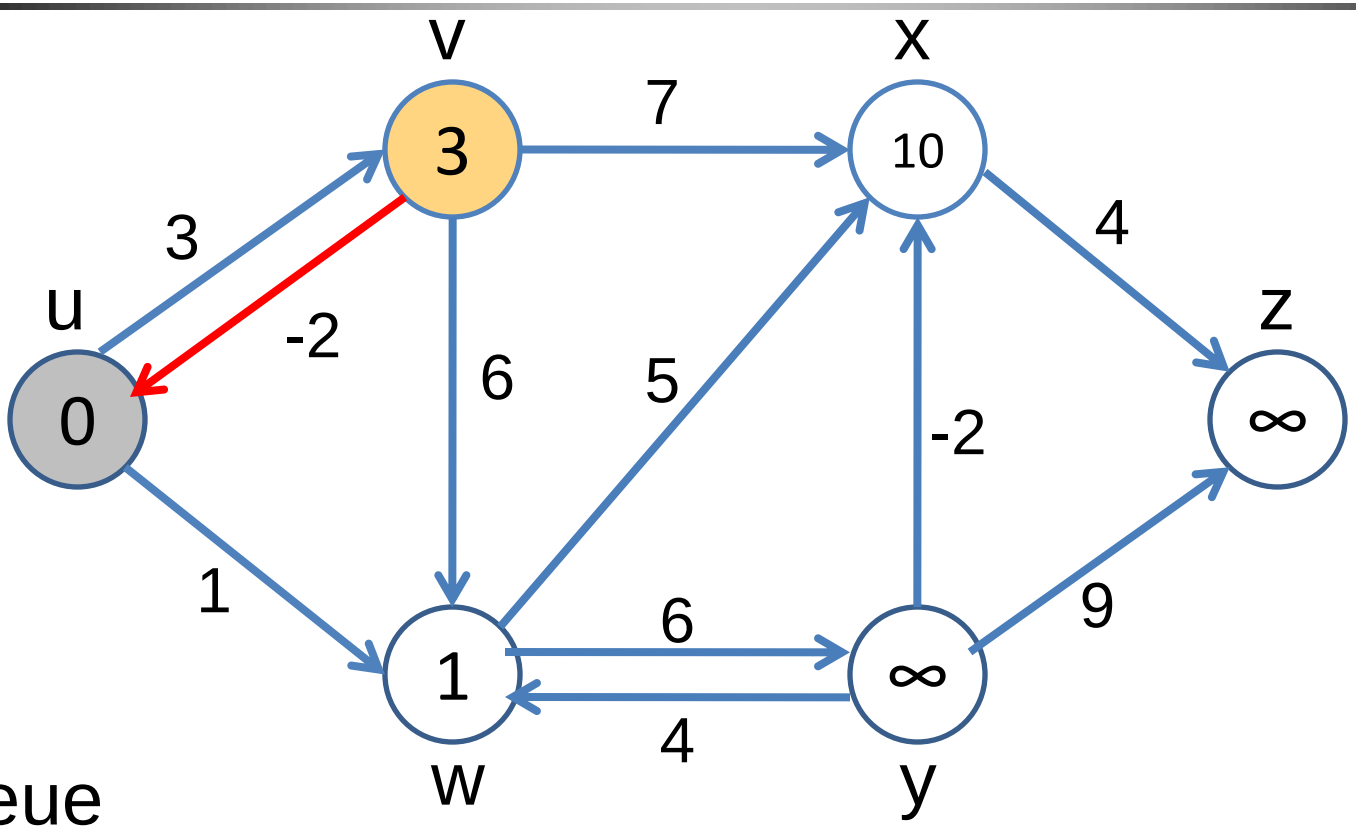
SPFA



Queue

Now: v x has changed

SPFA

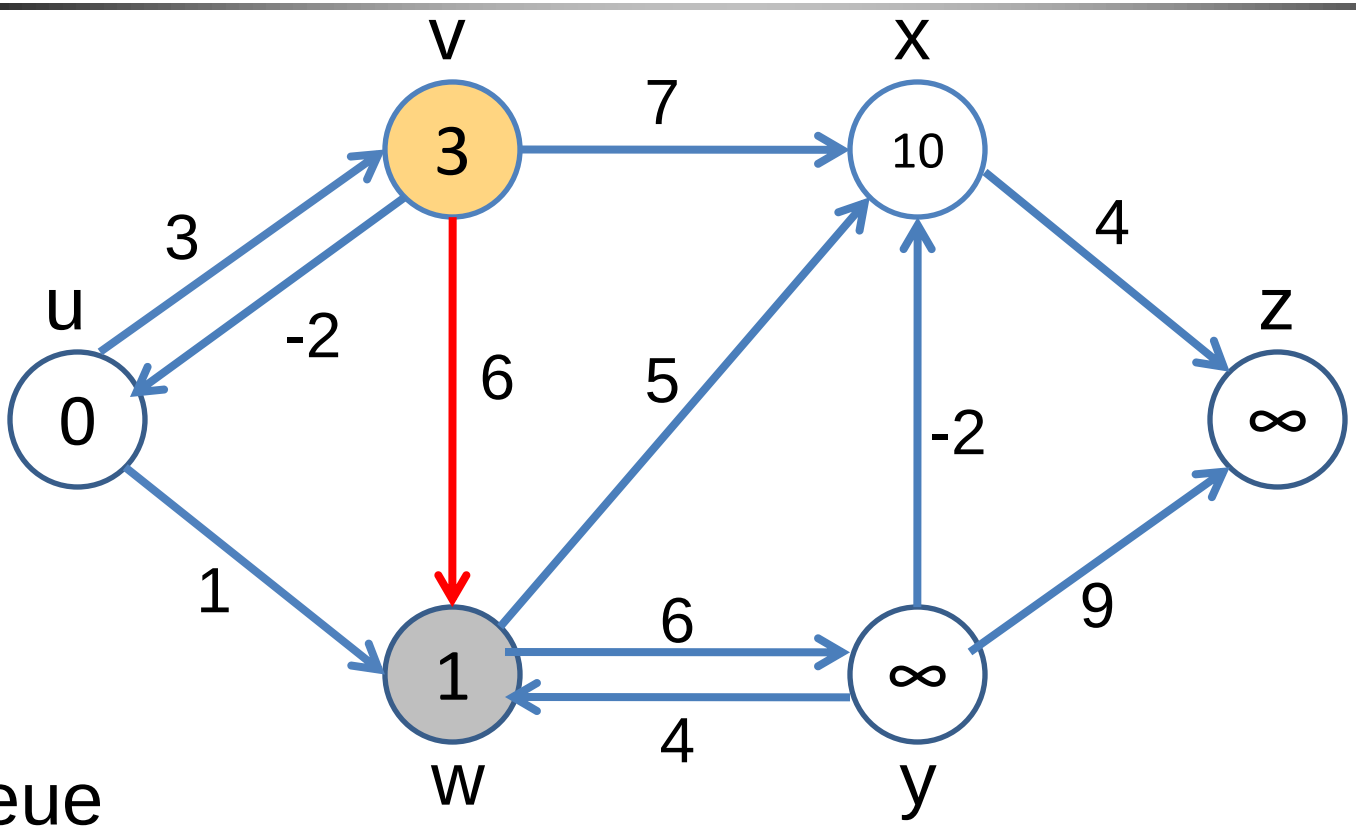


Queue

Now: v

u remains unchanged

SPFA

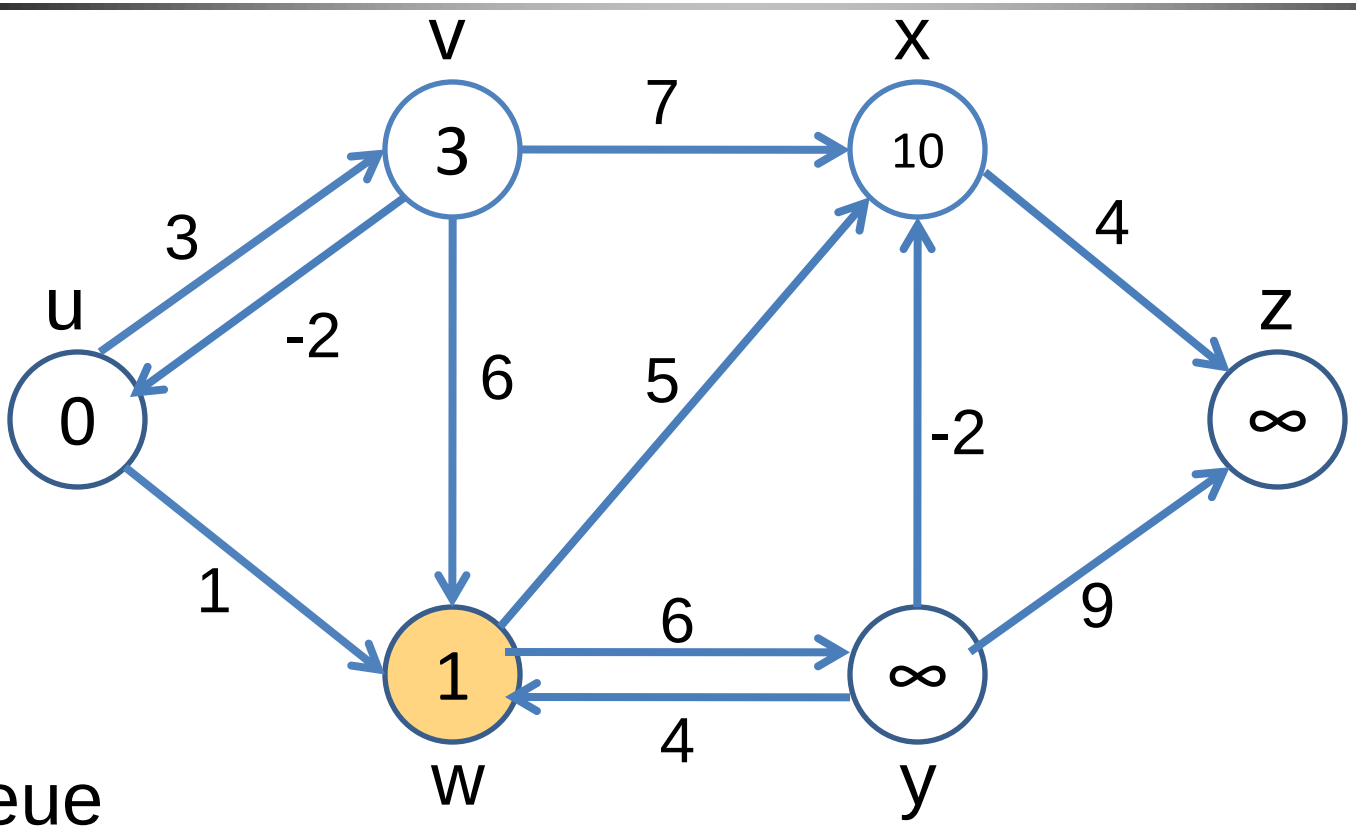


Queue

Now: v

w remains unchanged

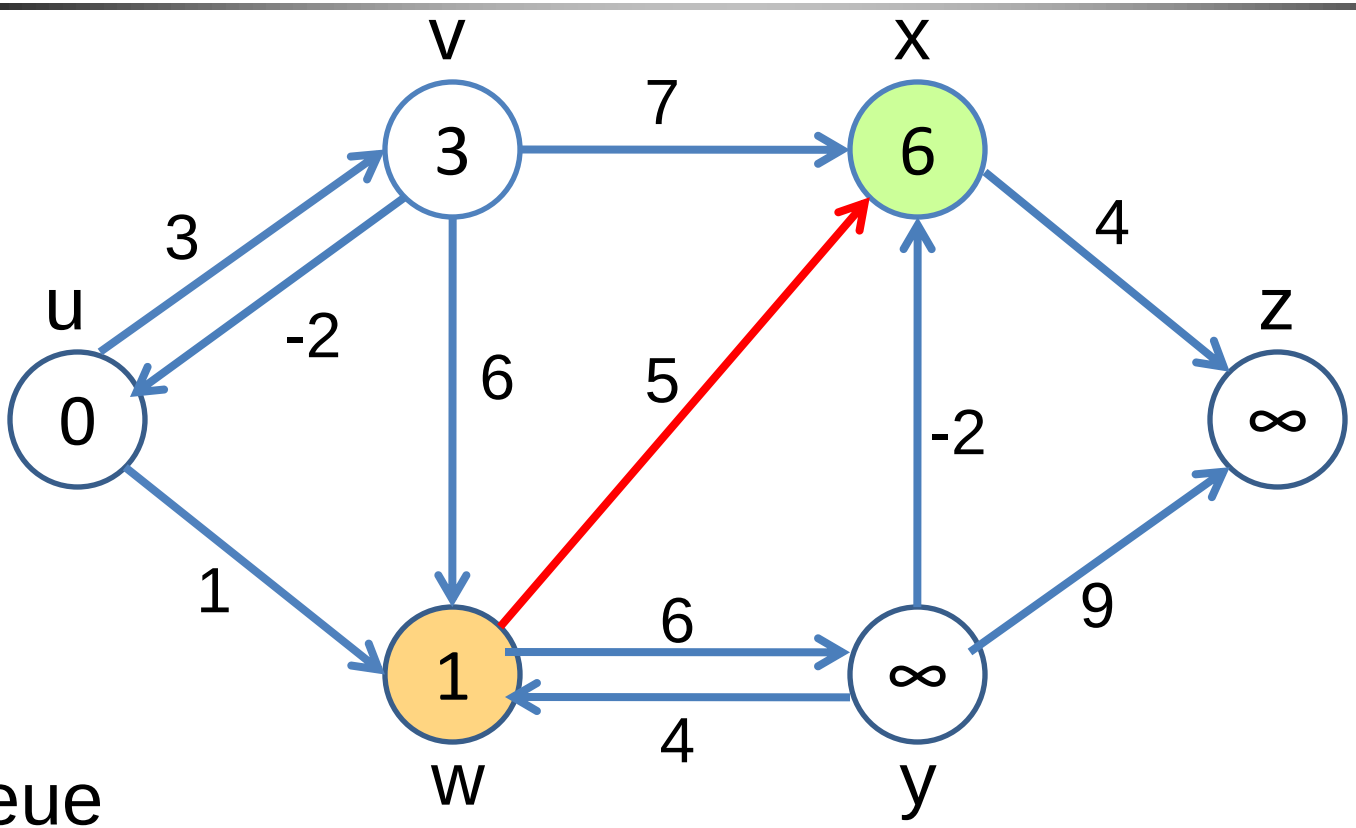
SPFA



Queue

Now: w

SPFA

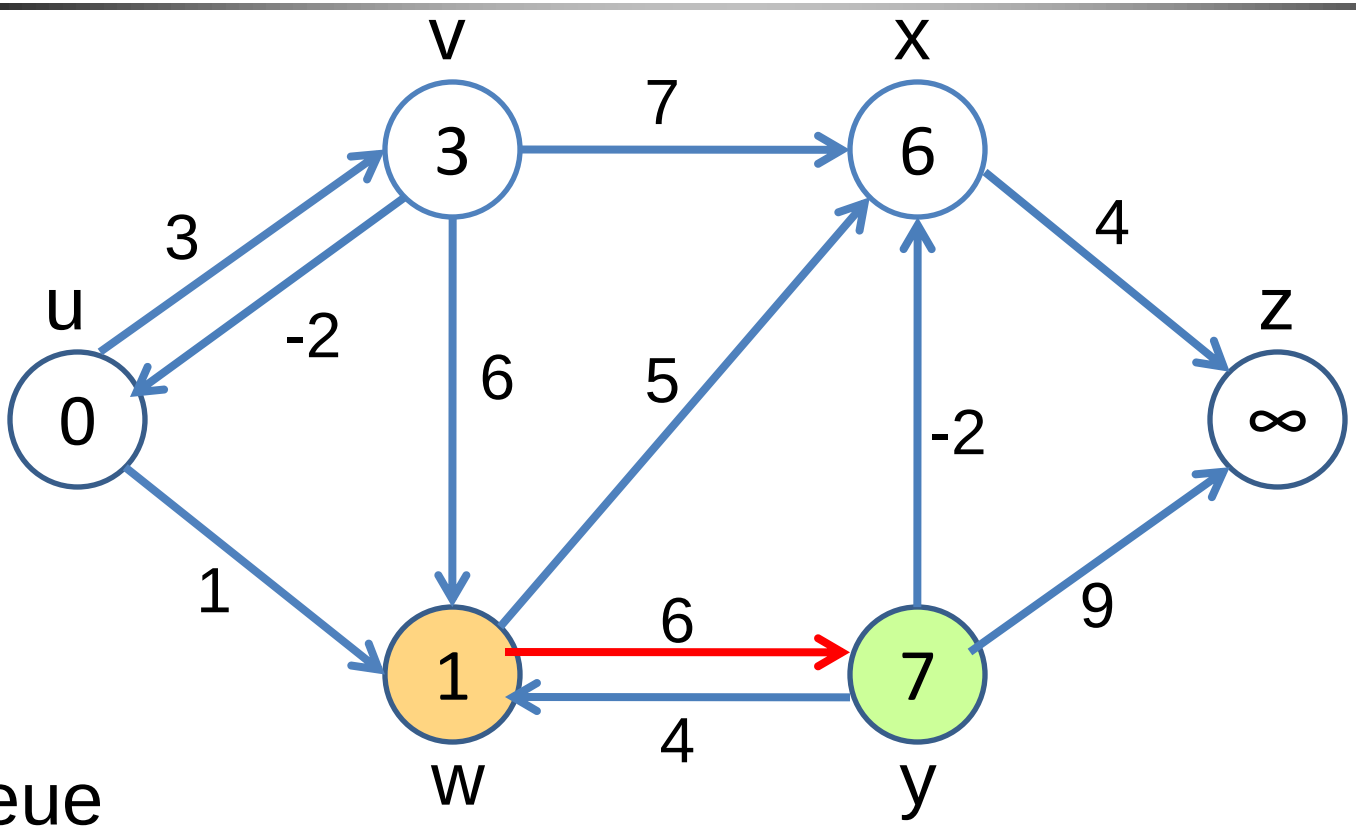


Queue

x	already in queue

Now: w x has changed

SPFA

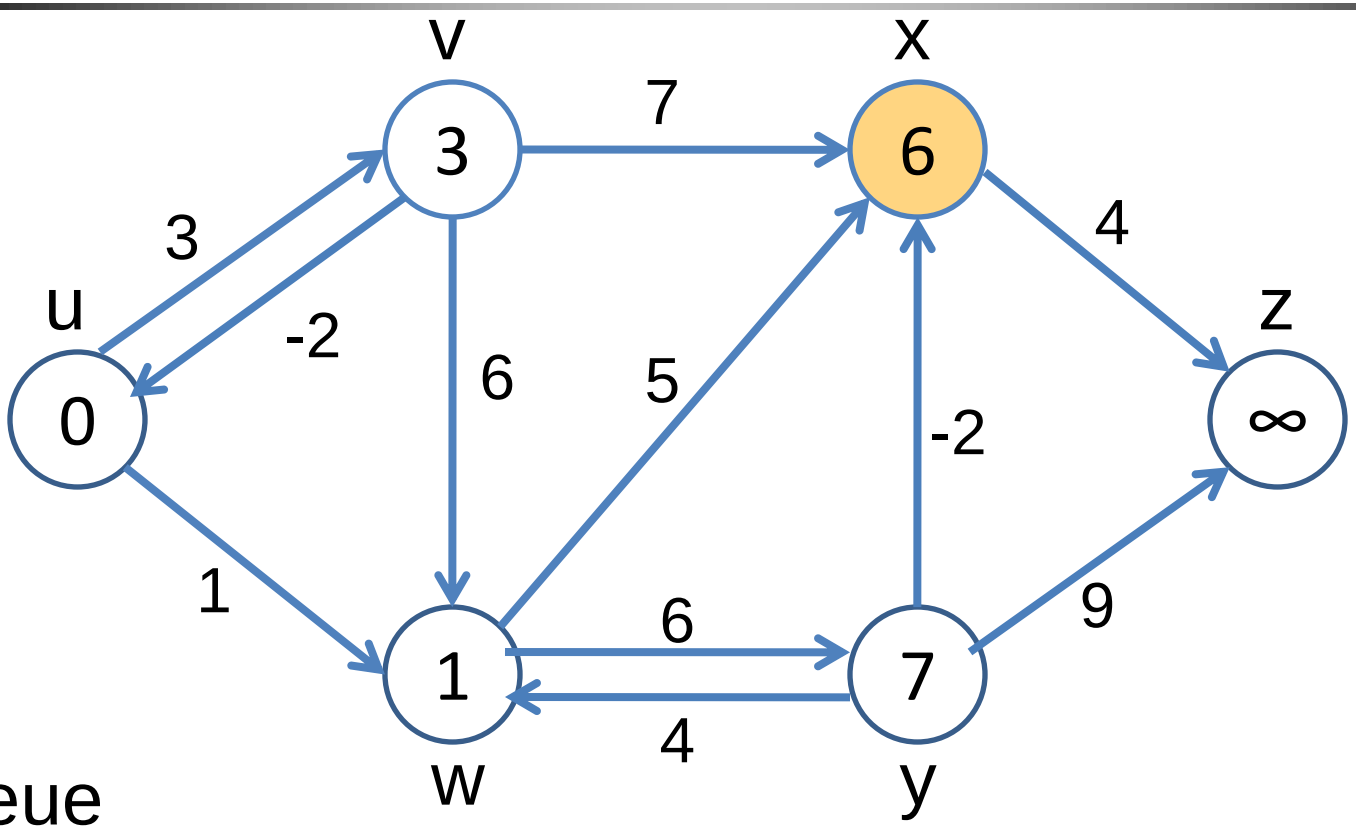


Queue

x	y	

Now: w y has changed

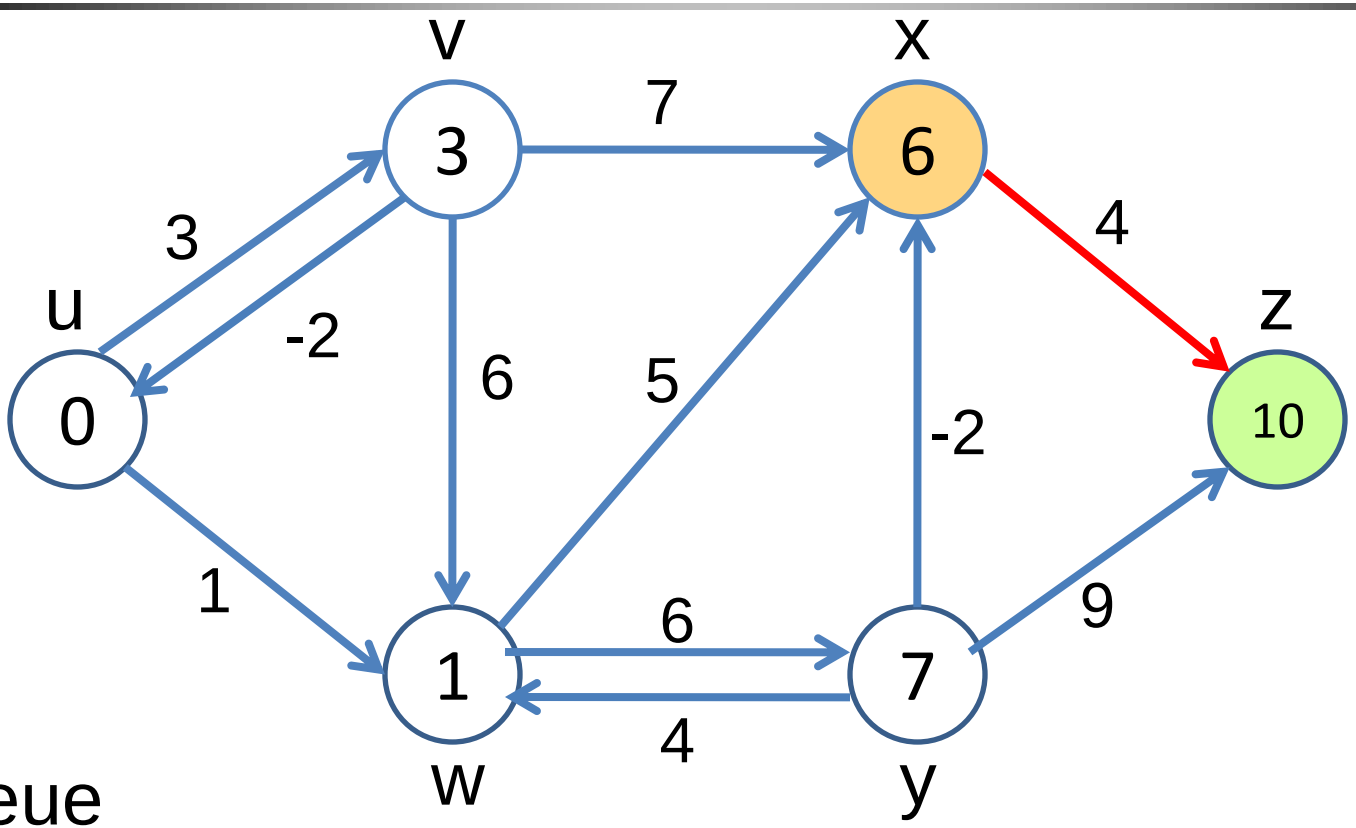
SPFA



Queue

Now: x

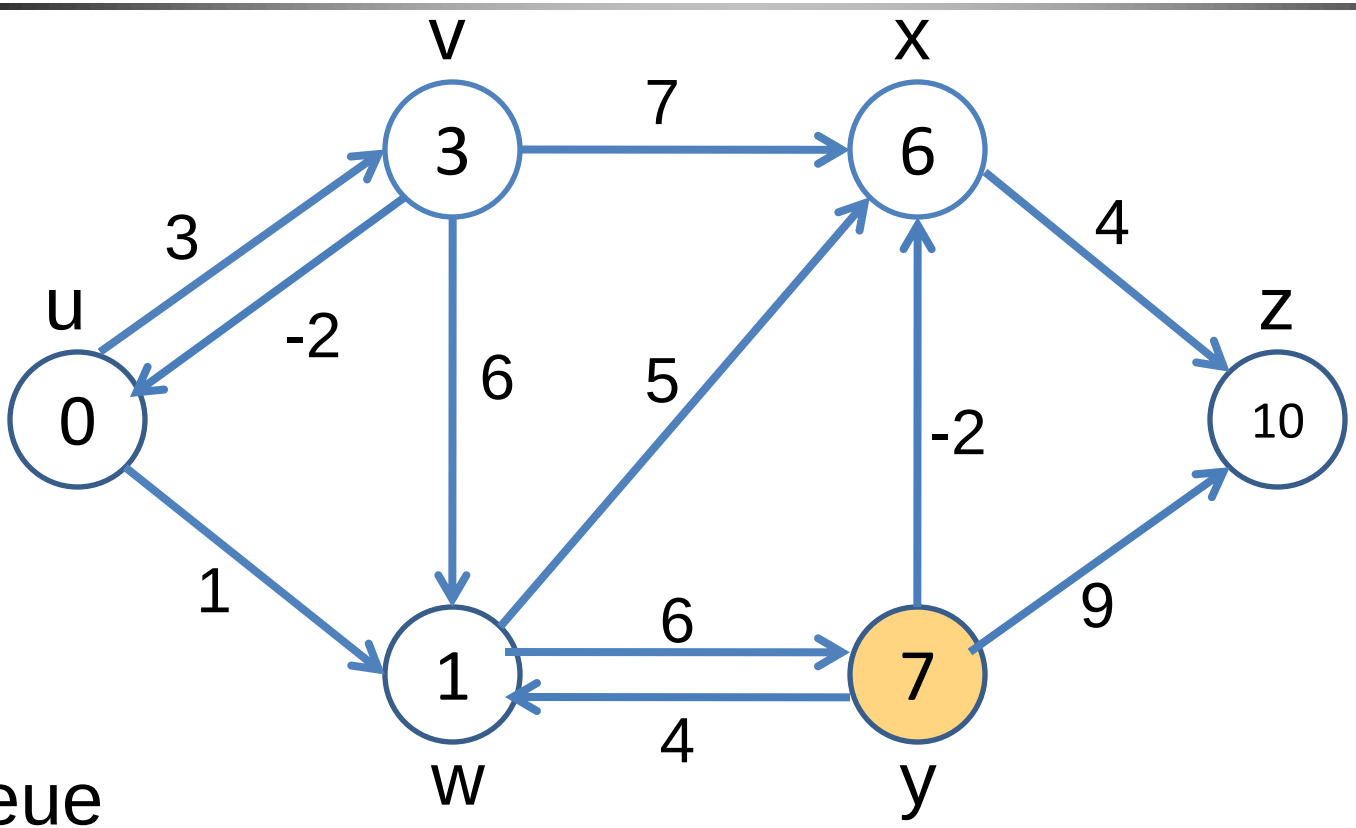
SPFA



Queue

Now: x z has changed

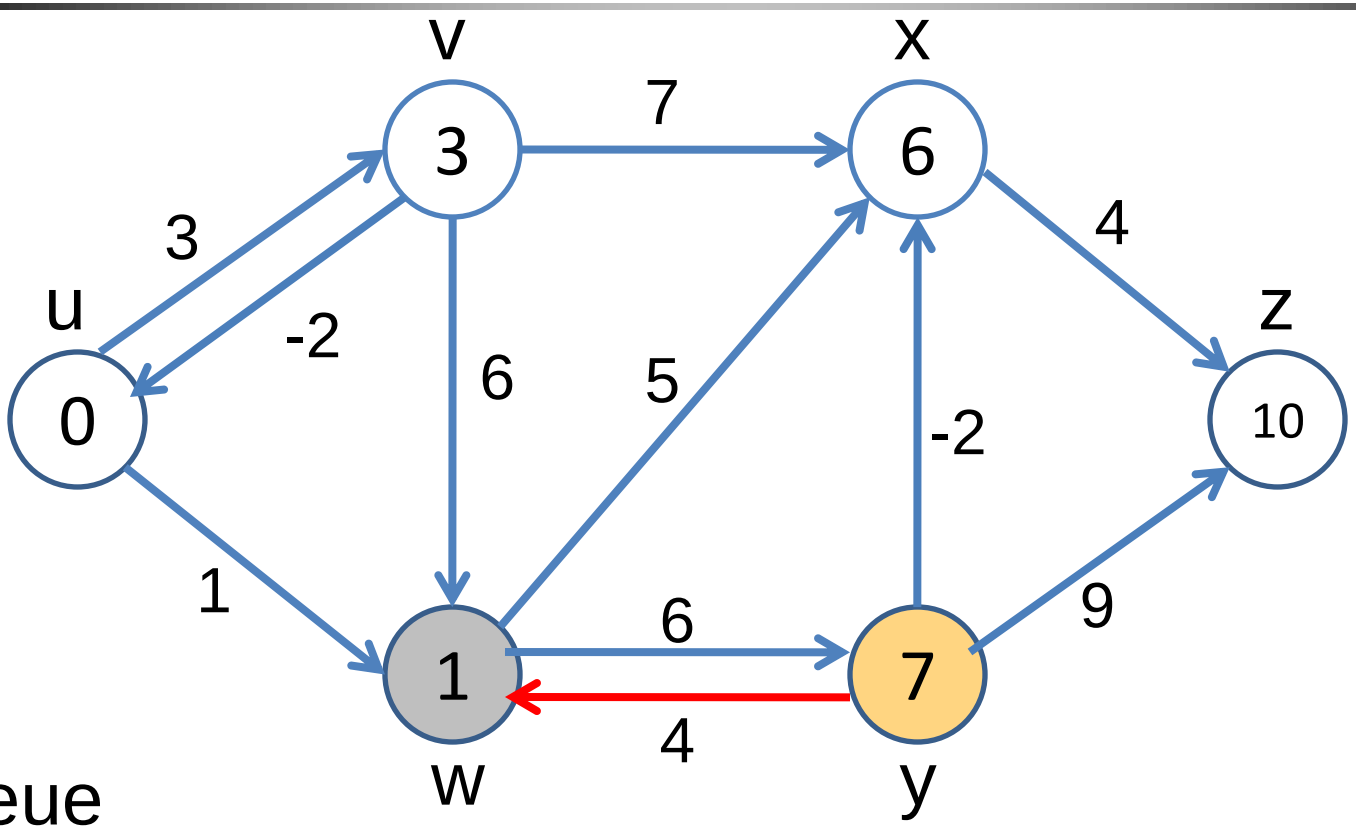
SPFA



Queue

Now: y

SPFA

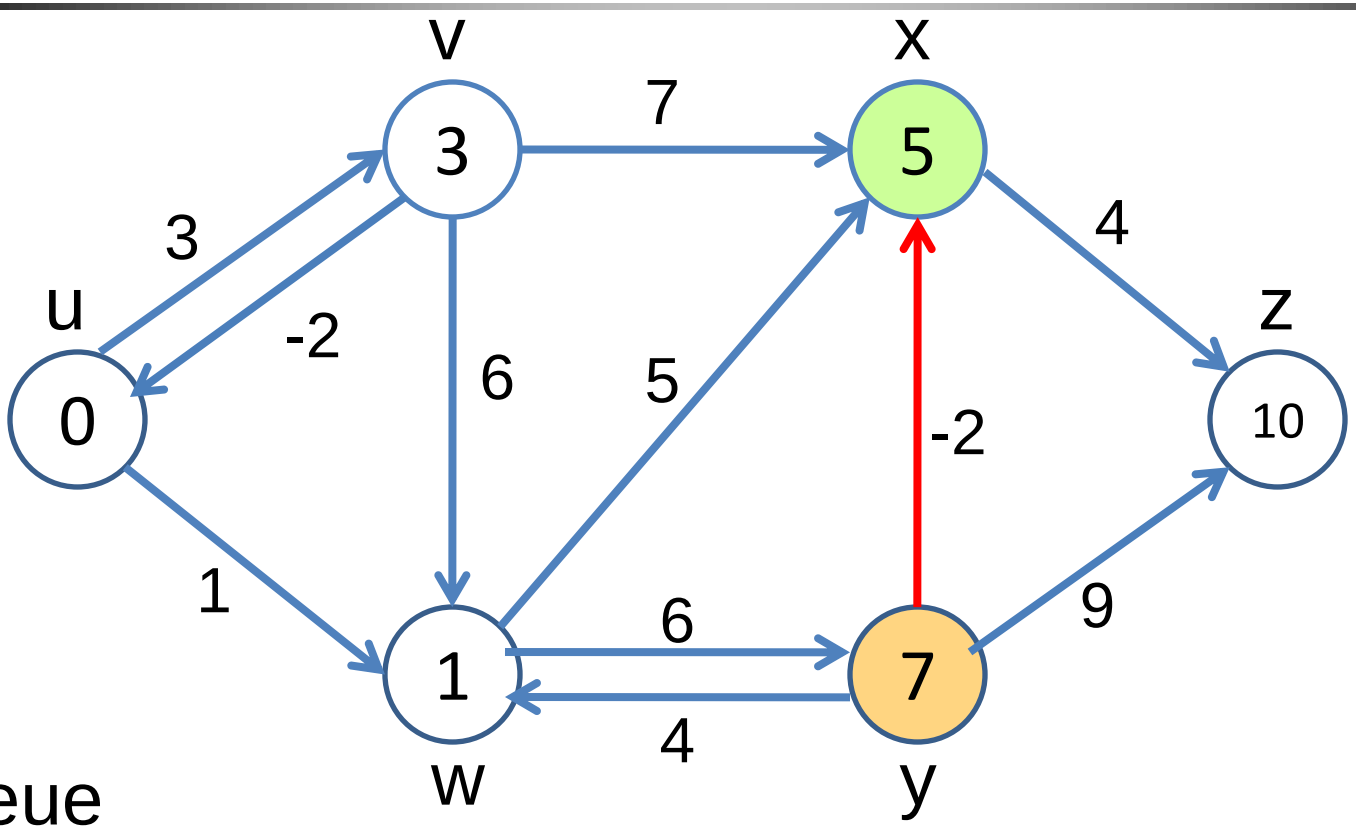


Queue

Now: y

w remains unchanged

SPFA

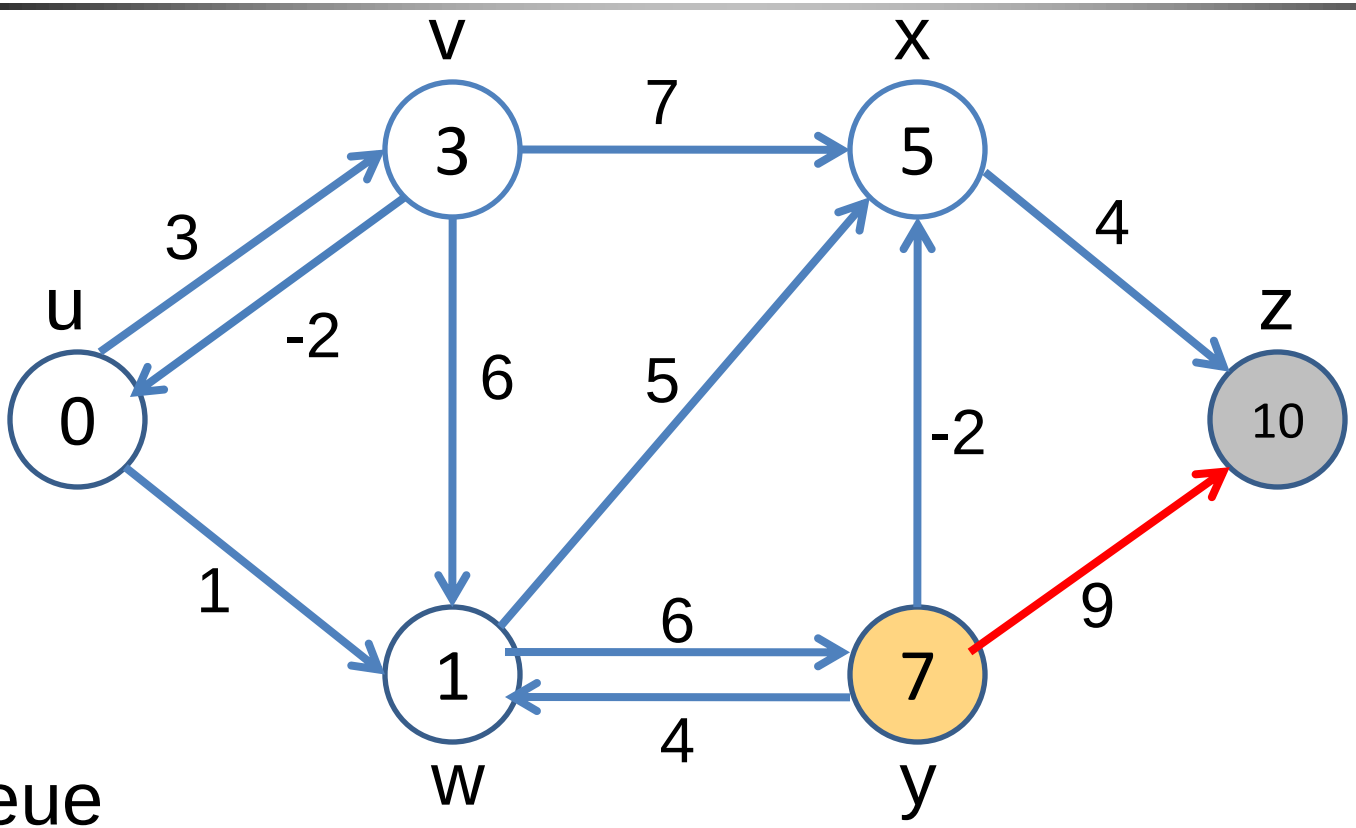


Queue

Now: y

x has changed

SPFA



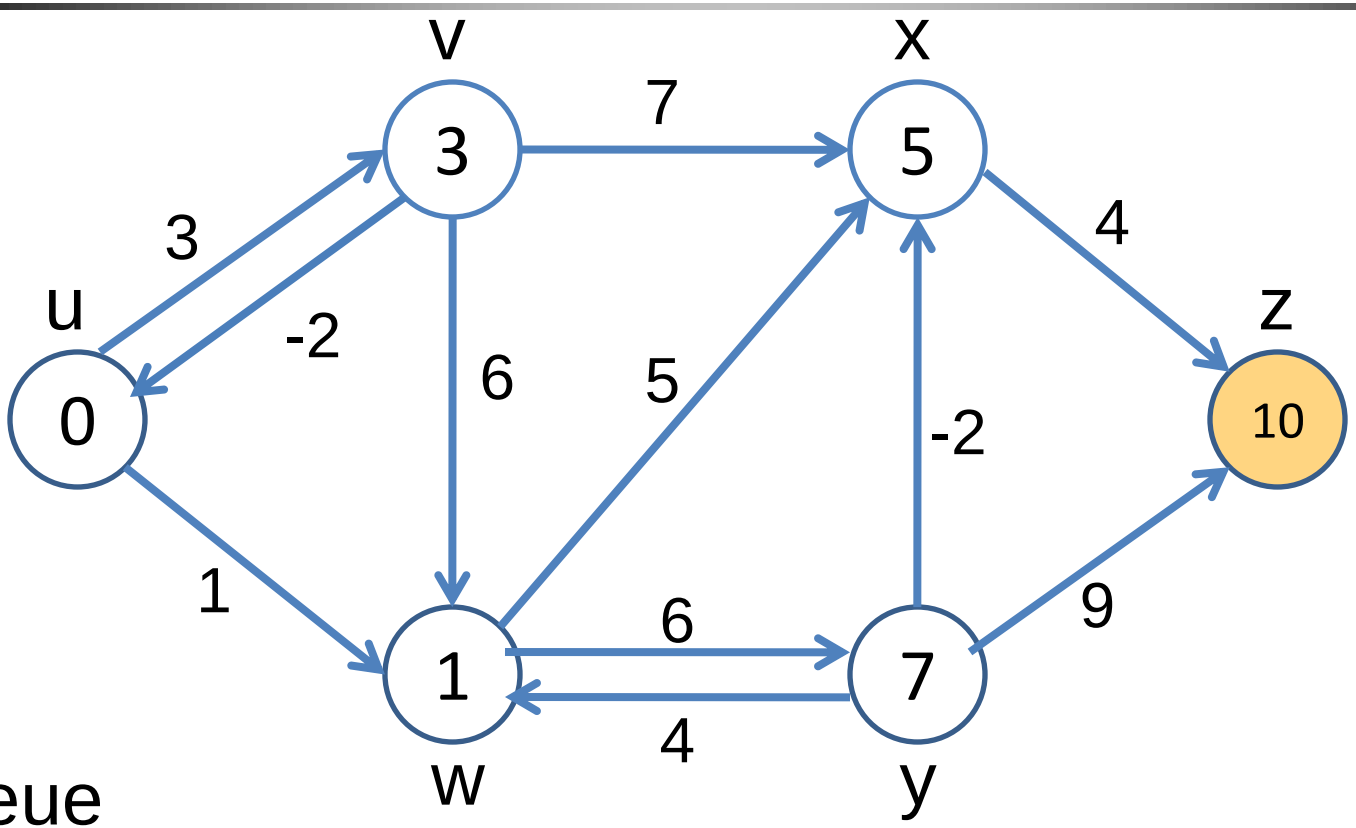
Queue

z	x	

Now: y

z remains unchanged

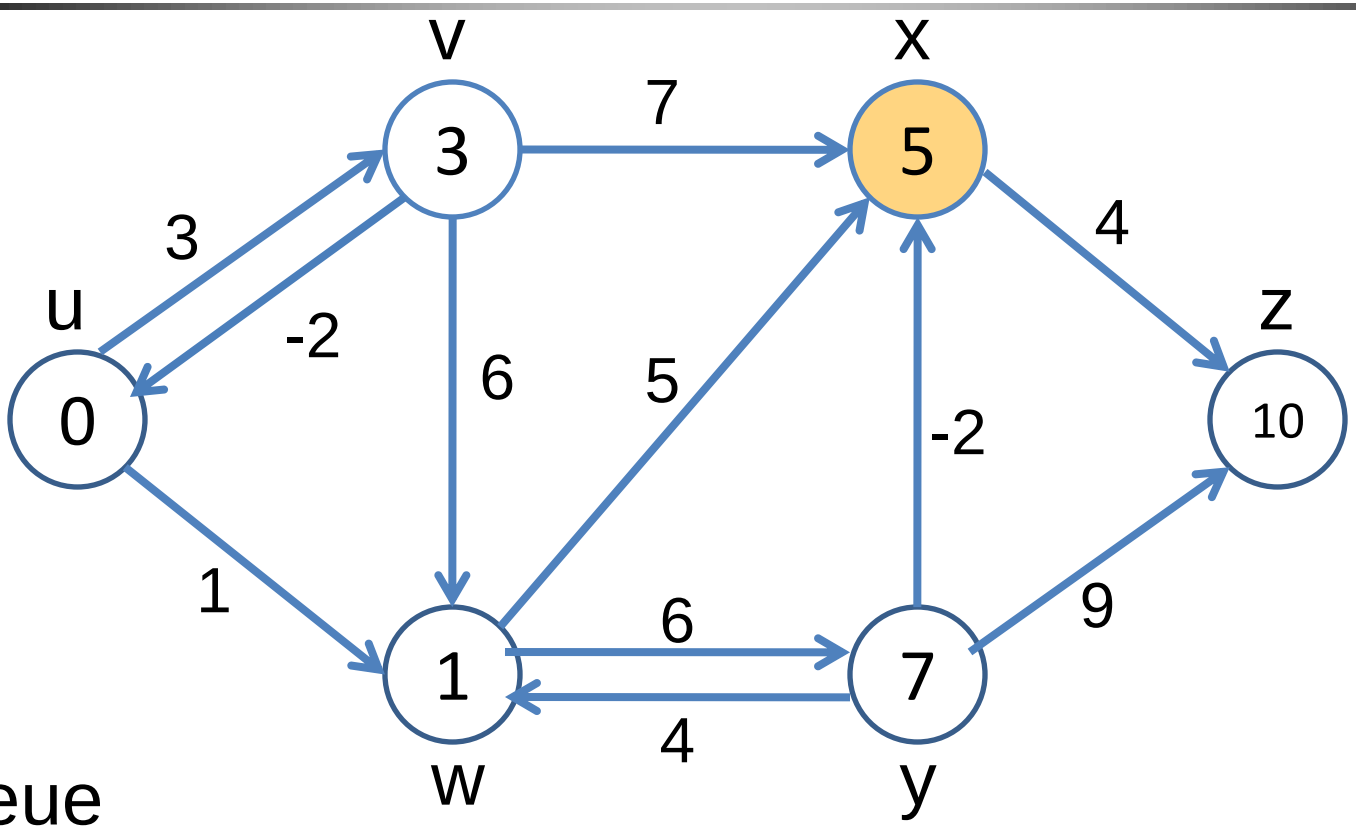
SPFA



Queue

Now: z

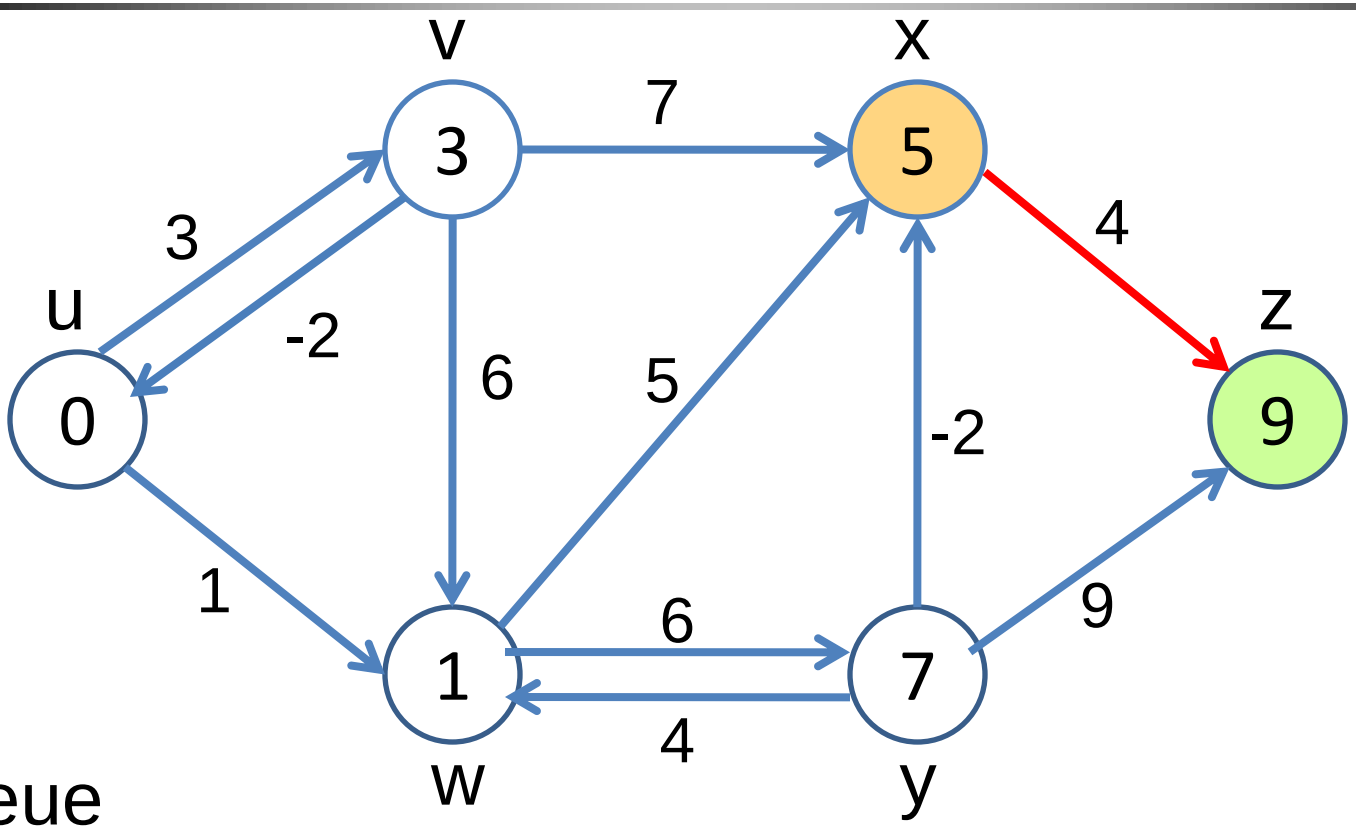
SPFA



Queue

Now: x

SPFA

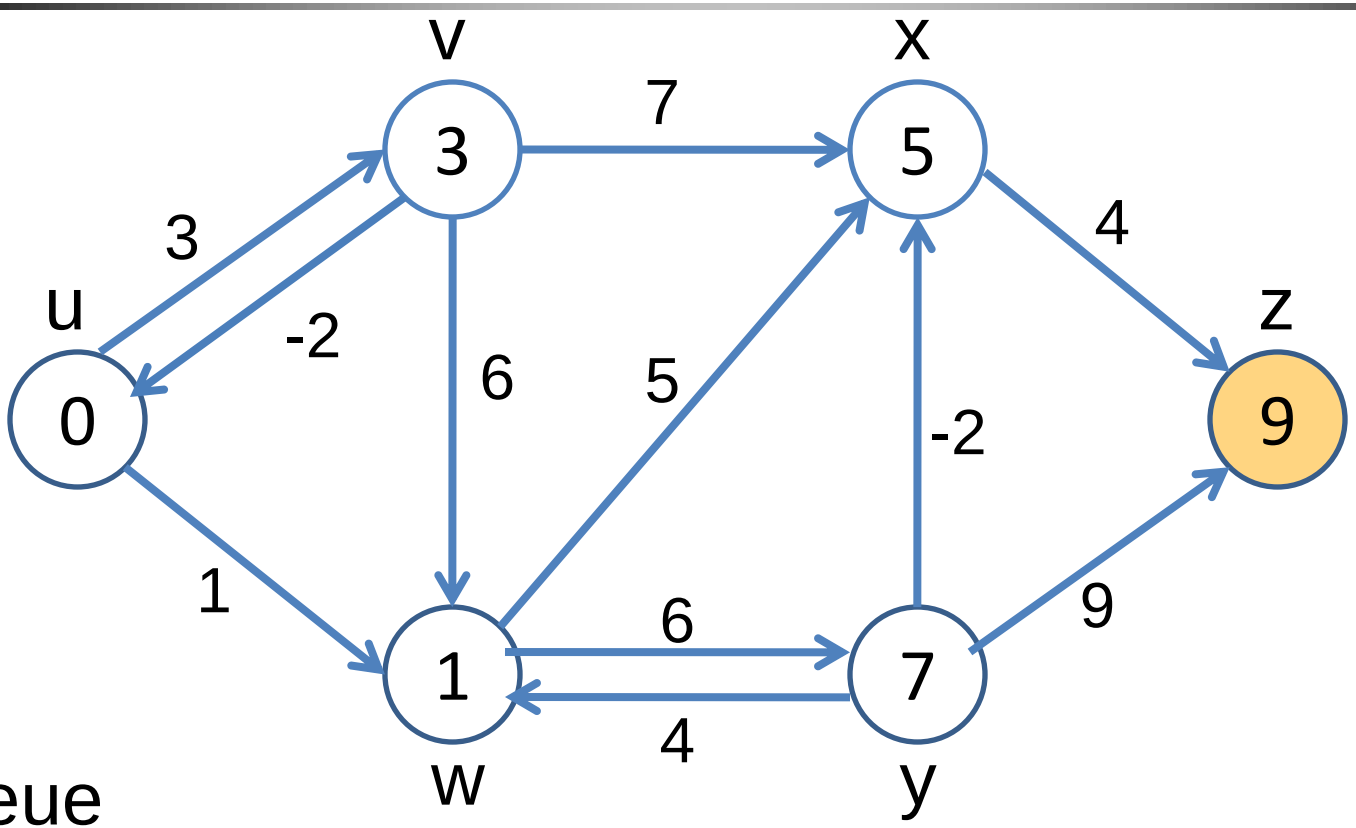


Queue

z	

Now: x z has changed

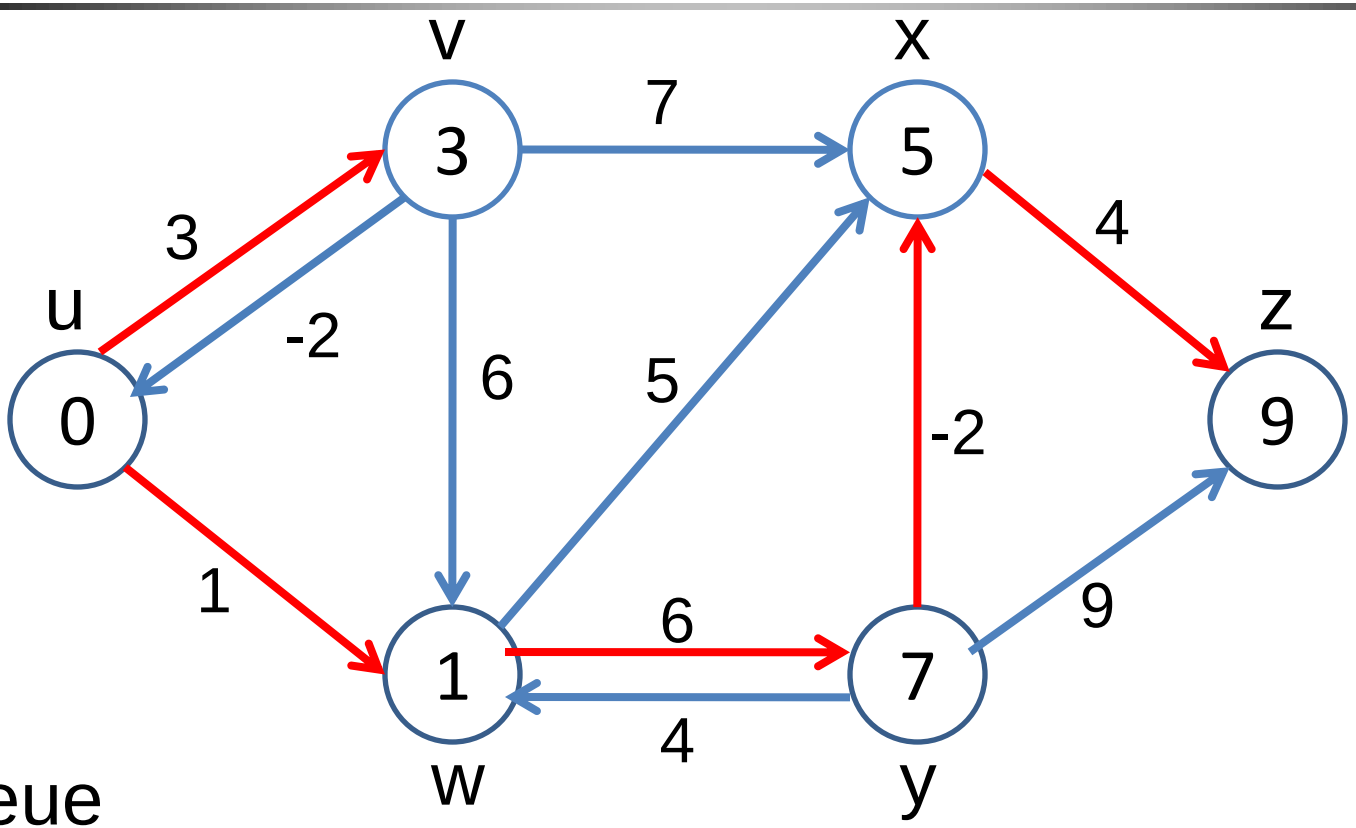
SPFA



Queue

Now: z

SPFA



Queue

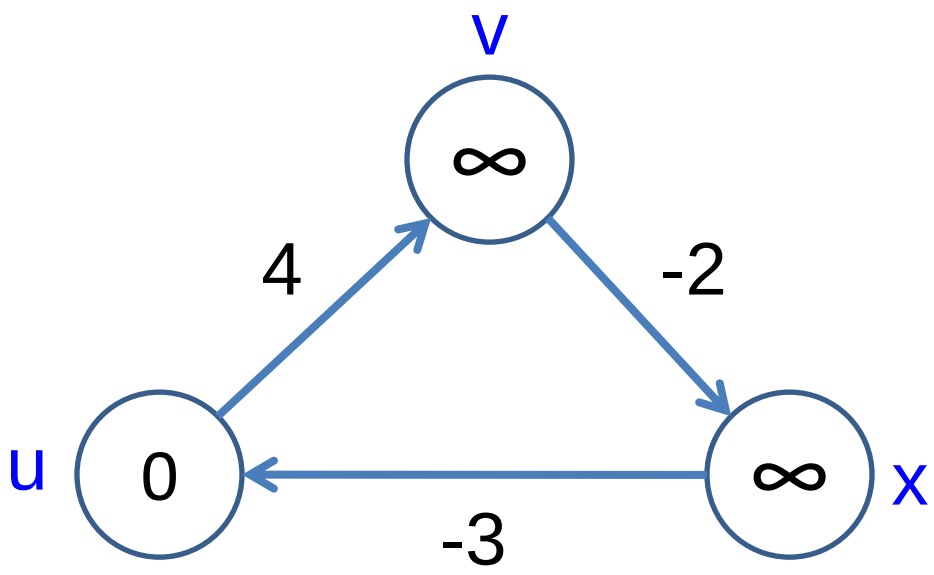
Done!

SPFA

```
SPFA(){
    dis[i]=INF, inqueue[i]=false, for all i
    dis[source]=0;
    inqueue[source]=true;
    queue.push(source);
    while(!queue.empty()){
        now=queue.front();
        inqueue[now]=false;
        queue.pop();
        for each node v adjacent to now{
            if(dis[now]+w(now,v)<dis[v]){
                dis[v]=dis[now]+w(now,v);
                if(!inqueue[v]){
                    queue.push(v);
                    inqueue[v]=true;
                }
            }
        }
    }
}
```

SPFA

- Negative cycle?

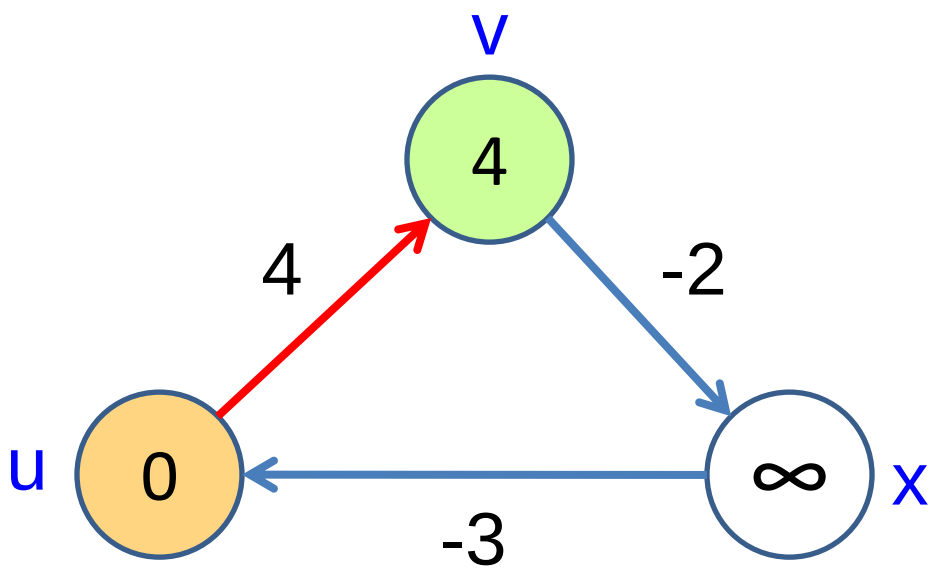


Queue

u	

SPFA

- Negative cycle?



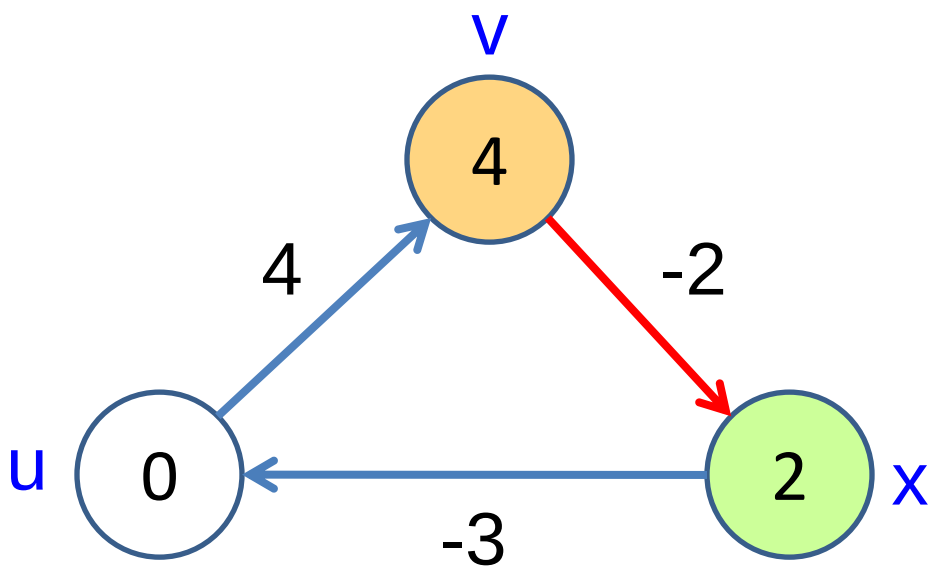
Queue

Now: u

v has changed

SPFA

- Negative cycle?



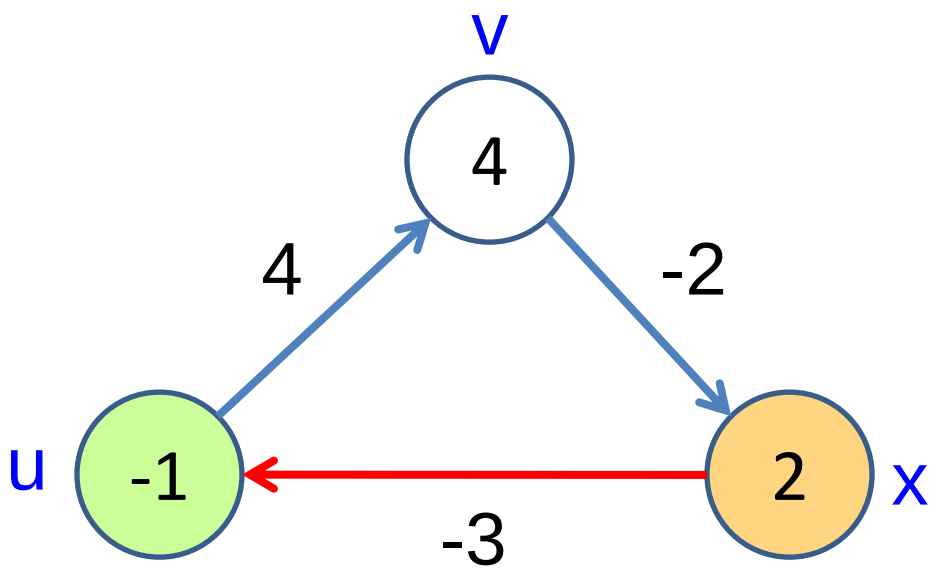
Queue

Now: v

x has changed

SPFA

- Negative cycle?



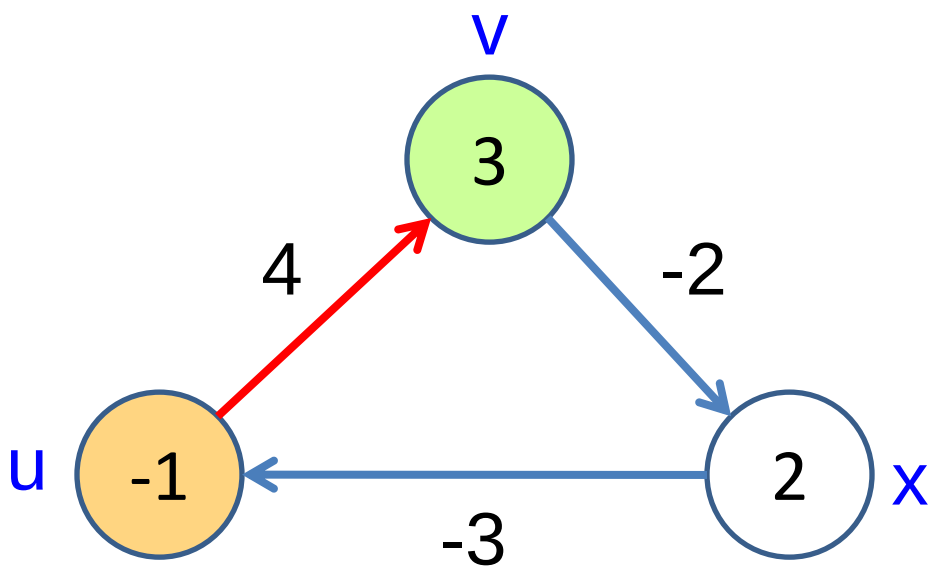
Queue

Now: x

u has changed

SPFA

- Negative cycle?



Queue

Now: u

v has changed

Infinite Loop…

SPFA

- Count the times of pushing a node in queue
  - No more than n-1 times
  - Otherwise, negative cycle exists.

- 若找到負環，必是由source出發後遇到的負環
  - source走不到的地方無從得知

SPFA

```
SPFA(){
    dis[i]=INF, inqueue[i]=false, count[i]=0 for all i
    dis[source]=0;
    inqueue[source]=true;
    queue.push(source);
    while(!queue.empty()){
        now=queue.front();
        inqueue[now]=false;
        queue.pop();
        for each node v adjacent to now{
            if(dis[now]+w(now,v)<dis[v]){
                dis[v]=dis[now]+w(now,v);
                if(!inqueue[v]){
                    queue.push(v);
                    inqueue[v]=true;
                    count[v]++;
                    if(count[v]>=n) return true;
                }
            }
        }
    }
    return false;
}
```

SPFA

- Complexity
  - O(kE), where k<<V, for average case
  - O(VE), for worst case

Bellman Ford vs SPFA

	Bellman Ford	SPFA
Edge	Discrete data structure	Adjacency list
Negative cycle	Some where in Graph	From source
Complexity	O(VE)	O(kE)

All Pair Shortest Path

APSP

- How?
  - Bellman Ford * V times: $O(V^2E)$
  - SPFA * V times: $O(kVE)$

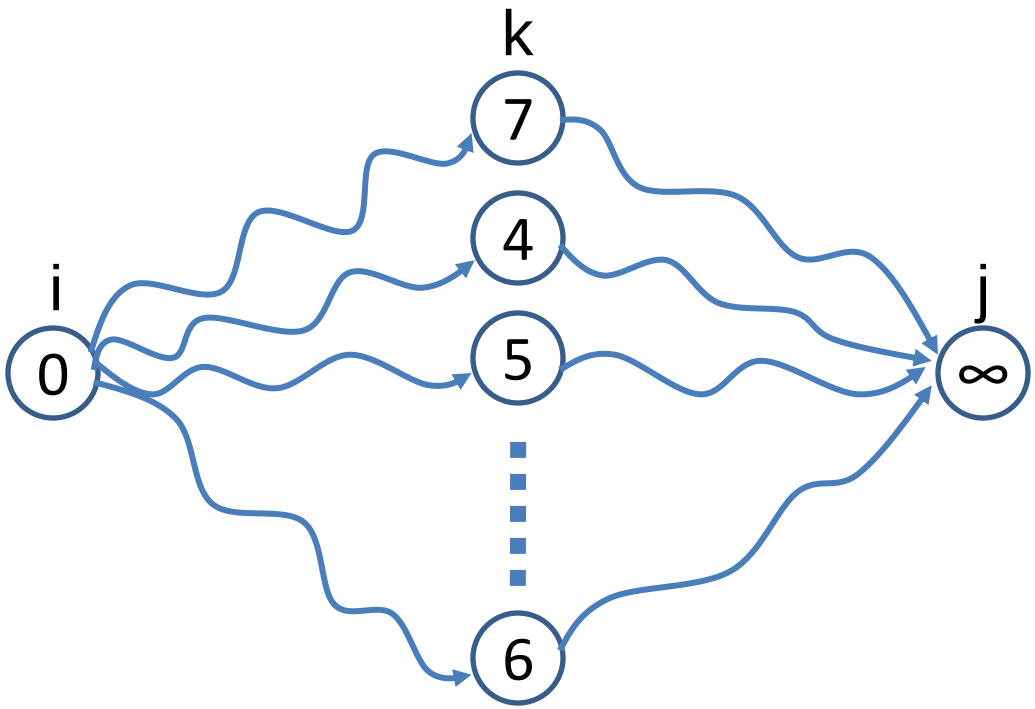
- Floyd-Warshall algorithm
  - $O(V^3)$

Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)

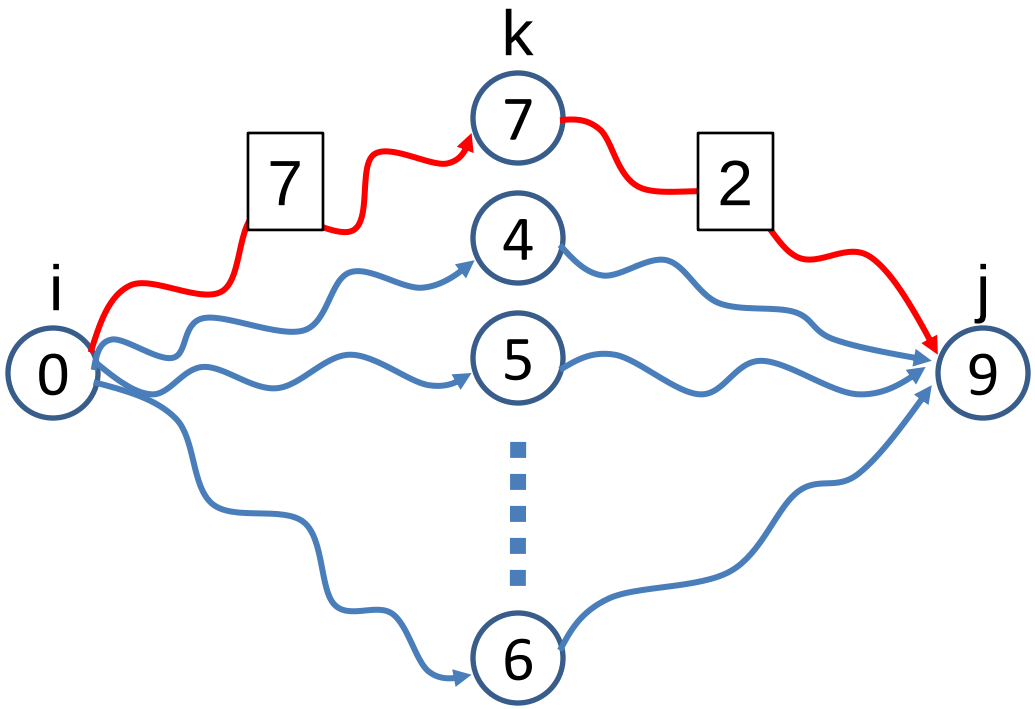
Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)



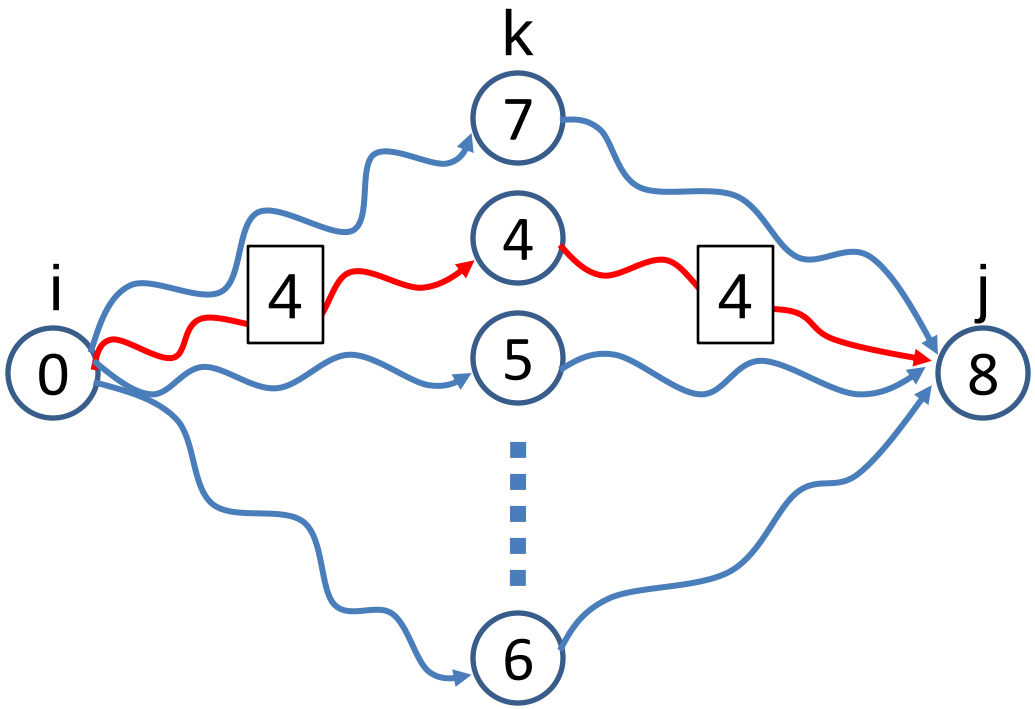
Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)



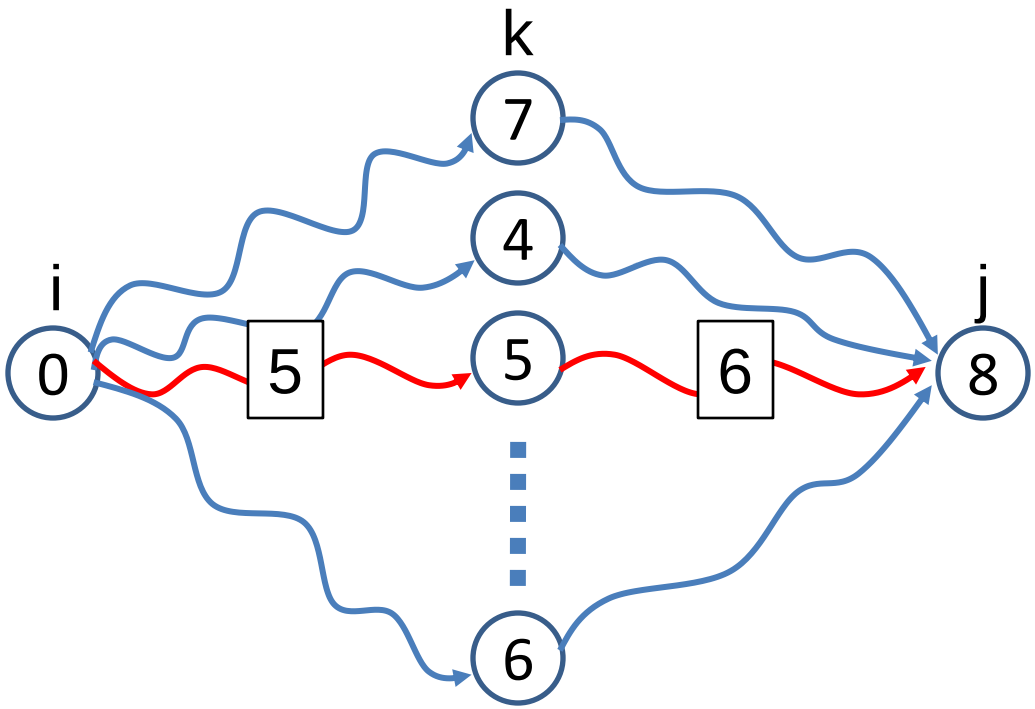
Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)



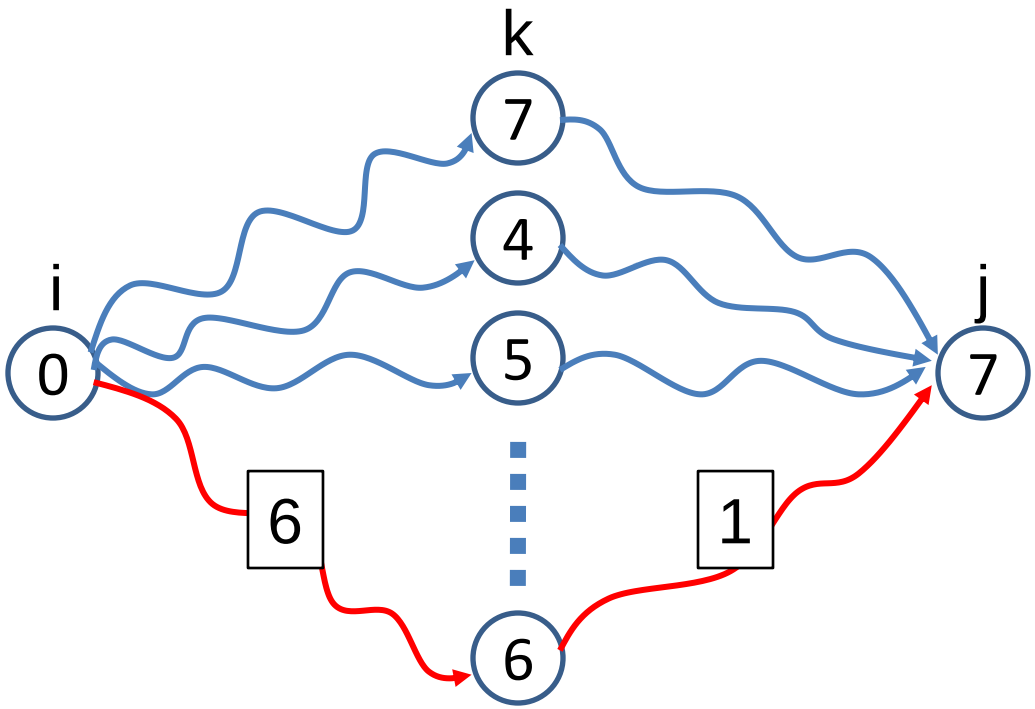
Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)



Floyd

- Enumerate all node k as relay point
  - Repeat for each pair(i,j)



Floyd

- Pseudo code

```
1  Floyd(){
2      dis[i][j]=0, for i=j
3      dis[i][j]=w(i,j), for each edge w(i,j)
4      dis[i][j]=INF, otherwise.
5      for(k=0;k<n;k++)
6          for(i=0;i<n;i++)
7              for(j=0;j<n;j++)
8                  if(dis[i][k]+dis[k][j]<dis[i][j])
9                      dis[i][j]=dis[i][k]+dis[k][j];
10 }
```

Floyd

- Complexity
  - $O(V^3)$

Floyd

- Complexity
  - $O(V^3)$
- How about multiple edges between (a,b)?
  - Use the shortest one

Practice3

- POJ 1125 - Stockbroker Grapevine

Summary

- SSSP:
  - Bellman Ford
  - SPFA
  - Dijkstra (Google it by yourself)

- APSP:
  - Floyd
  - SPFA * V times (Sometimes better than Floyd)

Learn more!

- How to output one of shortest paths?
  - Table: record the previous one

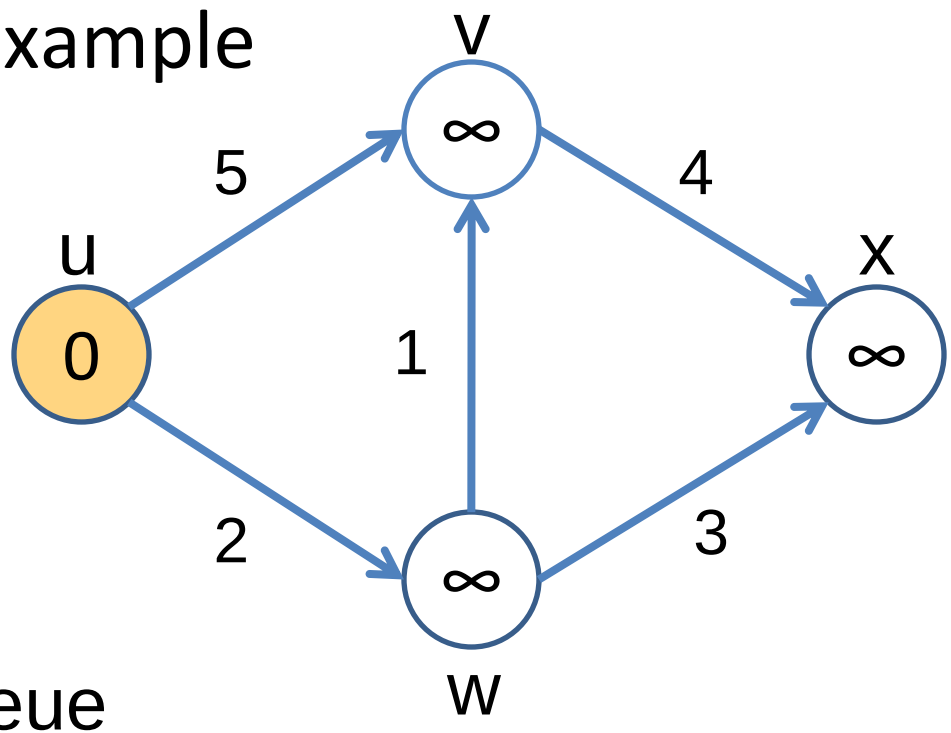
Learn more!

- How to output one of shortest paths?
  - Table: record the previous one

- Once relax successfully, update prev[id].

Learn more!

- Example



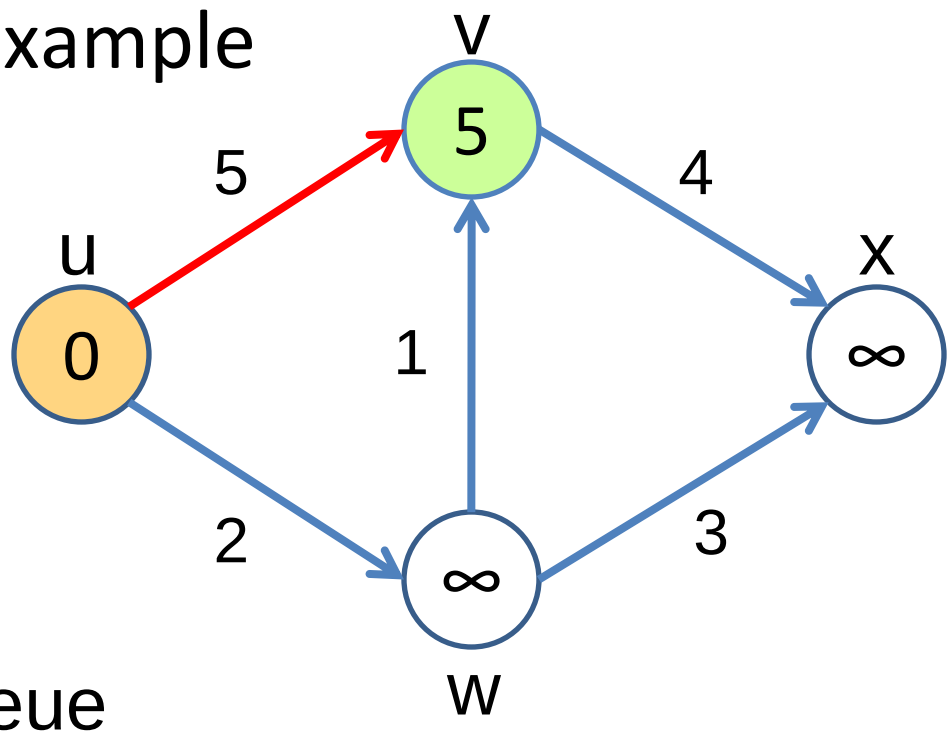
id	prev[id]
u	None
v	None
w	None
x	None

Queue

Now: u

Learn more!

- Example



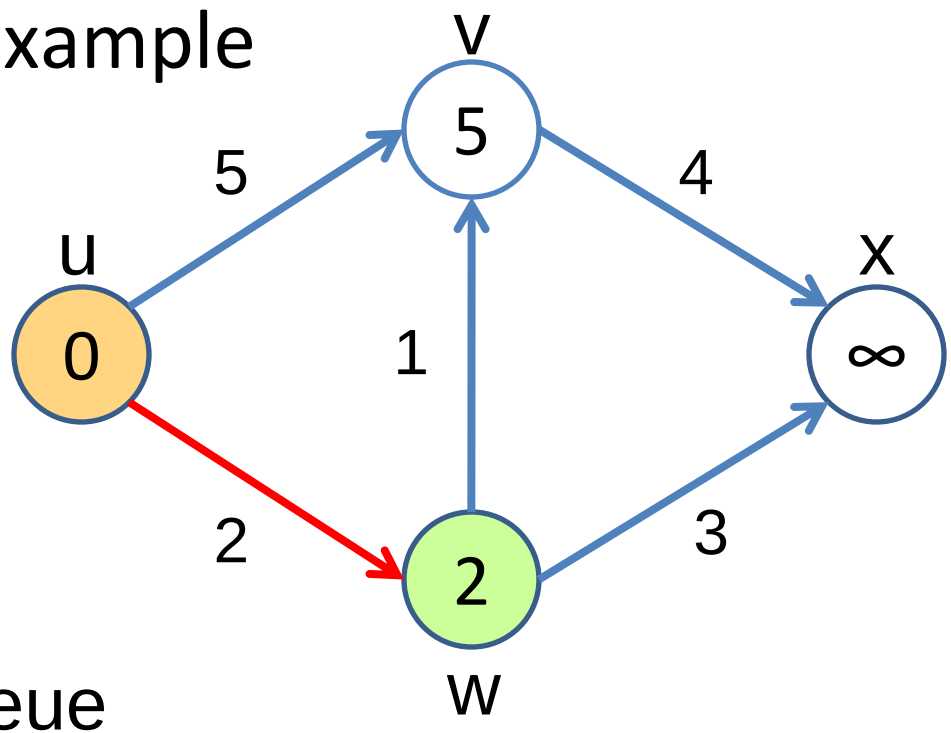
id	prev[id]
u	None
v	u
w	None
x	None

Queue

Now: u

Learn more!

- Example



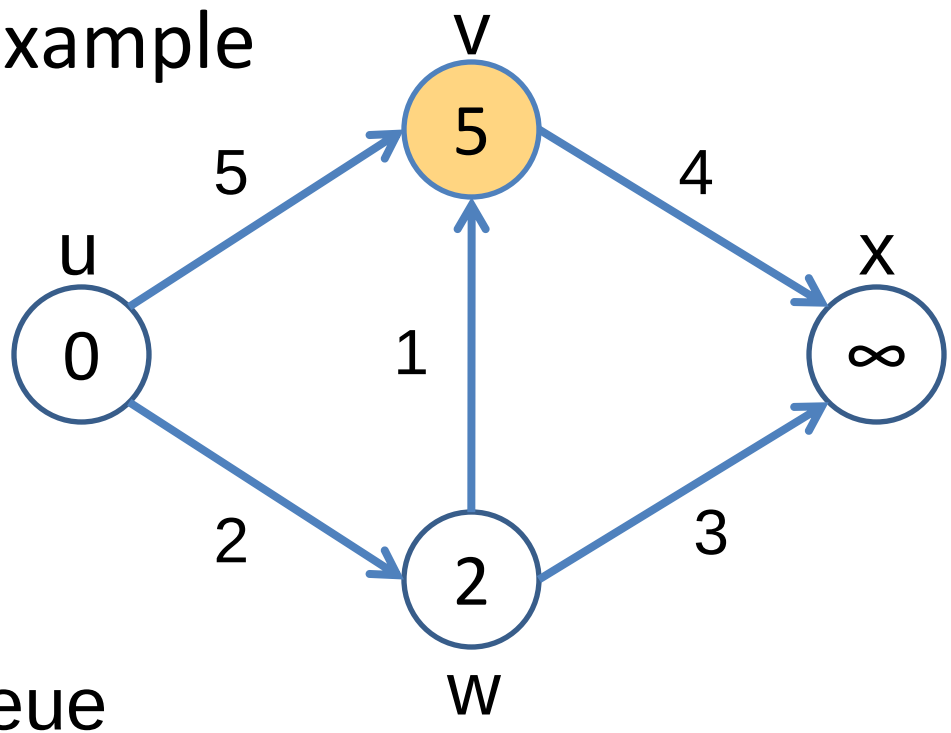
id	prev[id]
u	None
v	u
w	u
x	None

Queue

Now: u

Learn more!

- Example



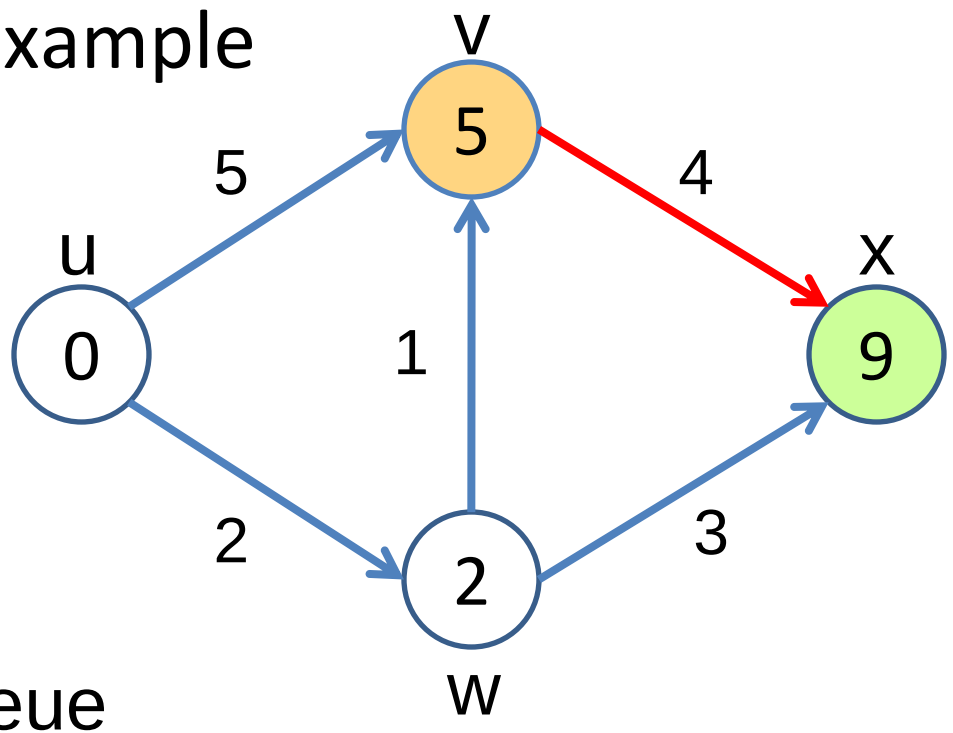
id	prev[id]
u	None
v	u
w	u
x	None

Queue

Now: v

Learn more!

- Example



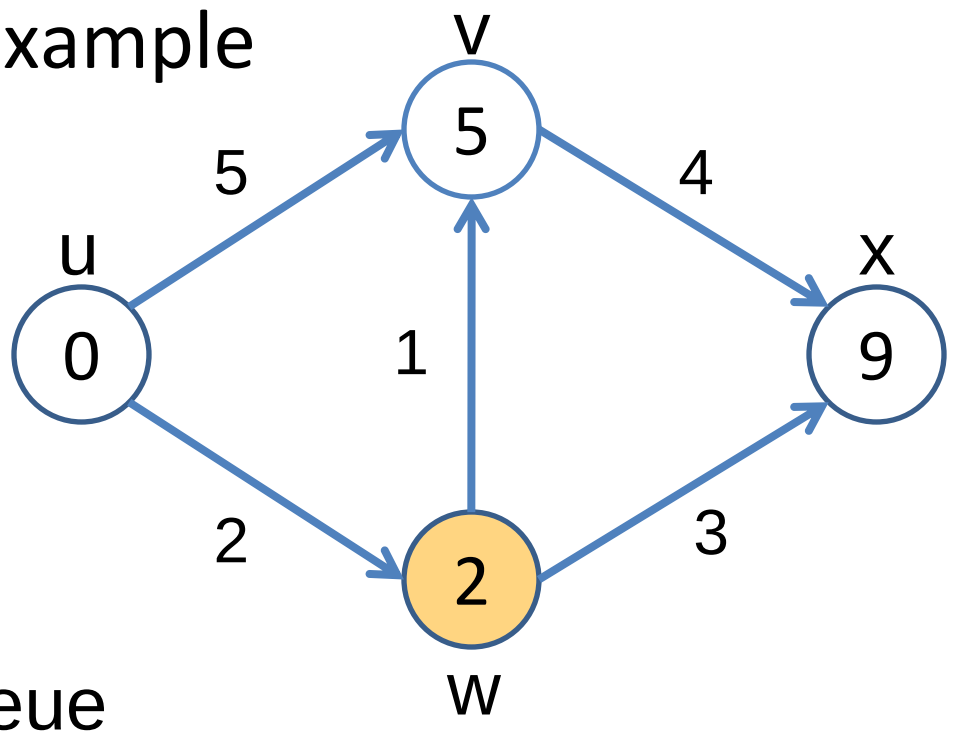
id	prev[id]
u	None
v	u
w	u
x	v

Queue

Now: v

Learn more!

- Example



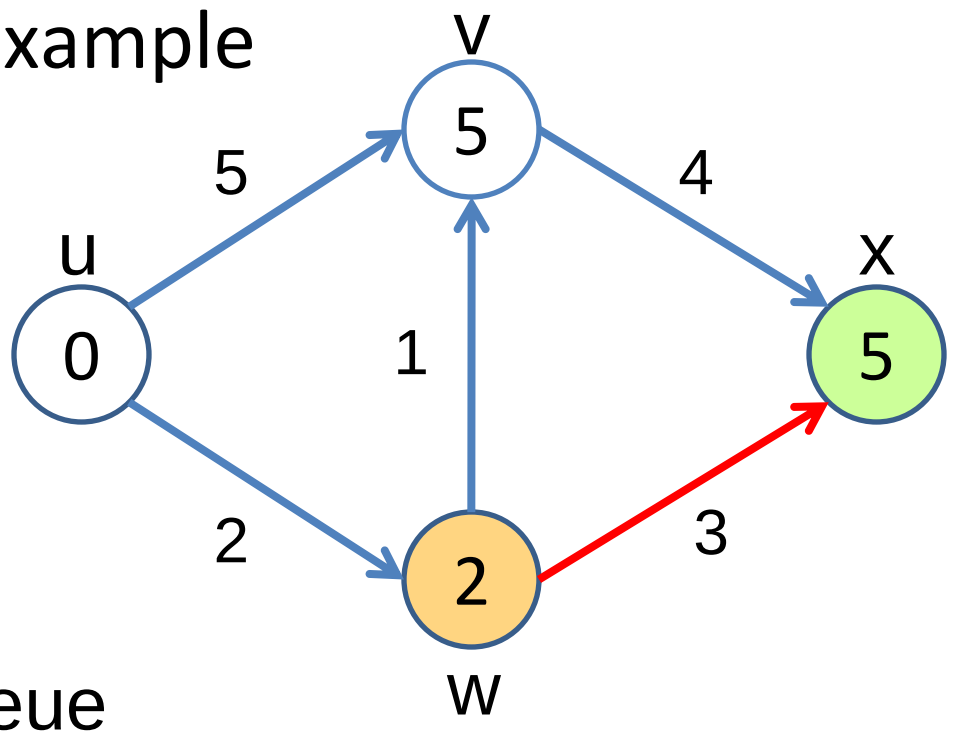
id	prev[id]
u	None
v	u
w	u
x	v

Queue

Now: w

Learn more!

- Example



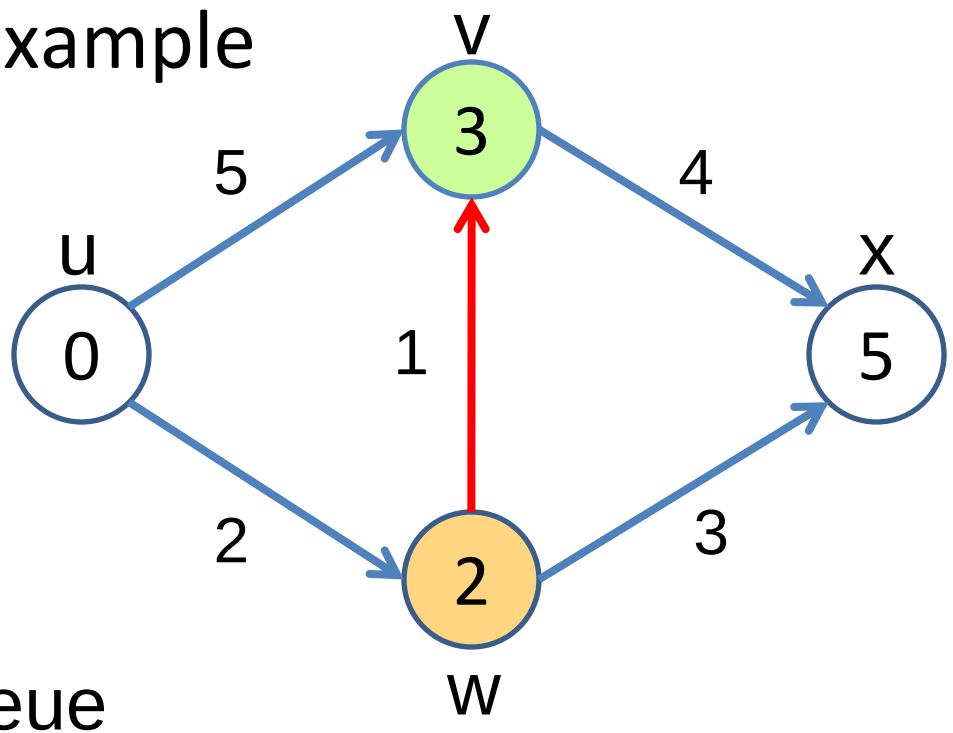
id	prev[id]
u	None
v	u
w	u
x	w

Queue

Now: w

Learn more!

- Example



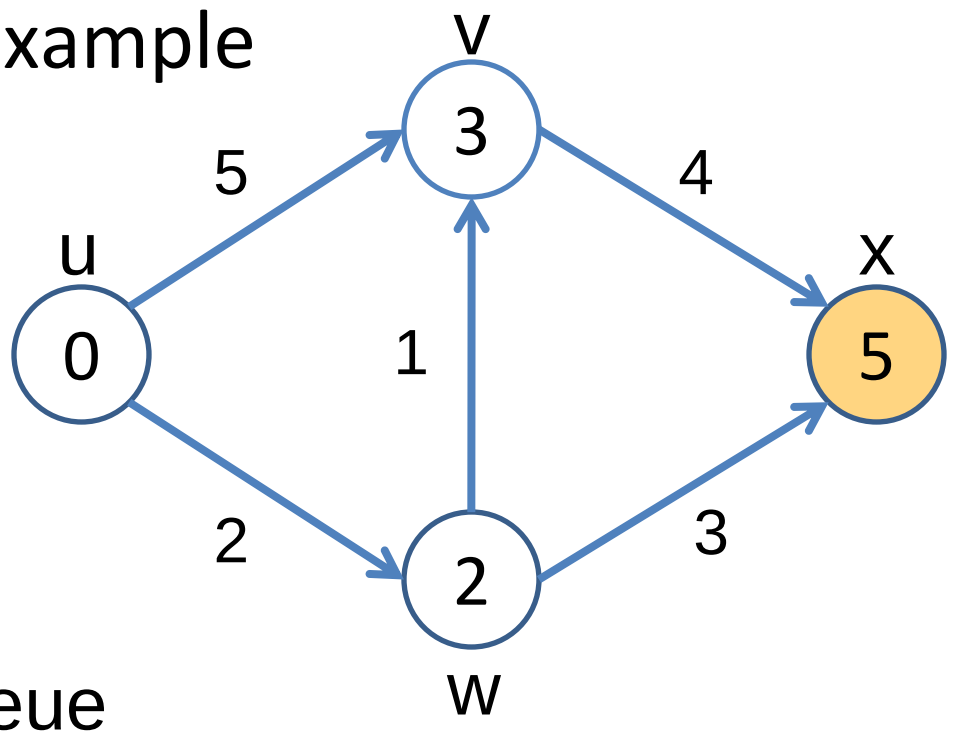
id	prev[id]
u	None
v	w
w	u
x	w

Queue

Now: w

Learn more!

- Example



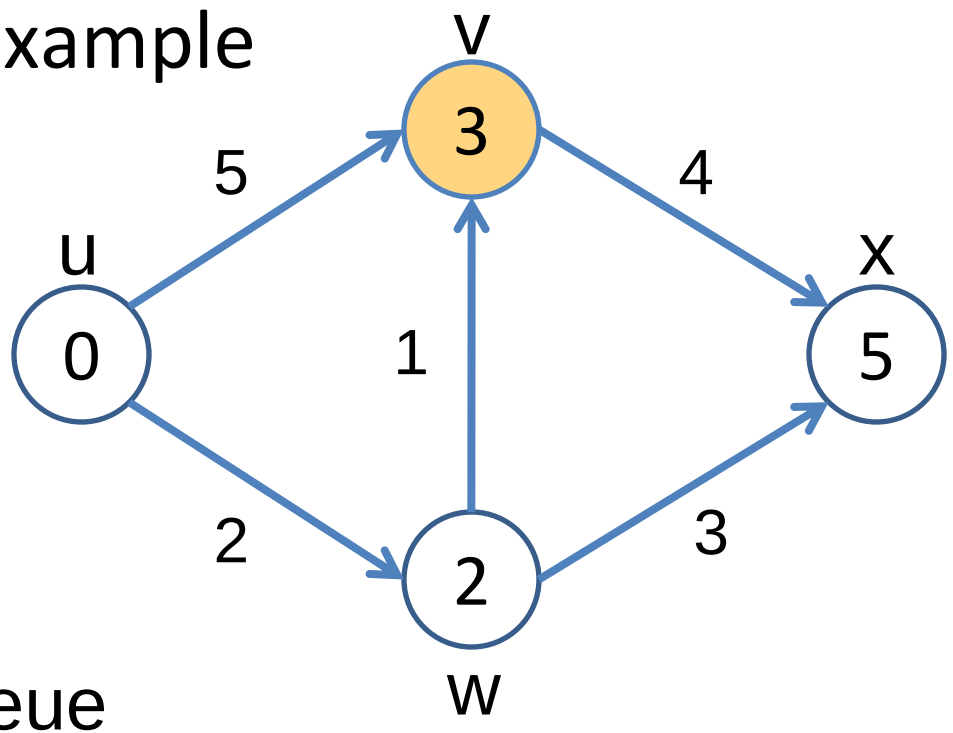
id	prev[id]
u	None
v	w
w	u
x	w

Queue

Now: x

Learn more!

- Example



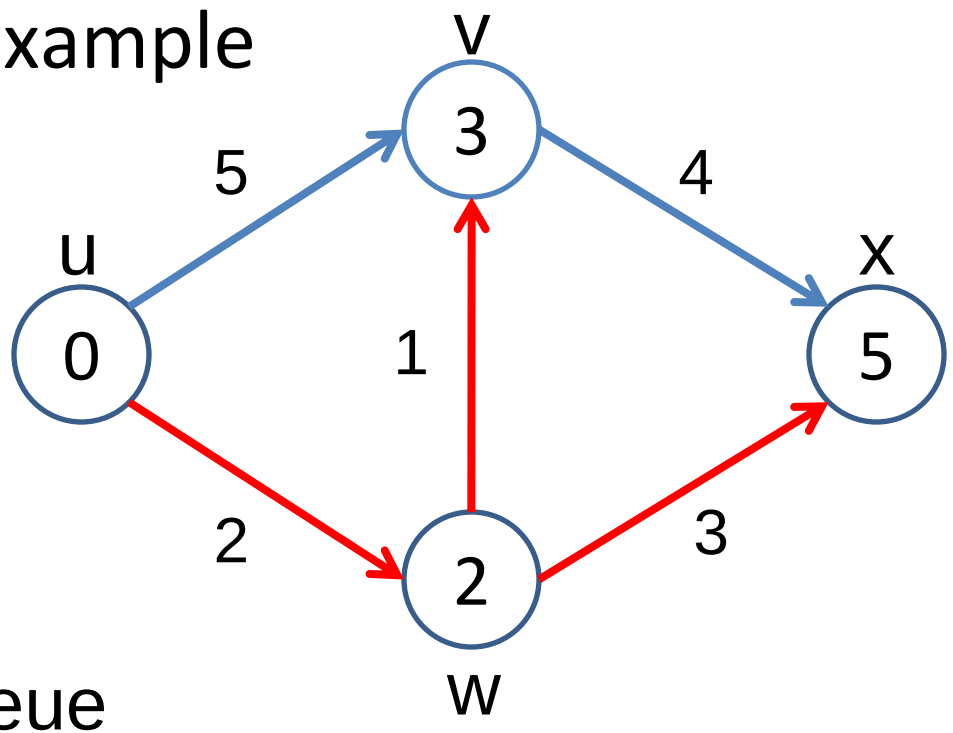
id	prev[id]
u	None
v	w
w	u
x	w

Queue

Now: v

Learn more!

- Example



id	prev[id]
u	None
v	w
w	u
x	w

Queue

Done!

Thank you for your attention!