Shortest Path

Definitions

- ***Formal definition:*** Given a graph G=(V, E, W), where each edge has a weight, find a shortest path from s to v for some interesting vertices s and v.
- s—source
- v—destination

Basic Property

- Suppose that a shortest path *p* from a source *s* to a vertex *v* can be decomposed into

$$s \xrightarrow{p'} u \longrightarrow v$$

for some vertex *u* and path *p'*.

Then, the weight of a shortest path from *s* to *v* is

$$\delta(s,v) = \delta(s,u) + w(u,v)$$

It implies that the sub-path of optimal (shortest) path is also optimal (shortest). Consider it the other way round, we could find a longer optimal path *(of length n)* by extending a shorter optimal path *(of length n-1)* with one extra edge

Relaxation

- The process of relaxing an edge (u,v) consists of testing whether we can improve the shortest path to v found so far by going through u and,if so,updating d[v] and π [v].
- define RELAX(u,v,w) as…..
- if d[v]>d[u]+w(u,v) { // if a shorter optimal path (*d[u]*) + one extra // edge w(u,v) is shorter than current best…
- d[v] = d[u]+w(u,v)
- π [v] = u // record the *"parent".. Needed for path tracing*
- }

Relaxation

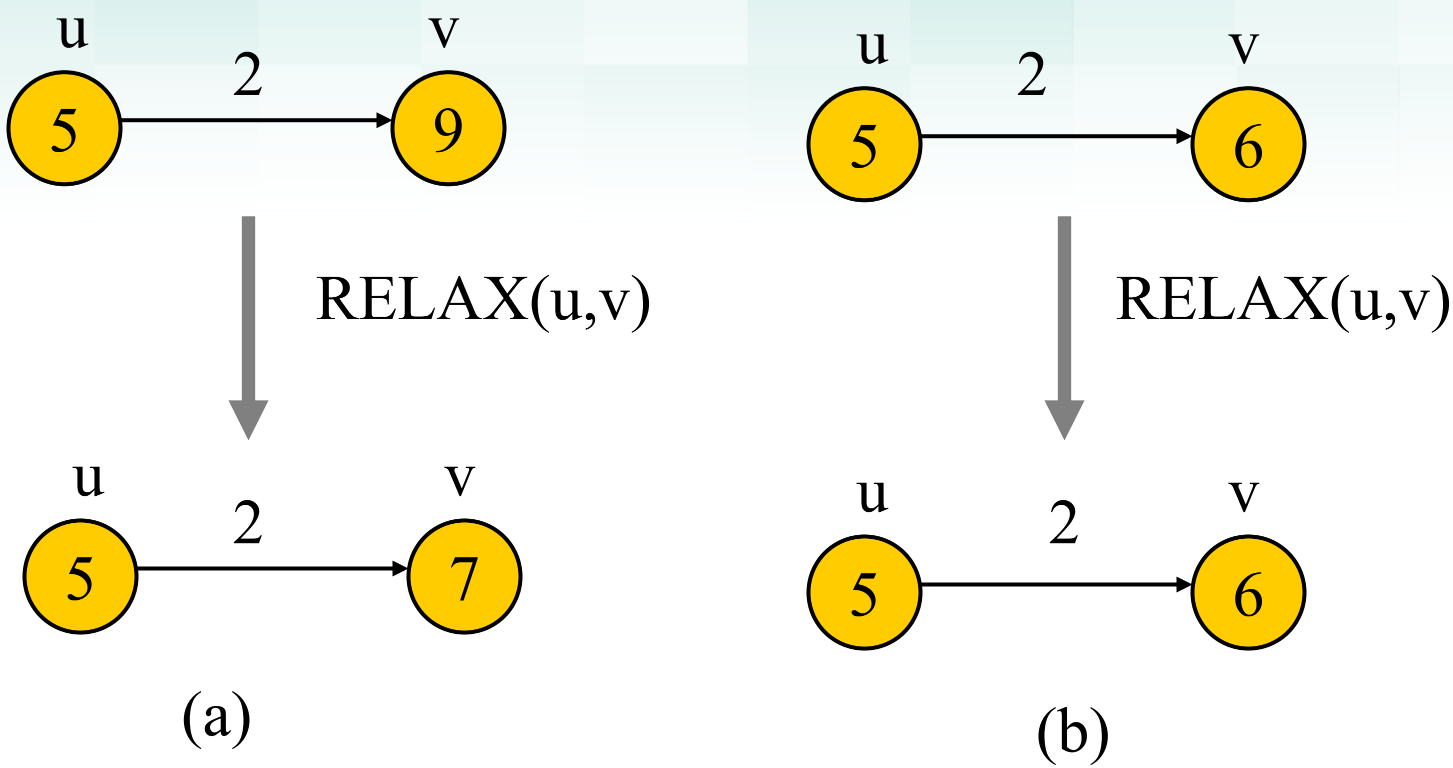


Figure2 Relaxation of an edge (u,v).The shortest-path estimate of each vertex is shown within the vertex. (a)Because d[v]>d[u]+w(u,v) prior to relaxation, the value of d[v] decreases. (b)Here, d[v] $\leq$ d[u]+w(u,v) before the relaxation step,so d[v] is unchanged by relaxation.

Single Source Shortest Path

Single Source Shortest Path

- Single Source Shortest Path
  - Only consider a fixed source ***s***.
  - To find out all shortest path to all ***v*** in ***V***
- Dijkstra's algorithm
  - Faster *(for max speed, need data struct such as heap)*
  - All edge must be **positive**
- Bellman-Ford algorithm
  - Simple design but slower
  - Can detect negative weight cycle

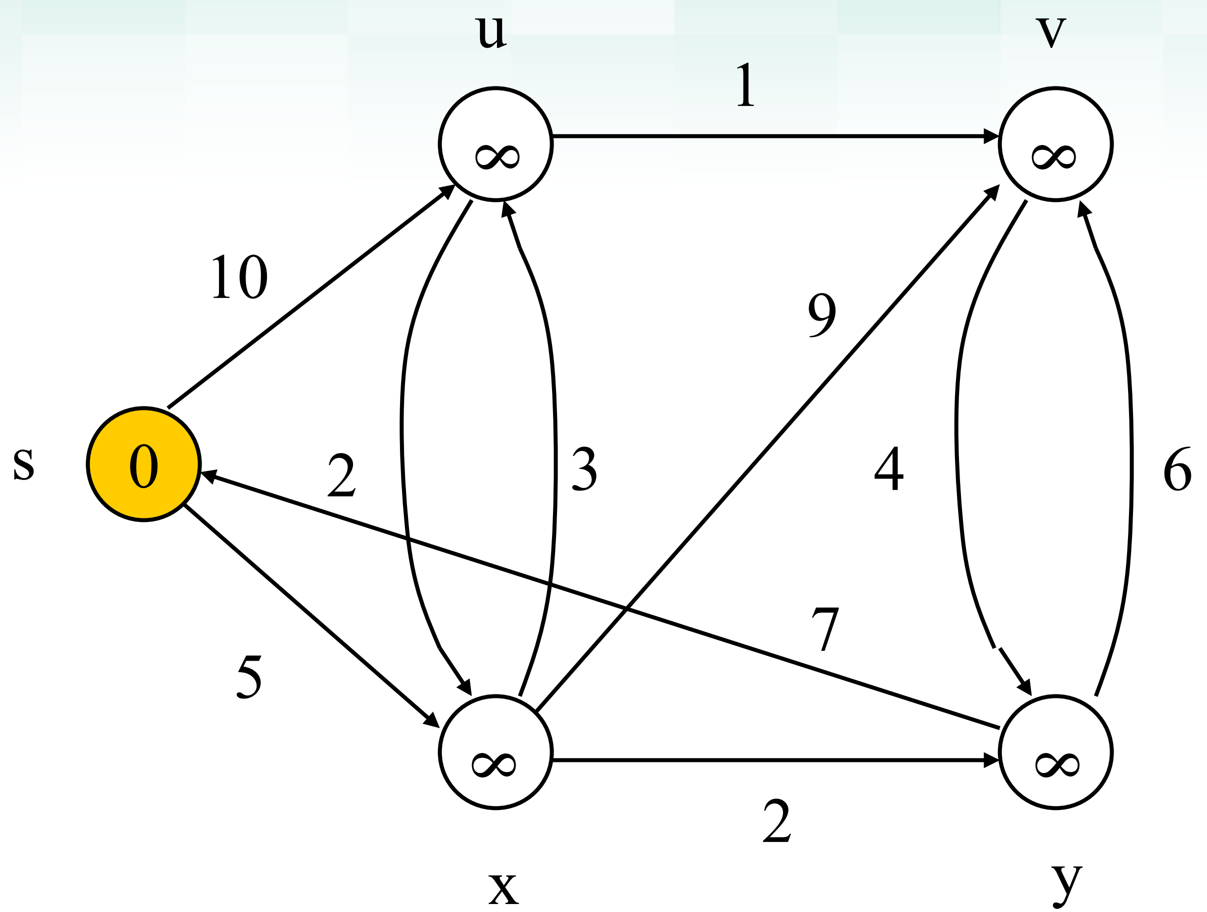
Dijkstra's algorithm

- Initialization
  - For each vertex v $\in$ V, d[v] denotes an upper bound on the weight of a shortest path from source s to v
  - d[v] will be the weight of shortest path after the execution of algorithm
  - Set d[] = Infinity at the beginning
  - Set d[s] = 0

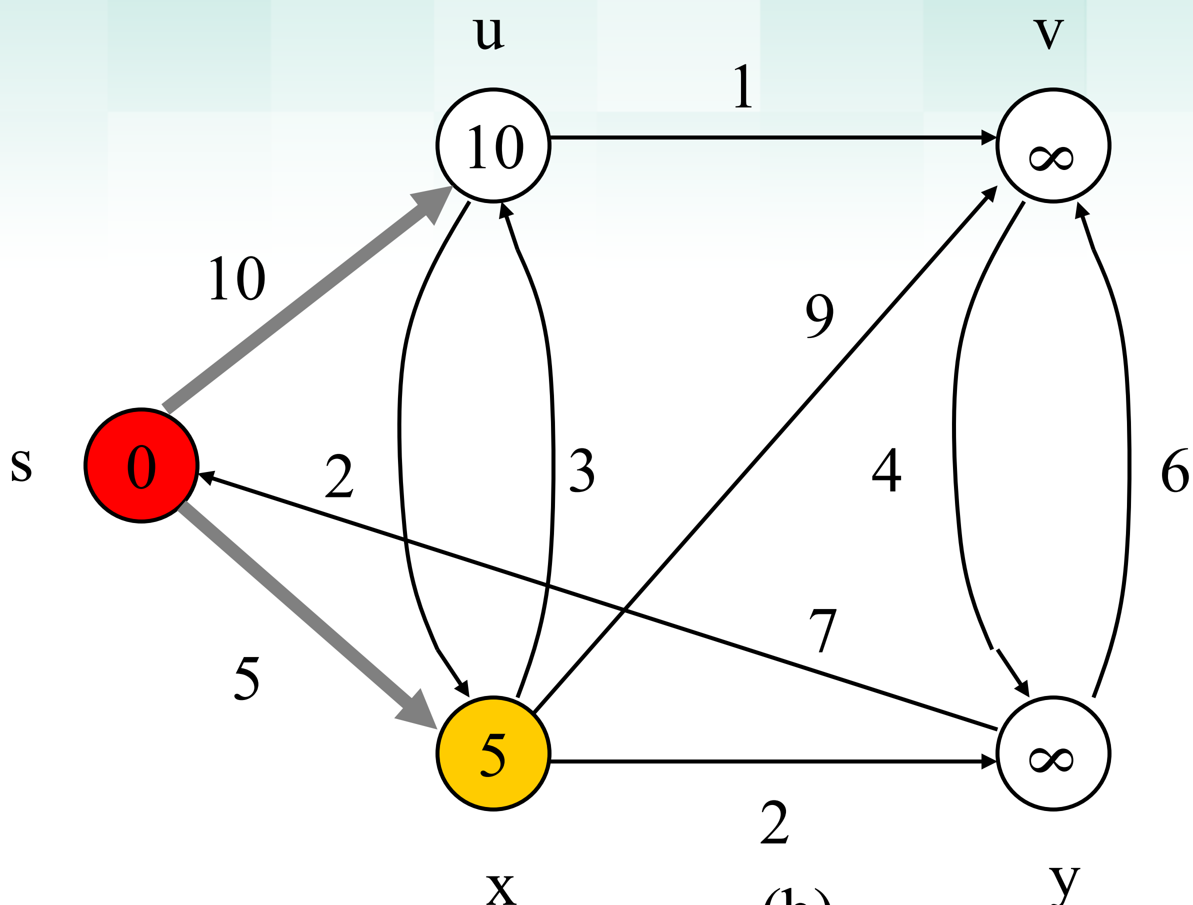
Dijkstra’s Pseudo code

```
Initialize
Q <- V[G]                                   // Q = set of “unsolved” vertices
While (|Q| != 0) {
        u <- extract-min (Q) //remove u from set Q
        // Path to  u is confirmed as shortest……but…why?
        for each v in adj[u] { // set if there’s better path
                Relax(u,v,w)   // which goes through u
        } // more precisely… for each v in adj[u] which is also in Q
}
```

Visual Demo

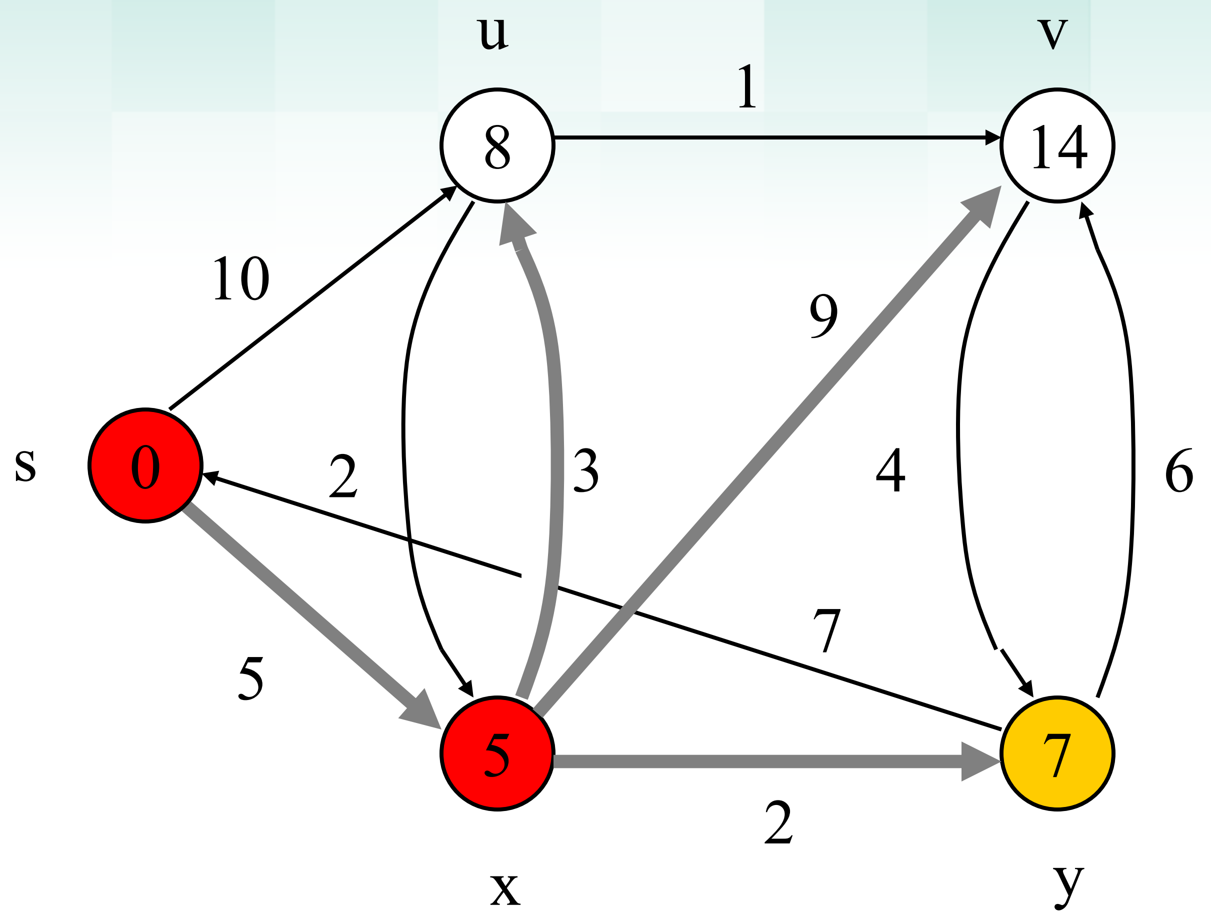


(a)

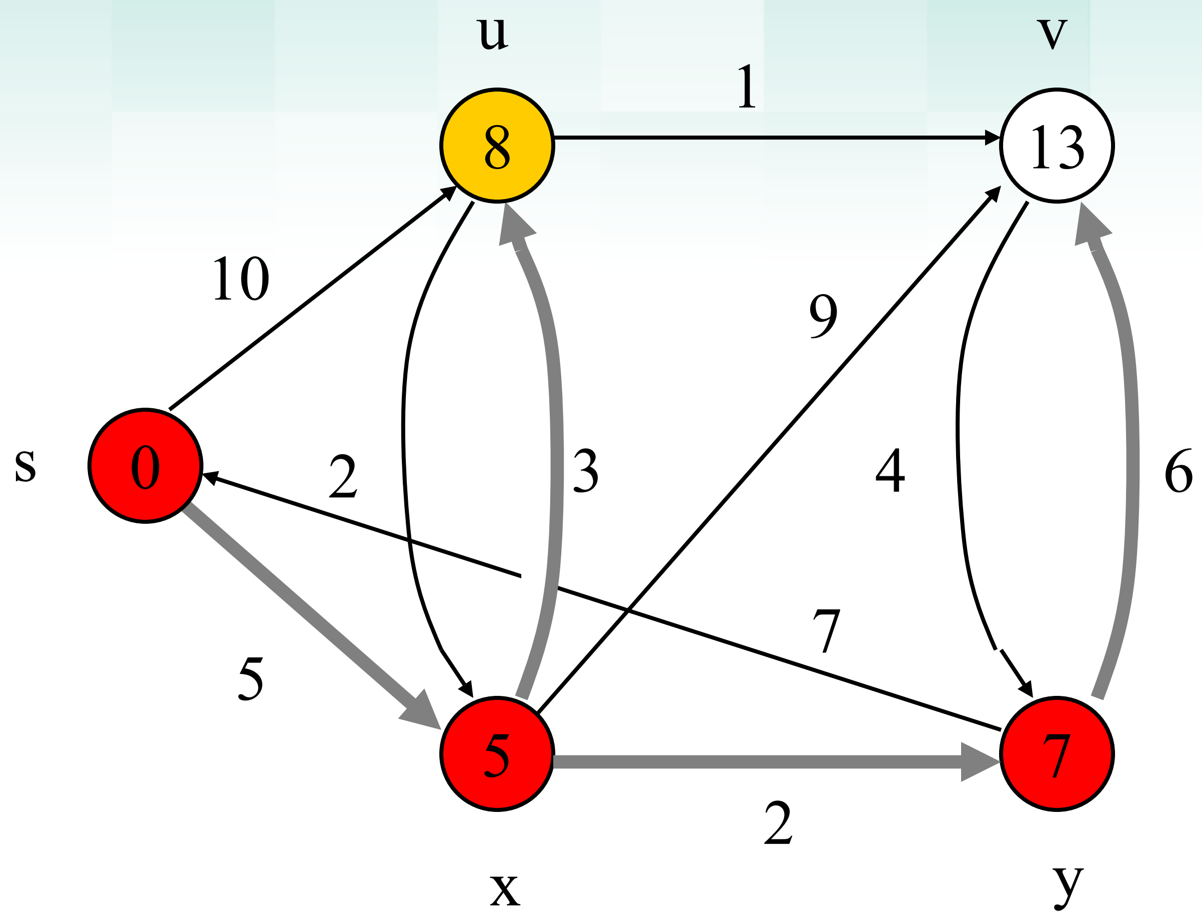


(b)

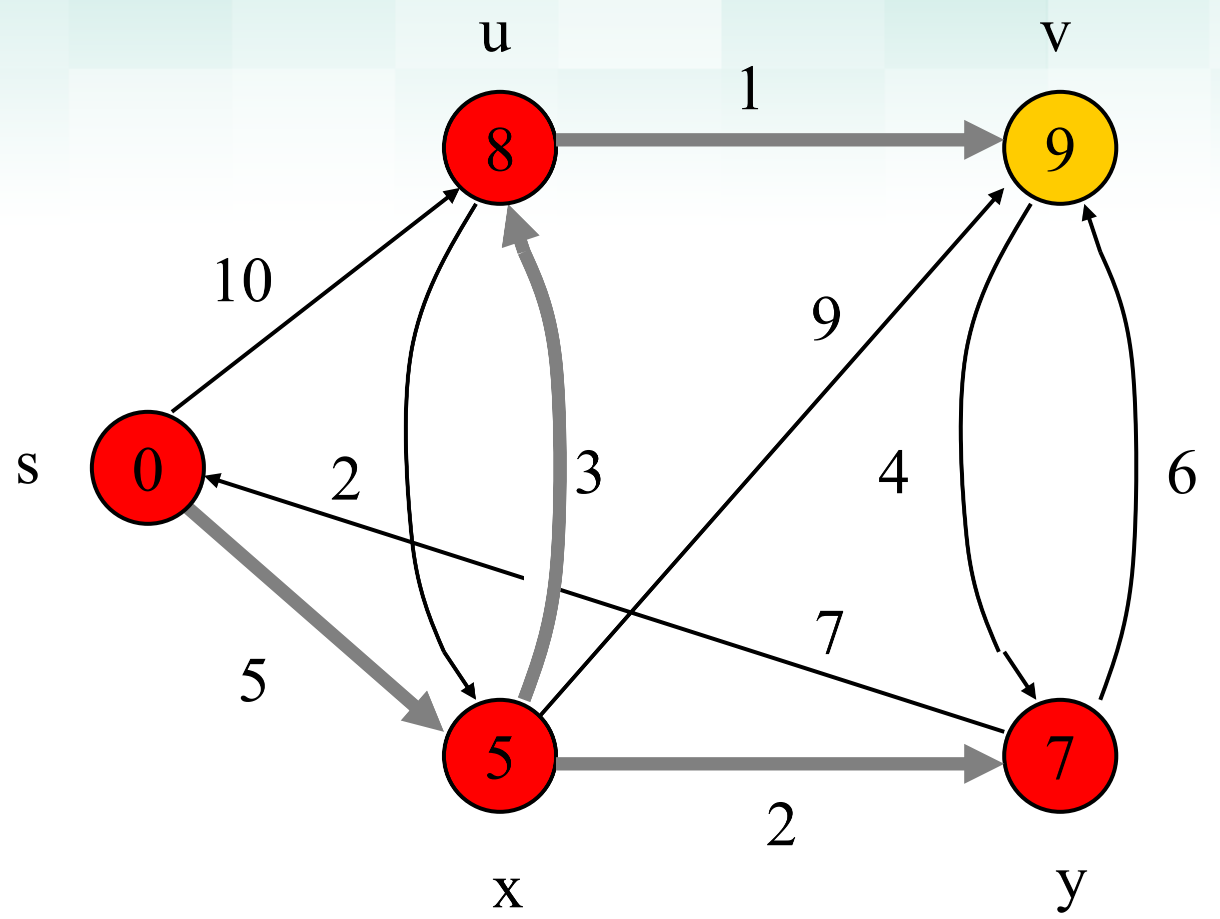
(s,x) is the shortest path using one edge. It is also the shortest path from s to x.



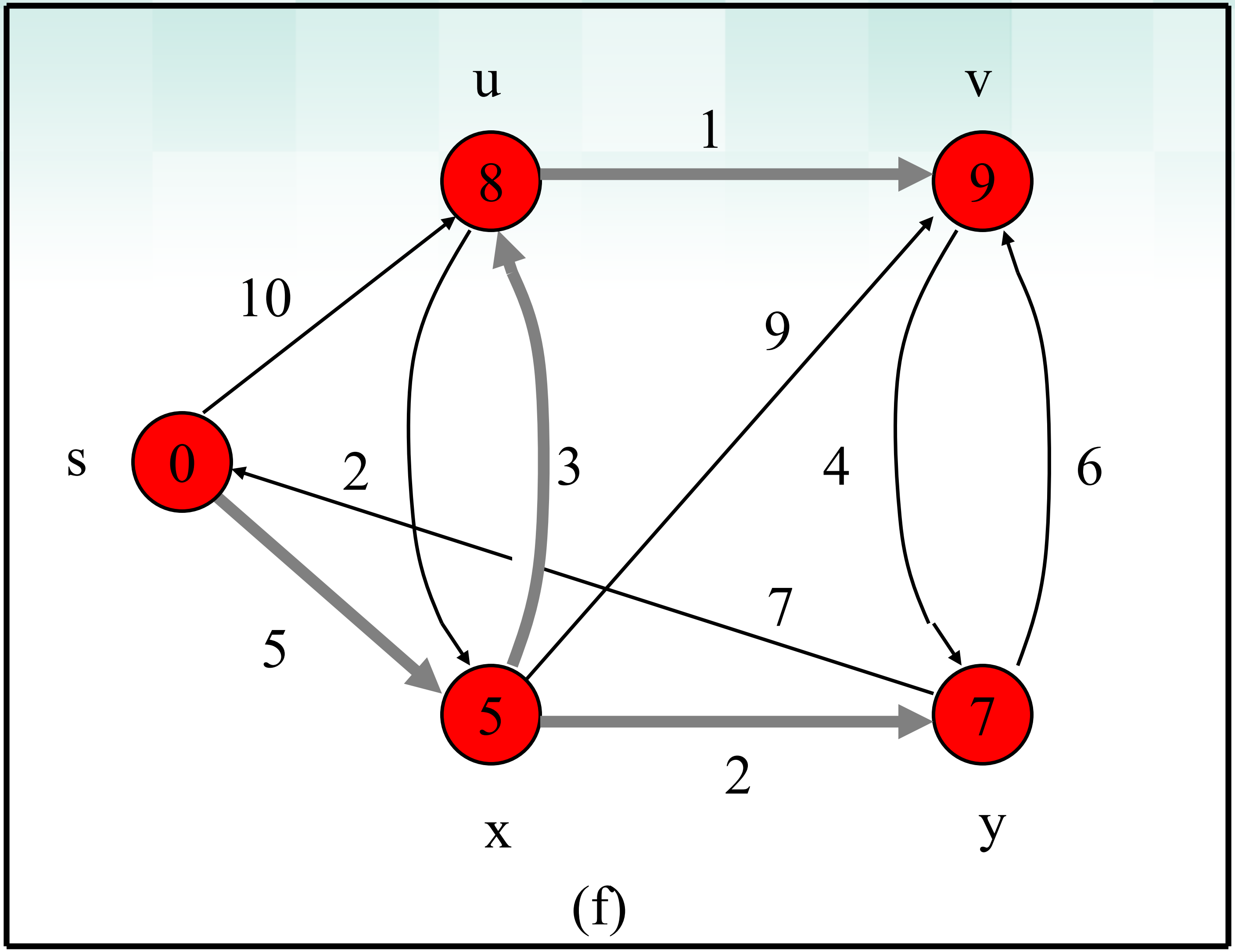
(c)



(d)



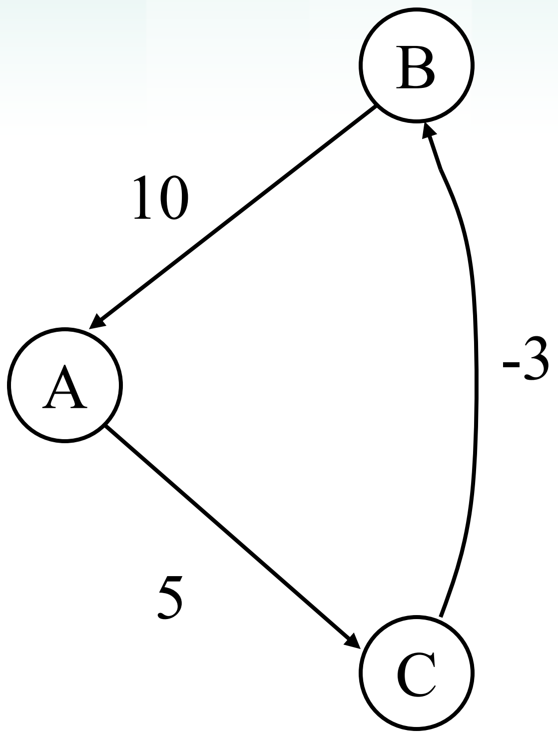
(e)


u
v
1
8
9
10
9
s
0
2
3
4
6
7
5
5
7
2
x
y
(f)

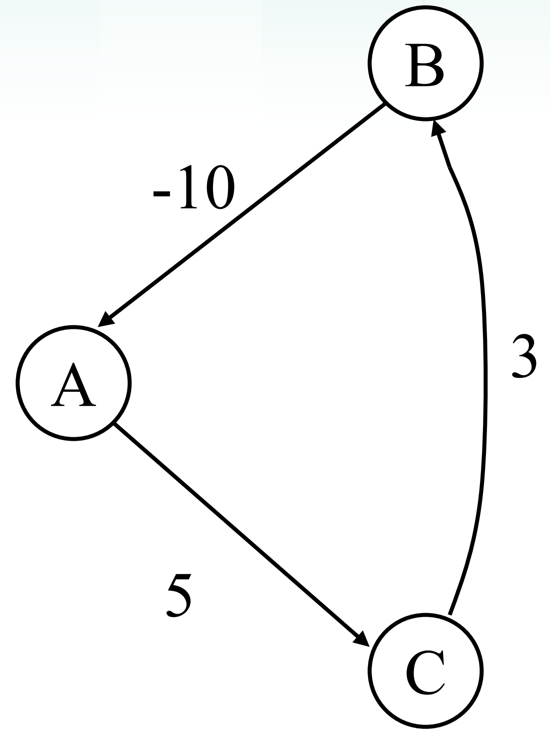
Consideration on Negative Weight

- Negative Weight
  - Edge weight may be negative!
- Negative-weight cycle
  - Total weight in a cycle (circuit) is negative
  - If no negative-weight cycles reachable from the source *s*, then all shortest path weights are well defined, even if it has a negative value.
  - If there is a negative-weight cycle on some path from s to v, we define it is $-\infty$

Consideration on Negative Weight



Negative Edge but no Negative Cycle



Negative Edge with Negative Cycle

Bellman-Ford algorithm

Algorithm:

```
Initialize
for i = 1 to |V[G]|-1   // run |V[G]|-1 times = length of longest possible path
    for each edge (u, v) in E[G]
        Relax(u, v, w)
// checking stage…if relaxation is still possible after |V[G]|-1 times...negative cycle
for each edge (u, v) in E[G]
    if d[v] > d[u] + w(u, v)
    then return false

return true
```

Simple proof for Bellman-Ford

- If there is a path p from some v to some u, p has at most |V[G]| - 1 edges
- Each iteration of the outer loop relax all edges in E[G]
- After the i iteration, d[v] stored the shortest distance from s to v for at most i edges
- Therefore, after the |V[G]| - 1 pass of the loop, d[] stored the shortest from s to every v in V[G]

What happen if there is –ve cycle?

- If there is a negative cycle reachable from s, walking every round in the cycle will make the path shorter, so there is no shortest path
- Therefore after |V[G]|-1 pass of the loop, d[v] will still greater than d[u] + w(u, v), for v is any vertex in the cycle
- This will make last for loop in the algorithm returns false

All-pair Shortest Paths

All-pair Shortest Paths

- ***Problem definition:***
  - Given a weighted,directed graph $G=(V,E)$, for every pair of vertices $u, v \in V$, find a shortest (least weight) path from u to v, where the weight of a path is the sum of the weights of its constituent edges.

Solve the problem by Single-Source shortest paths algorithm:

- If all edge weights are nonnegative, we can use Dijkstra's algorithm repeatly.
- Since Dijkstra's algorithm can compute all shortest path from a single point, it solves all-pairs shortest paths in

$$O(n^2) \times n = O(n^3)$$

- Any other method?
  - Method by Matrix

Method by Matrix

- Notations:
  - Let l be the minimum weight from ***i*** to ***j*** at most contain ***m*** edge
  - So for ***m*** = 0 then

$$l_{ij}^{(0)} = \begin{cases} 0 \; if \; i = j \\ \infty \; if \; i \neq j \end{cases}$$

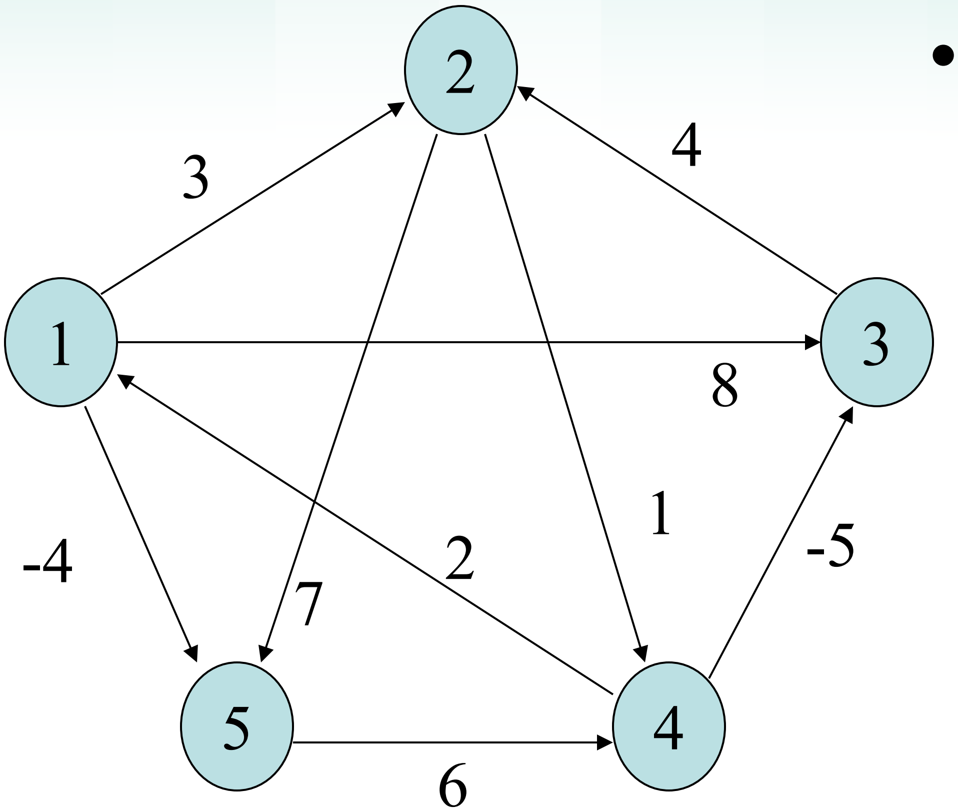
- Premise:
  - This is a recursive approach in solving this kind of problem.

$$l_{ij}^{(m)} = \min\left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \left(l_{ik}^{(m-1)} + w_{kj} \right) \right)$$

  - where w = the weight of the edge.

Well…what do you think?

Method by Matrix



- Input Adjacency Matrix

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

Pass 1: Input Adjacency Matrix

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

We define:

dij(m) is the element in matrix D(m)

$d_{ij}(m)$ --- minimum weight of any path from vertex ***i*** to vertex ***j*** that contains at most ***m*** edges

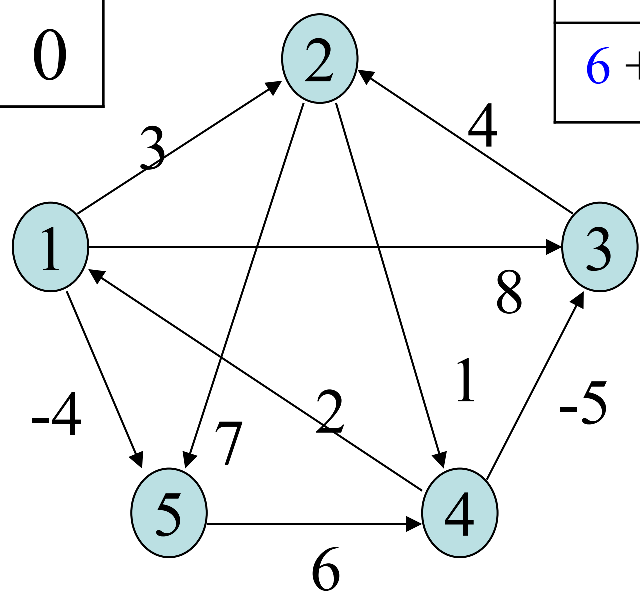
How to get this?

Pass 2 in matrix

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

→

0	3	8	-4 + 6	-4
1 + 2	0	-5+1	1	7
∞	4	0	4+1	4+7
2	-5 + 4	-5	0	-4+2
6 + 2	∞	-5+6	6	0



Pass 2 in matrix

0	3	8	$\boxed{\infty}$	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

We try to work out this cell

Pass 2 in matrix

0	3	8	$\boxed{\infty}$	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

1. Who can connect to this cell?

ANS: The corresponding column tell us

Remarks: we can ignore infinity (∞) and zero

Pass 2 in matrix

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

2. Which path is the smallest?

ANS: The corresponding row tell us

Path 1->2->4 = 3+1 = 4 units

Path 1->5->4 = -4+6 = 2 units

Pass 2 in matrix

0	3	8	2	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

We can see:

Path 1->5->4 is the smallest

We update the cell now

The whole process

$$D^{(1)}=\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)}=\begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(3)}=\begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(4)}=\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Consideration on All-Pair shortest path

- Matrix Method is easy to understand, but the performance is not very outstanding
- Optimization/shortcut may be needed to make it faster *(e.g. stop if no improvement, reconsidering formula)*
- Depending on situation, running Dijkstra n times is not necessary a bad choice
- There are other better methods, such as Floyd-Warshall algorithm, which make use of Dynamic Programming (*http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm)*

Exercises

Exercises

- 10171
  http://acm.uva.es/p/v101/10171.html
- 10356
  http://acm.uva.es/p/v103/10356.html
- 10285
  http://acm.uva.es/p/v102/10285.html
- 523
  http://acm.uva.es/p/v5/523.html