Graph Algorithms

CS3104

Dr. Samit Biswas, *Assistant Professor,*
Department of Computer Sc. and Technology,
Indian Institute of Engineering Science and Technology, Shibpur

Email: samit@cs.iiests.ac.in

Shortest Path Problem

- How can we find the shortest route between two points on a road map?
- Model the problem as a graph problem:
  - Road map is a weighted graph:

    vertices = cities

    edges = road segments between cities

    edge weights = road distances
  - Goal: find a shortest path between two vertices (cities)

Shortest Path Problem

- **Input:**
  - Directed graph G = (V, E)
  - Weight function w : E → **R**
- **Weight of path** p = $\langle v_0, v_1, \ldots, v_k \rangle$

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- **Shortest-path weight** from u to v:

$$\delta(u, v) = \begin{cases} \min\ w(p) : u \overset{p}{\rightsquigarrow} v & \text{if there exists a path from u to v} \\ \infty & \text{otherwise} \end{cases}$$

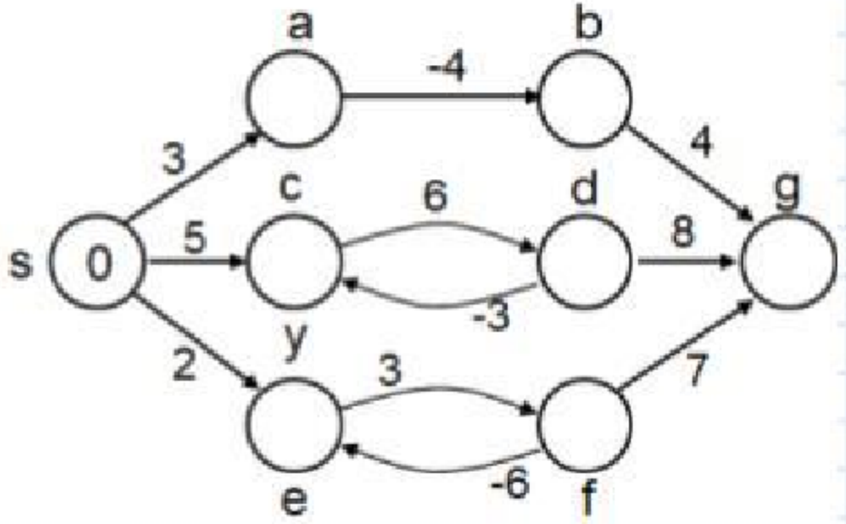
- **Note:** there might be multiple shortest paths from u to v

Variants of Shortest Path

- **Single-source shortest paths**
  - G = (V, E) ⇒ find a shortest path from a given source vertex s to each vertex v ∈ V
- **Single-destination shortest paths**
  - Find a shortest path to a given destination vertex **t** from each vertex v
  - Reversing the direction of each edge ⇒ single-source
- **Single-pair shortest path**
  - Find a shortest path from u to v for given vertices u and v
- **All-pairs shortest-paths**
  - Find a shortest path from u to v for every pair of vertices u and v

Negative-Weight Edges

- Negative-weight edges may form negative-weight cycles
- If such cycles are reachable from the source, then $\delta(s, v)$ is not properly defined!
  - Keep going around the cycle, and get $w(s, v) = -\infty$ for all v on the cycle



Negative-weight edges

$s \to a$; only one path

$\delta(s,a) = w(s,a) = 3$

$s \to b$ only one path

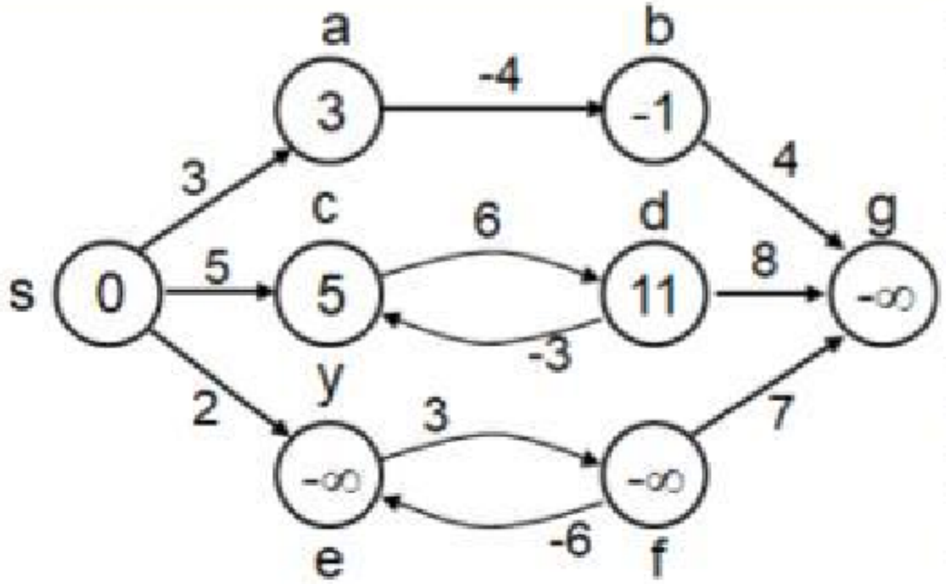
$\delta(s,b) = w(s,a) + w(a,b) = 3 + (-4) = -1$

$s \to c$: infinitely many paths

$\{s,c\}, \{s,c,d,c\}, \{s,c,d,c,d,c\}, \ldots$

cycle weight $= 6 - 3 = 3$

positive



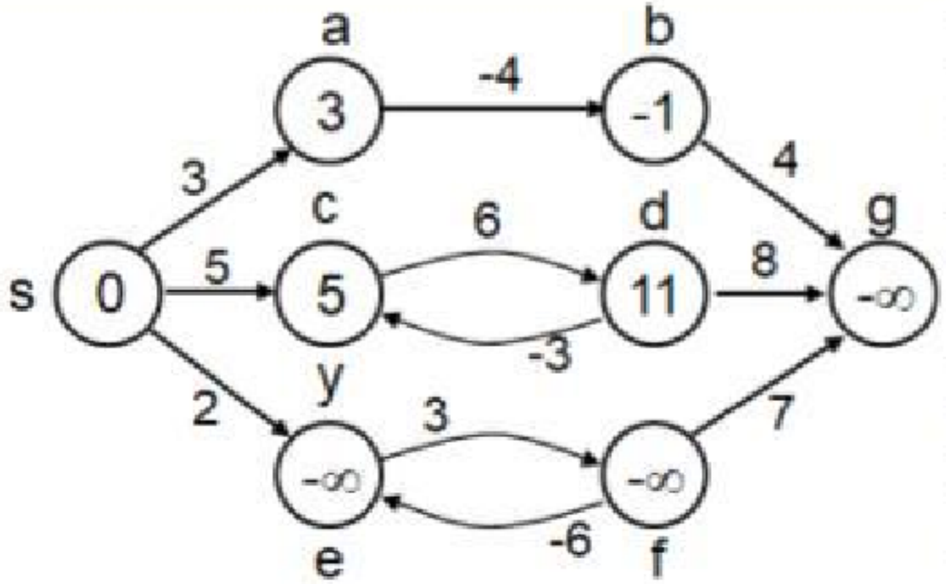
Negative-weight edges

$s \to e$, infinitely many paths

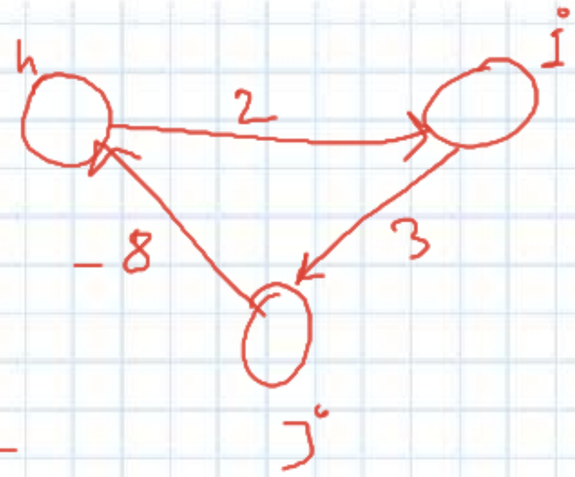
$\{s, e\}, \{s, e, f, e\}, \{s, e, f, e, f, e\} \ldots$

cycle $\{e, f, e\}$, weight $= (3 - 6) = -3$ negative

$\delta(s, e) = -\infty$; no shortest path exists bet^{n} s & e.



Similarly, $\delta(s, f) = -\infty$



$\delta(s, h) = \infty$

h, i, j is not reachable from s.

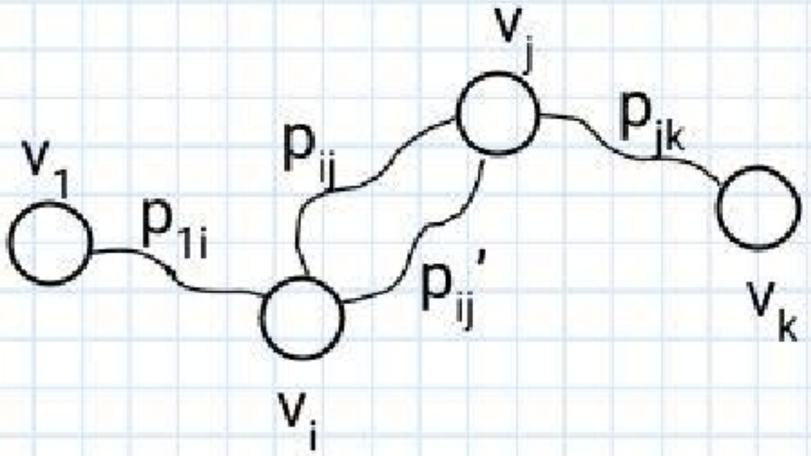
Cycles

Can shortest paths contain cycles?

- Negative-weight cycles No!
  - Shortest path is not well defined
- Positive-weight cycles: No!
  - By removing the cycle, we can get a shorter path
- Zero-weight cycles
  - No reason to use them
  - Can remove them to obtain a path with same weight

Optimal Substructure Theorem

- Given:
  - A weighted, directed graph G = (V, E)
  - A weight function w: E → R,
  - A shortest path $p = \langle v_1, v_2, \ldots, v_k \rangle$ from v_1 to v_k
  - A subpath of p: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \le i \le j \le k$



- Then: p_{ij} is a shortest path from v_i to v_j

Proof:

$p = v_1 \overset{p_{1i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$

$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$

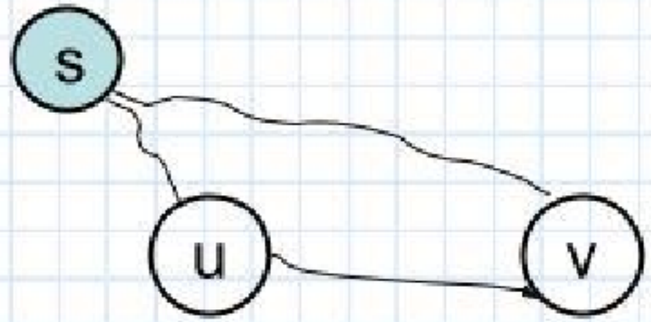
Assume $\exists\ p_{ij}'$ from v_i to v_j with $w(p_{ij}') < w(p_{ij})$

$\Rightarrow w(p') = w(p_{1i}) + w(p_{ij}') + w(p_{jk}) < w(p)$ contradiction!

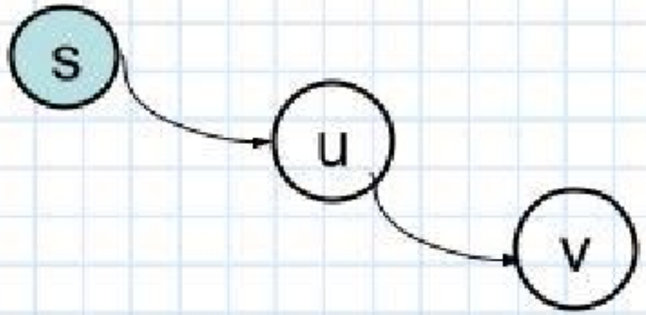
Triangle Inequality

- For all $(u, v) \in E$, we have:

  $\delta(s, v) \leq \delta(s, u) + \delta(u, v)$



  - If u is on the shortest path to v we have the equality sign



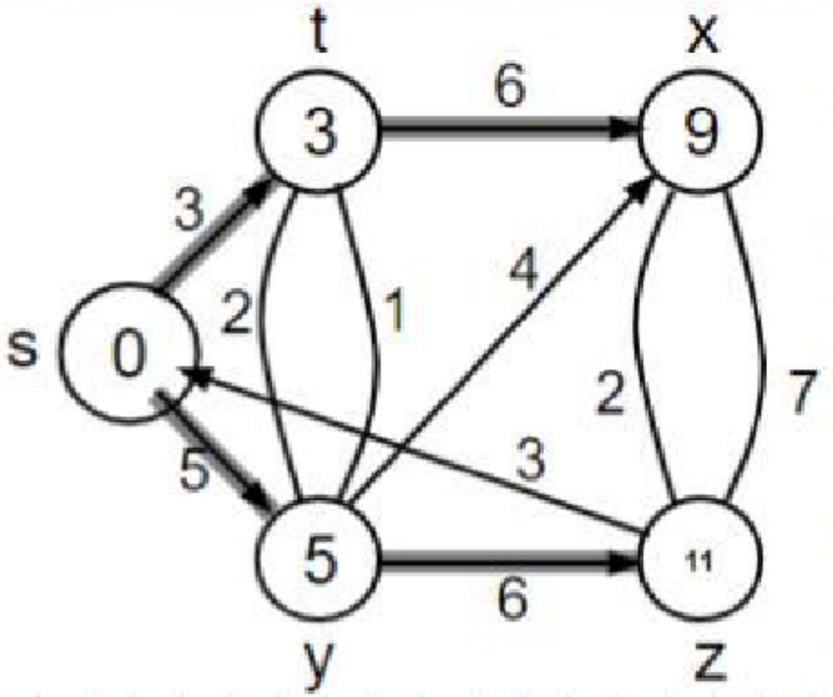
Shortest Path Algorithms

- Bellman-Ford algorithm
  - Negative weights are allowed
  - Negative cycles reachable from the source are not allowed.
- Dijkstra's algorithm
  - Negative weights are not allowed
- Operations common in both algorithms:
  - Initialization
  - Relaxation

Shortest Path Notation

For each vertex $v \in V$:

- $\delta(s, v)$: **shortest-path weight**
- $d[v]$: shortest-path weight **estimate**
  - Initially, $d[v]=\infty$
  - $d[v] \geq \delta(s,v)$ as algorithm progresses
- $\pi[v]$ = **predecessor** of v on a shortest path from s
  - If no predecessor, $\pi[v]$ = NIL
  - π induces a tree—**shortest-path tree**



Initialization

Algorithm: INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$

- All the shortest-paths algorithms start with **INITIALIZE-SINGLE-SOURCE**

Relaxation Step

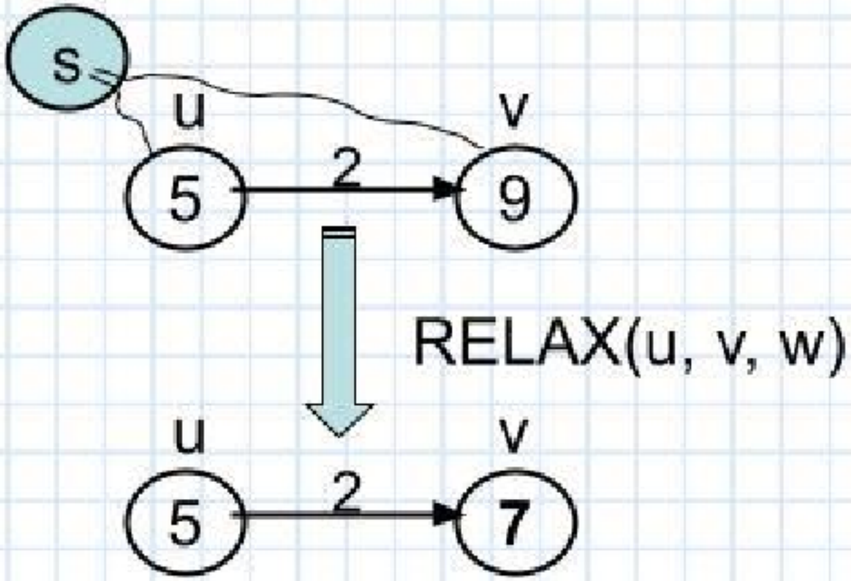
- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u

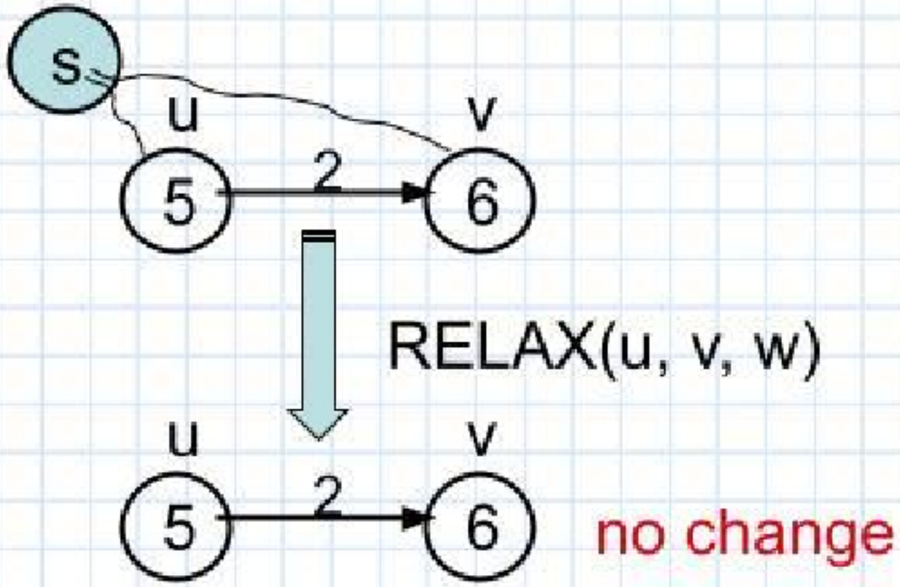
  If d[v] > d[u] + w(u, v)

  we can improve the shortest path to v

  ⇒ d[v]=d[u]+w(u,v)

  ⇒ π[v] ← u





Bellman-Ford Algorithm

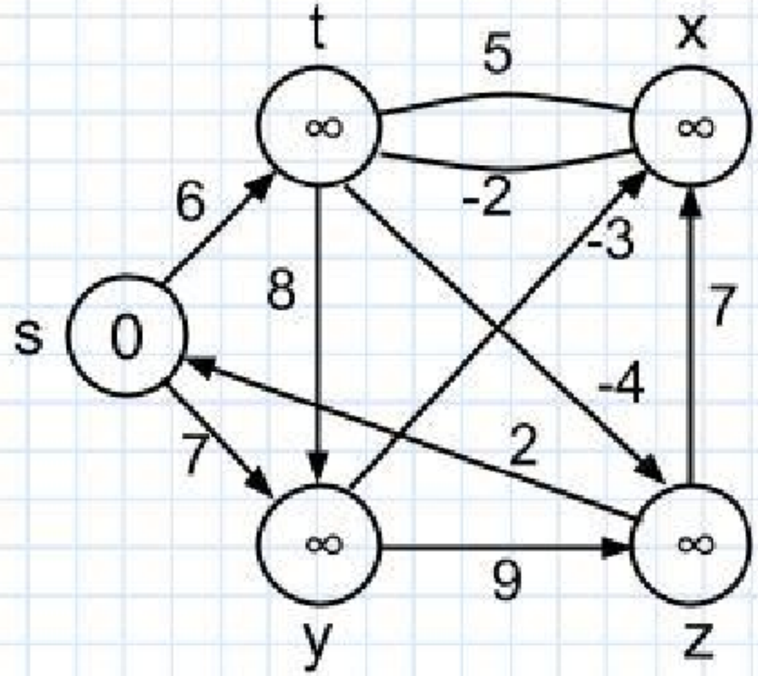
- Single-source shortest path problem
  - Computes $\delta(s, v)$ and $\pi[v]$ for all $v \in V$

- Allows negative edge weights - can detect negative cycles.
  - Returns TRUE if no negative-weight cycles are reachable from the source s
  - Returns FALSE otherwise ⇒ no solution exists

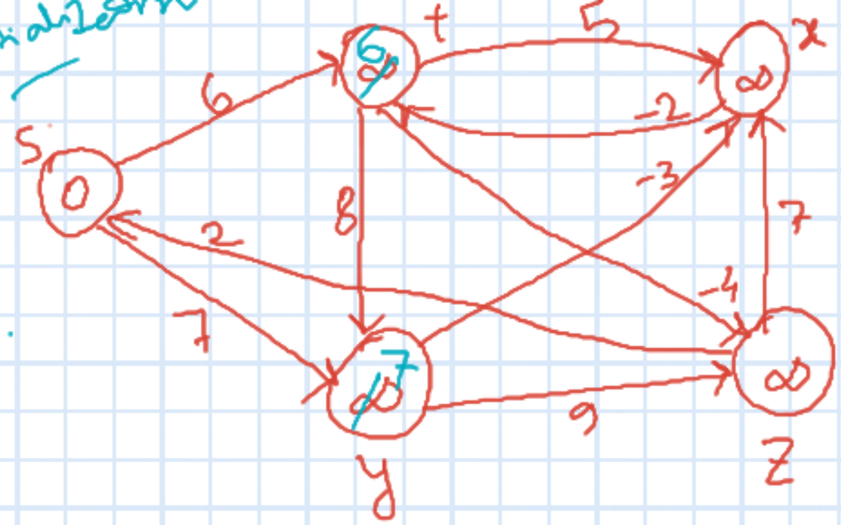
Bellman-Ford Algorithm (Contd ..)

- **Idea:**
  - Each edge is relaxed |V−1| times by making |V-1| passes over the whole edge set.
  - To make sure that each edge is relaxed exactly |V − 1| times, it puts the edges in an unordered list and goes over the list |V − 1| times.

(t, x), (t, y), (t, z), (x, t), (y, x),
(y, z), (z, x), (z, s), (s, t), (s, y)



Initialization

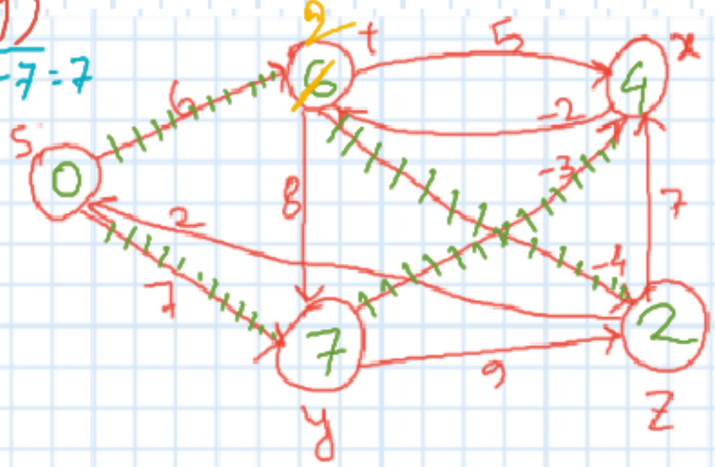


(t,x) (t,y) (t,z) (x,t) (y,x) (y,7)

(z,x) , (z,s) , (s,t) , (s,y)

0+6=6 0+7=7

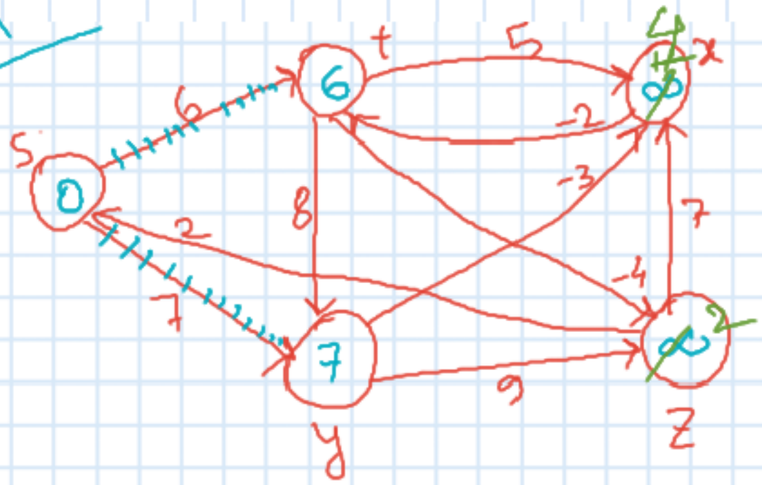
Pass-2



(t,x) (t,y) (t,z) (x,t) (y,x) (y,7)

(z,x) , (z,s) , (s,t) , (s,y)

Pass

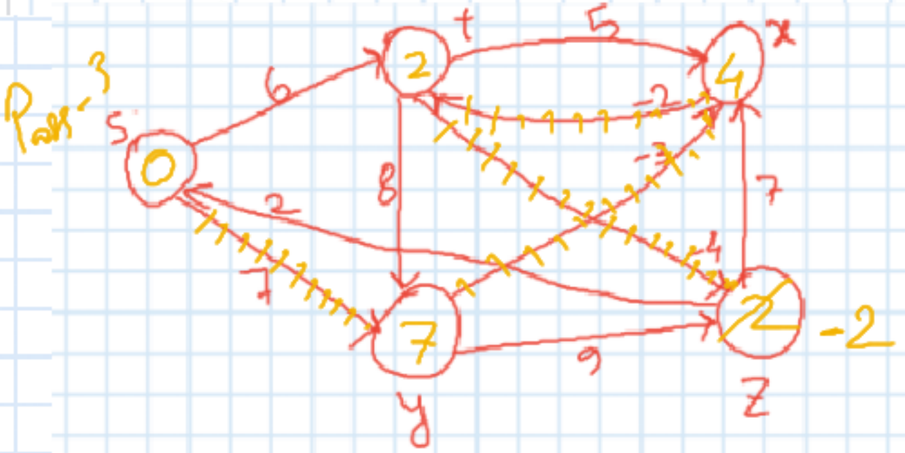


(t,x) (t,y) (t,z) (x,t) (y,x) (y,7)

6-4=2

(z,x) , (z,s) , (s,t) , (s,y)

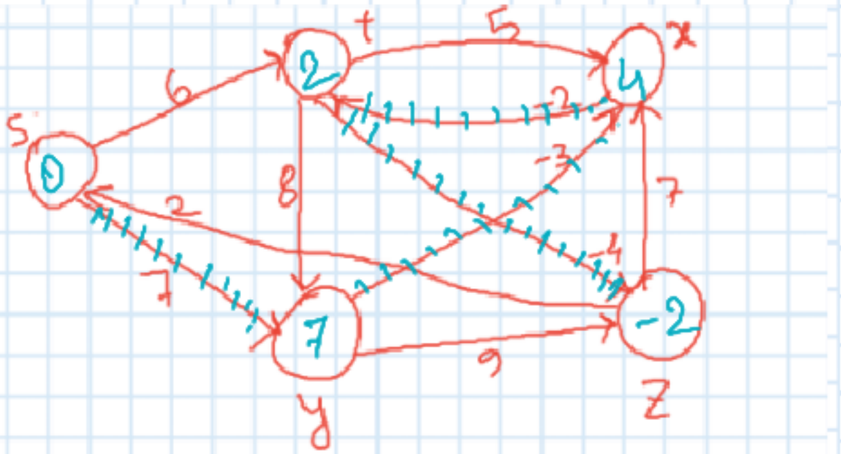
Pass-3



(t,x) (t,y) (t,z) (x,t) (y,x) (y,7)

(z,x) , (z,s) , (s,t) , (s,y)

Рис-4



(t,x) (t,y) (t,z) (x,t) (y,x) (y,7)

(z,x), (z,s), (s,t), (s,y)

BELLMAN-FORD (V, E, w, s)

```
1.  INITIALIZE-SINGLE-SOURCE(V, s)
2.  for i ← 1 to |V| - 1
3.      do for each edge (u, v) ∈ E
4.          do RELAX(u, v, w)
5.  for each edge (u, v) ∈ E
6.      do if d[v] > d[u] + w(u, v)
7.          then return FALSE
8.  return TRUE
```

$\Theta(V)$

$O(V)$

$O(E)$

$O(VE)$

$O(E)$

Total Running time = $O(V + VE + E)$ = $O(VE)$