CS & IT ENGINEERING

Algorithms

Algorithms

Lecture No. 12

By- Ravindrababu Ravula Sir

Recap of Previous Lecture

Topic Dijkstra Algorithm

Topics to be Covered

Topic Bellman ford Algorithm

Topic Shortest path in DAGS

Topic: Bellman ford Algorithm

Bellman ford Introduction: (single source shortest path)

- Can find out whether a graph is having negative weight cycle or not.

Topic: Bellman ford Algorithm

Bellman ford Introduction:

- Can find out whether a graph is having negative weight cycle or not.

Topic: Bellman ford Algorithm

Bellman ford Introduction:

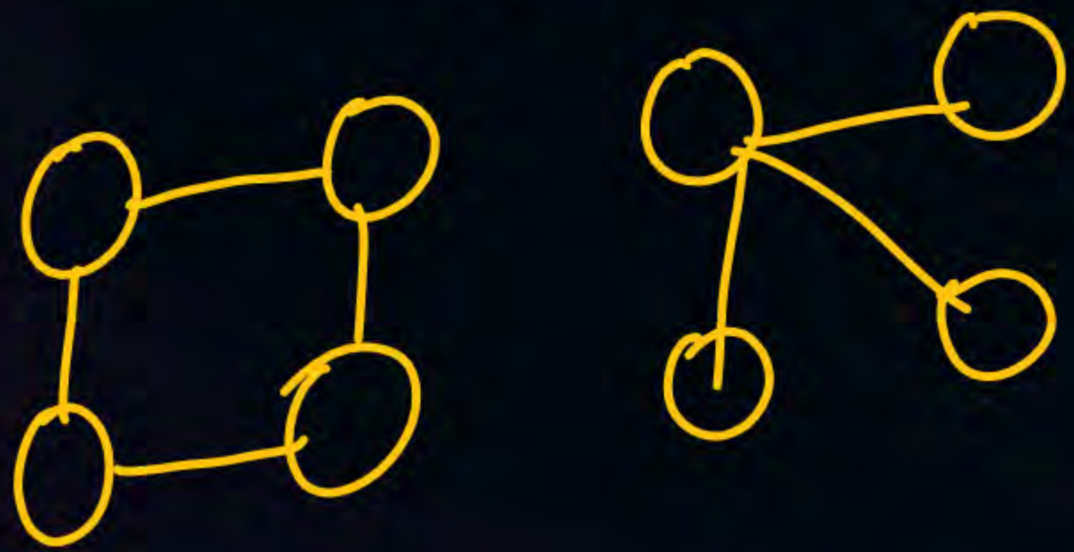
- Can find out whether a graph is having negative weight cycle or not.

- Bellman ford algorithm is slower than Dijkstra ✓

Topic: Bellman ford Algorithm

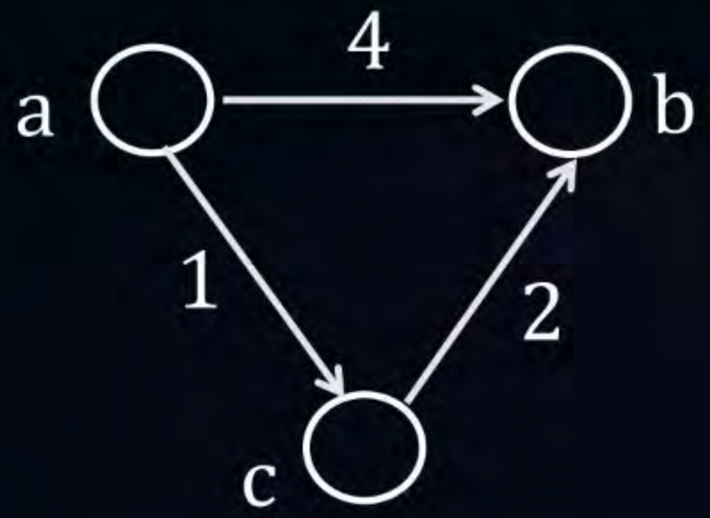
Bellman ford works on this rule:

- Shortest path between 2 nodes in the graph will not contain more than (n–1) edges if there are n vertices



Topic: Bellman ford Algorithm

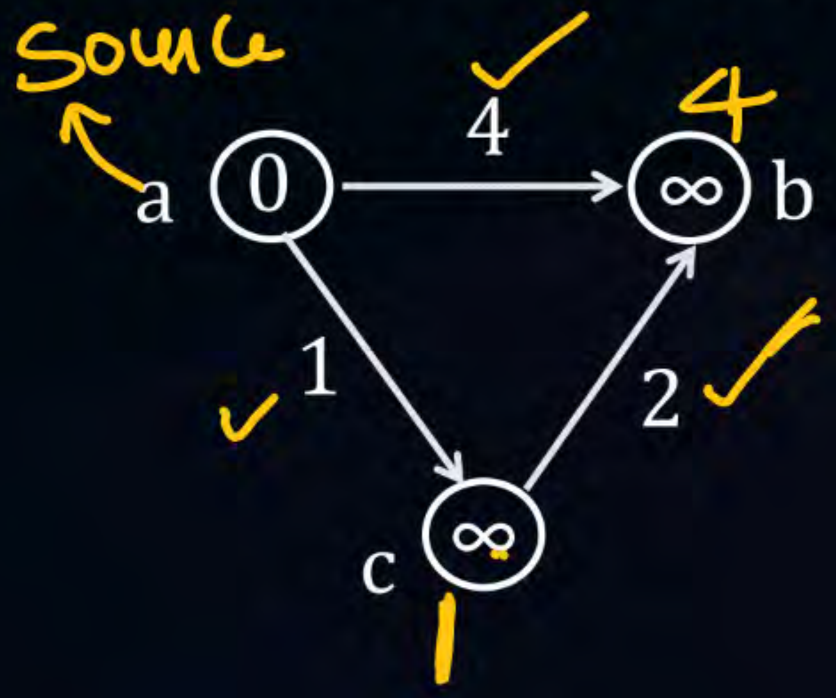
The node contains n = 3 nodes, so edges are relaxed 2 times



Topic: Bellman ford Algorithm

3 2 times

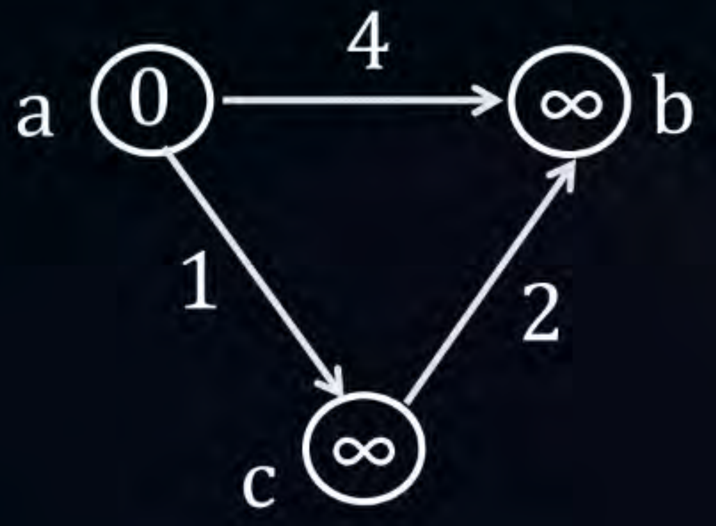
The node contains n = 3 nodes, so edges are relaxed 2 times

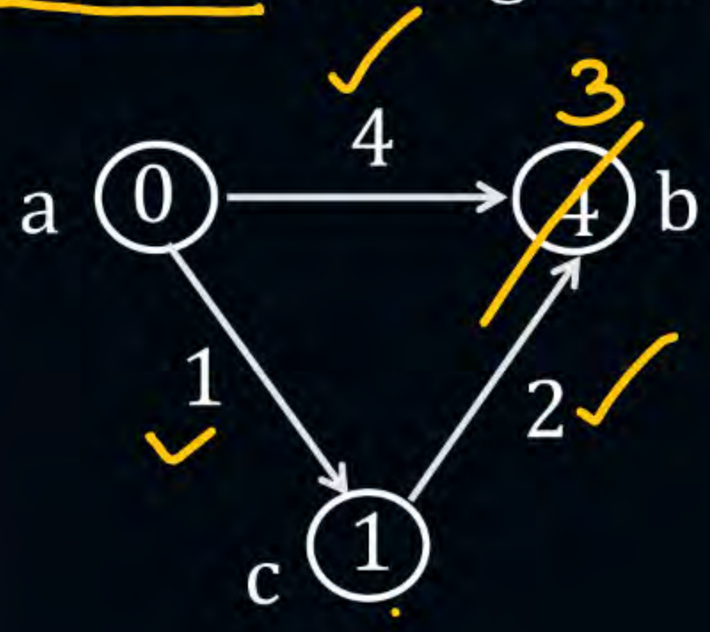


Topic: Bellman ford Algorithm

2 times → all edges

The node contains n = 3 nodes, so edges are relaxed 2 times





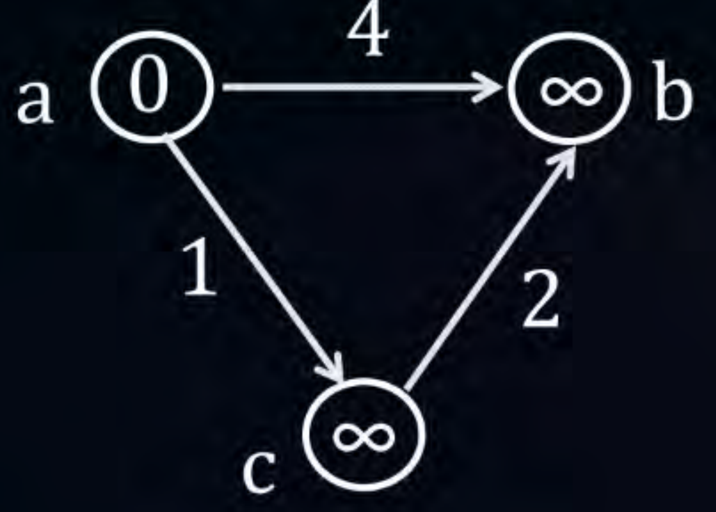
1st time

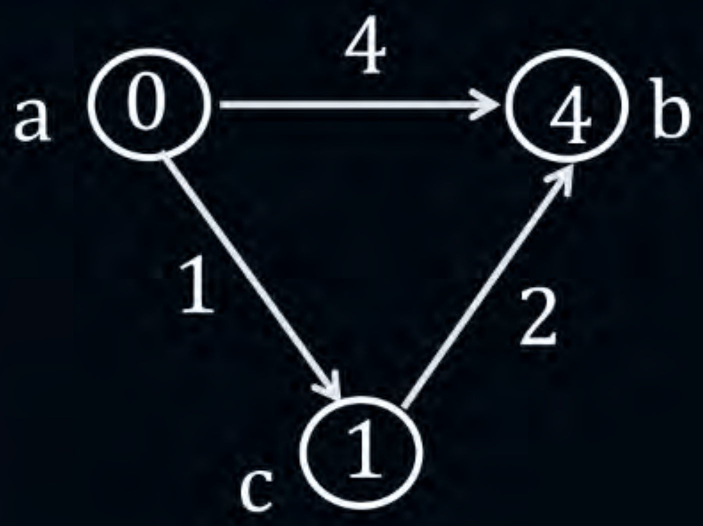
↓

Lenght of shortest path with almost one edge

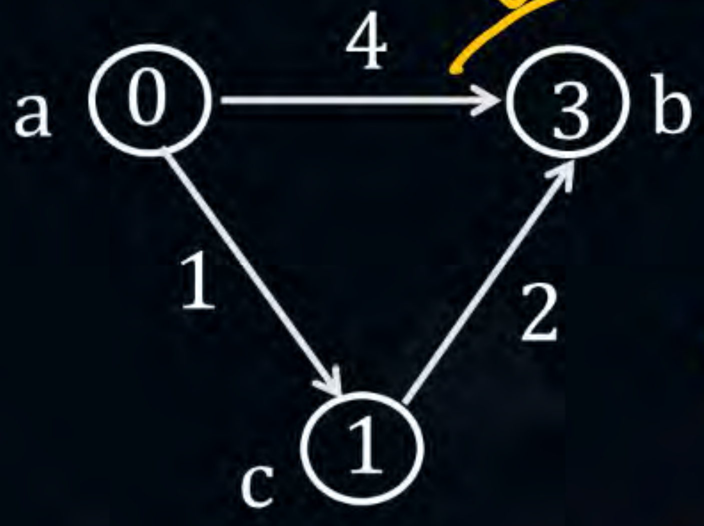
Topic: Bellman ford Algorithm

The node contains n = 3 nodes, so edges are relaxed 2 times $(n-1)^{th}$ $(n)^{th}$





1st time



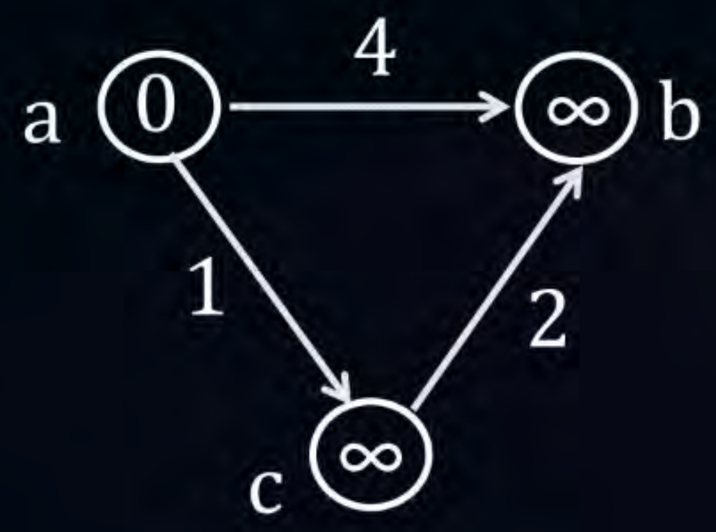
2nd time = 3rd

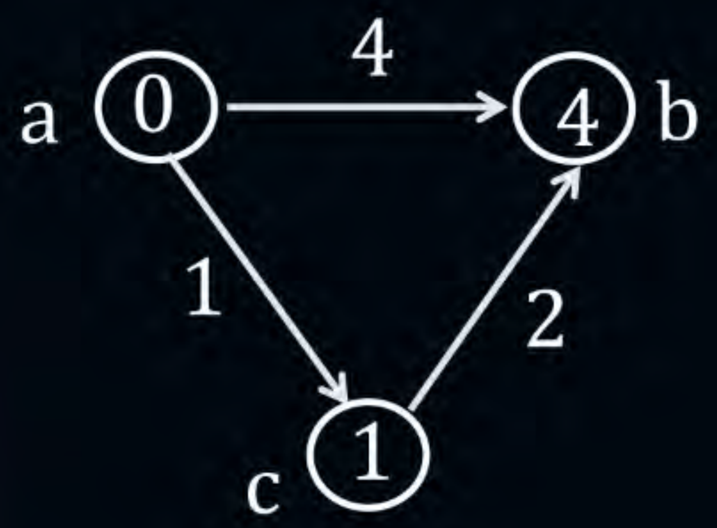
⇓

lenght of shortest path with atmost 2 edges

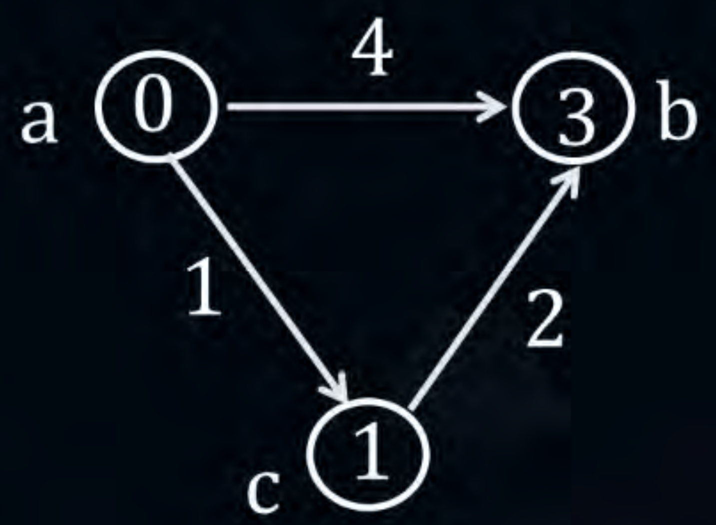
Topic: Bellman ford Algorithm

The node contains n = 3 nodes, so edges are relaxed 2 times





1st time



2nd time

If the nth and (n-1)th iterations give same values then there are no negative weight ~~edge~~ cycle present and the solution is correct else discarded ✓

Topic: Bellman ford Algorithm

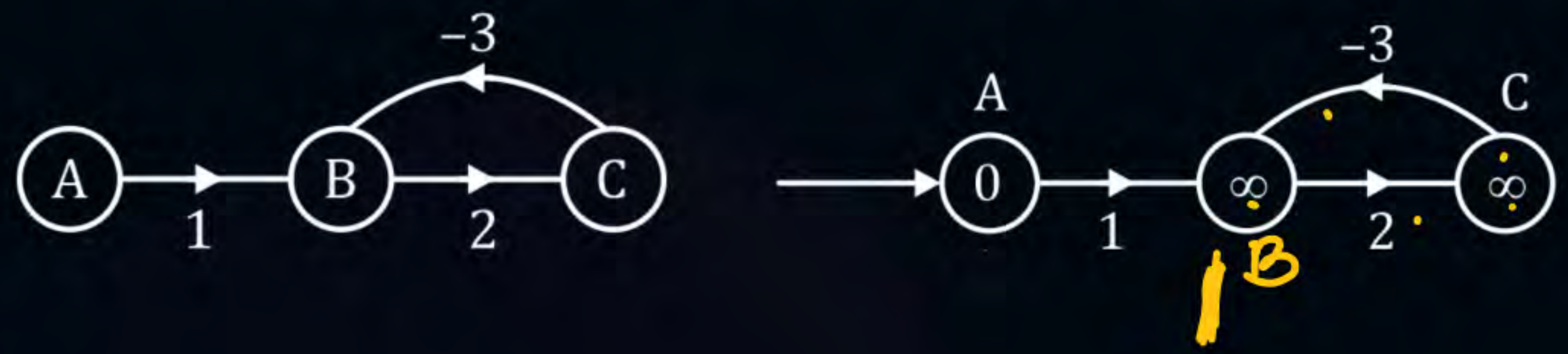
Example:-

Bellman ford find out whether there is negative weight cycle.

Topic: Bellman ford Algorithm

Example:-

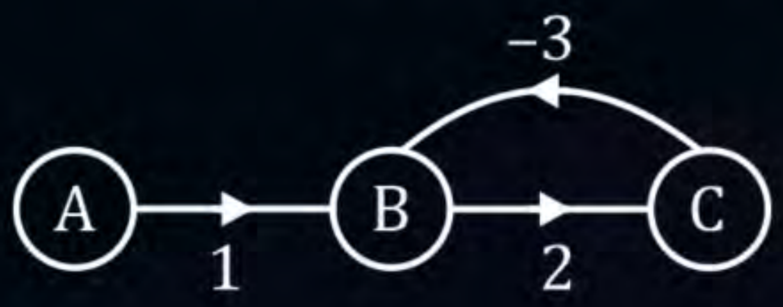
Bellman ford find out whether there is negative weight cycle.

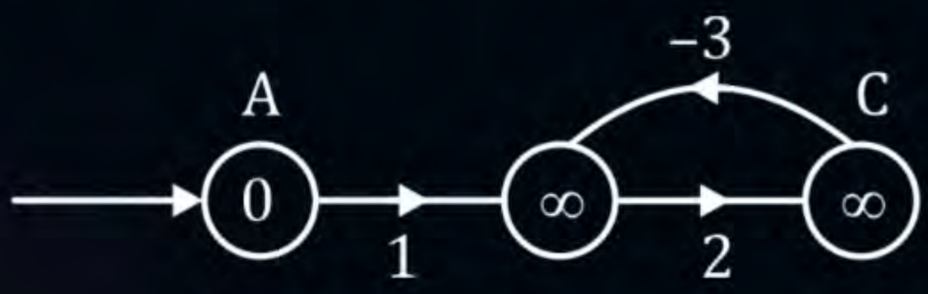


Topic: Bellman ford Algorithm

Example:-

Bellman ford find out whether there is negative weight cycle.



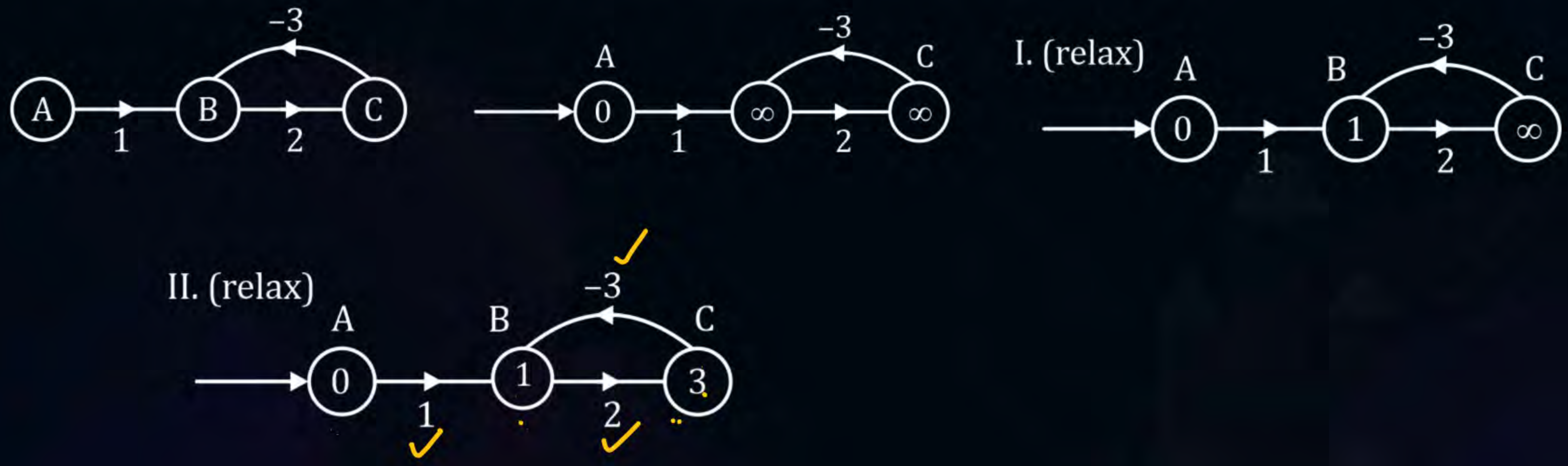


I. (relax)

Topic: Bellman ford Algorithm

Example:-

Bellman ford find out whether there is negative weight cycle.

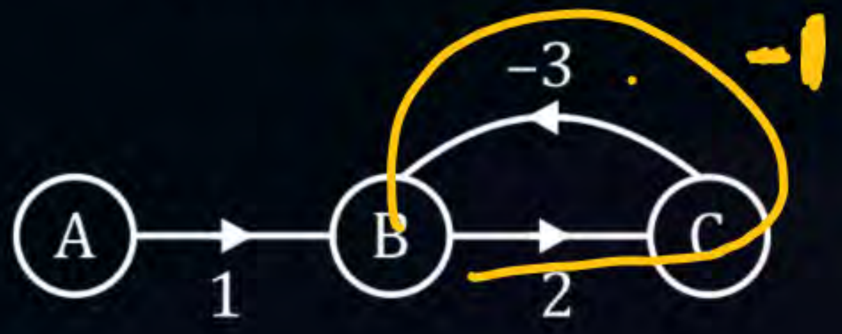


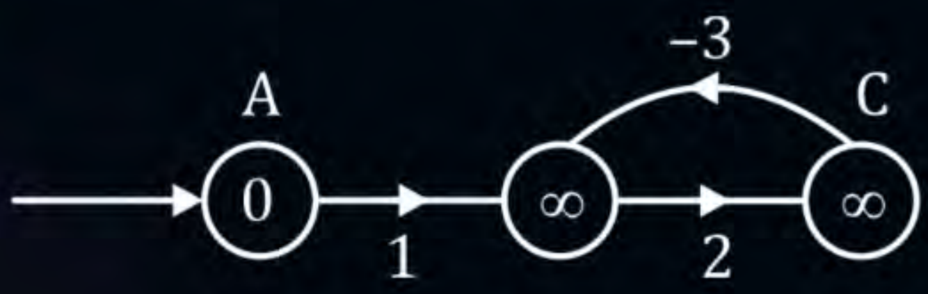
Topic: Bellman ford Algorithm

nodes = 3   2 + 1

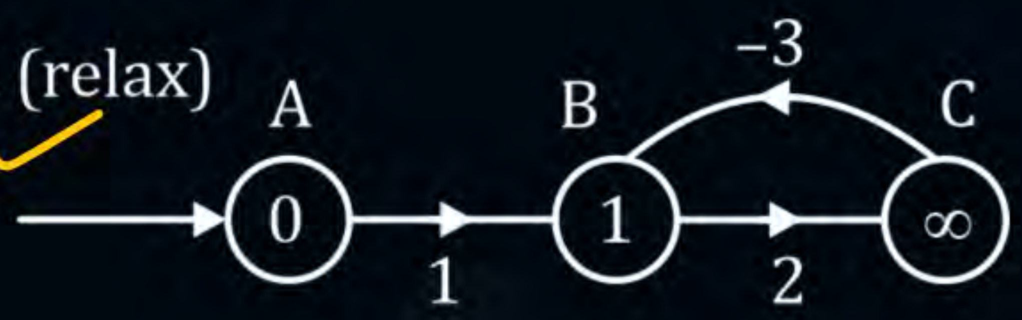
Example:-

Bellman ford find out whether there is negative weight cycle.

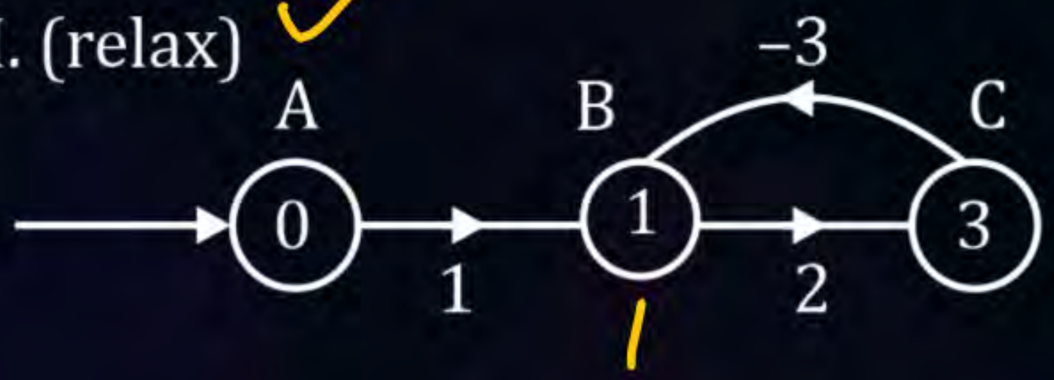




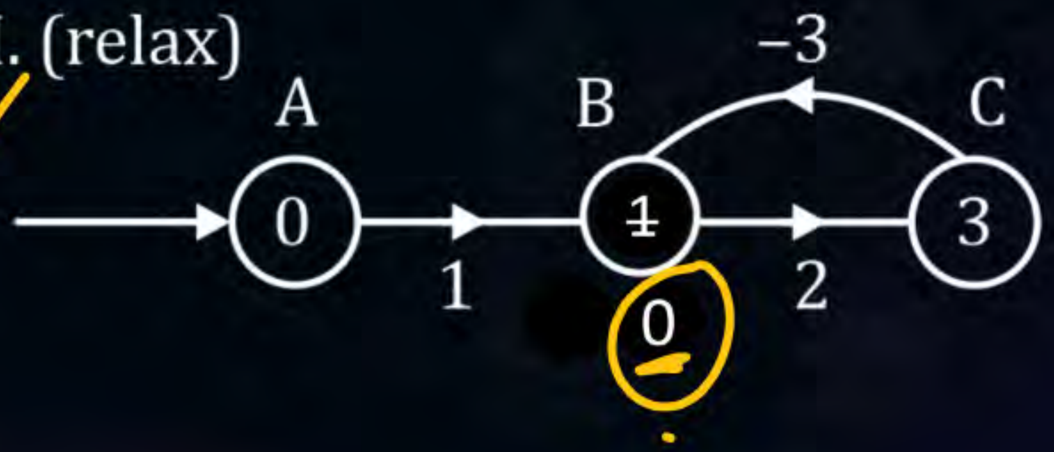
I. (relax)



II. (relax)



III. (relax)



Topic: Bellman ford Algorithm

Relaxing time = $O(V) \times O(E)$

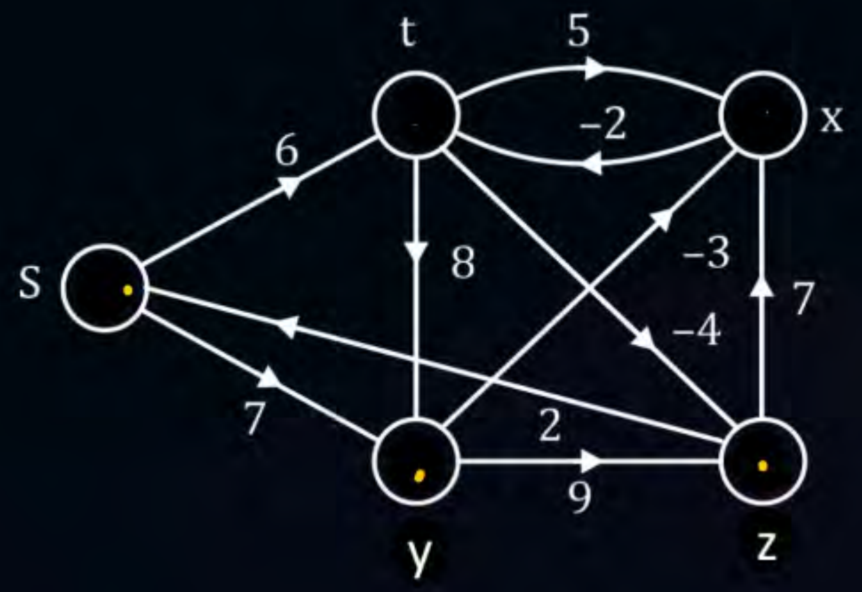
$= O(VE)$

Time complexity = $O(VE)$.

Topic: : Bellman ford Algorithm

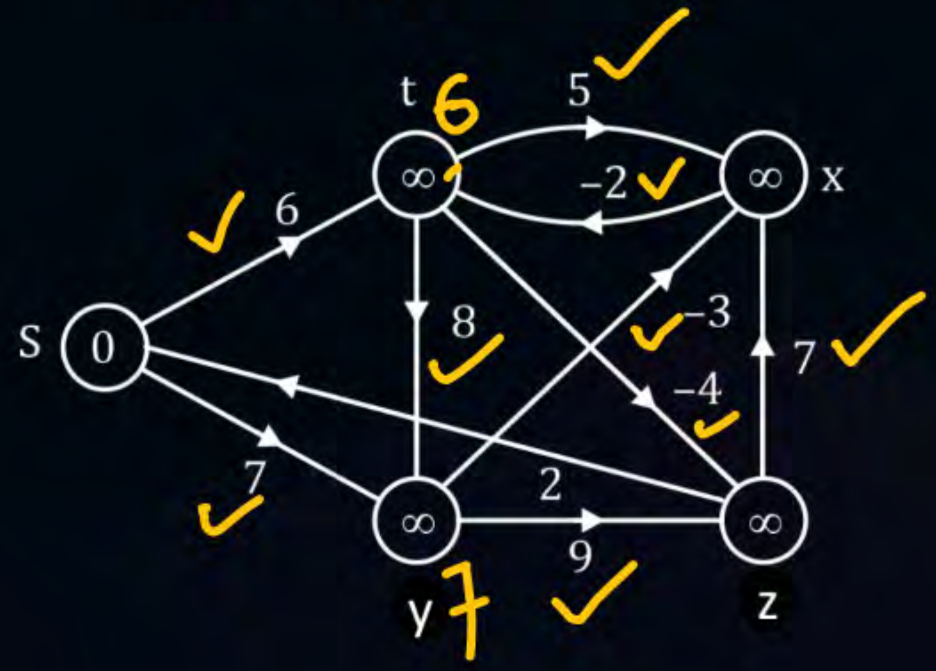
Example:

4 times relax all edges



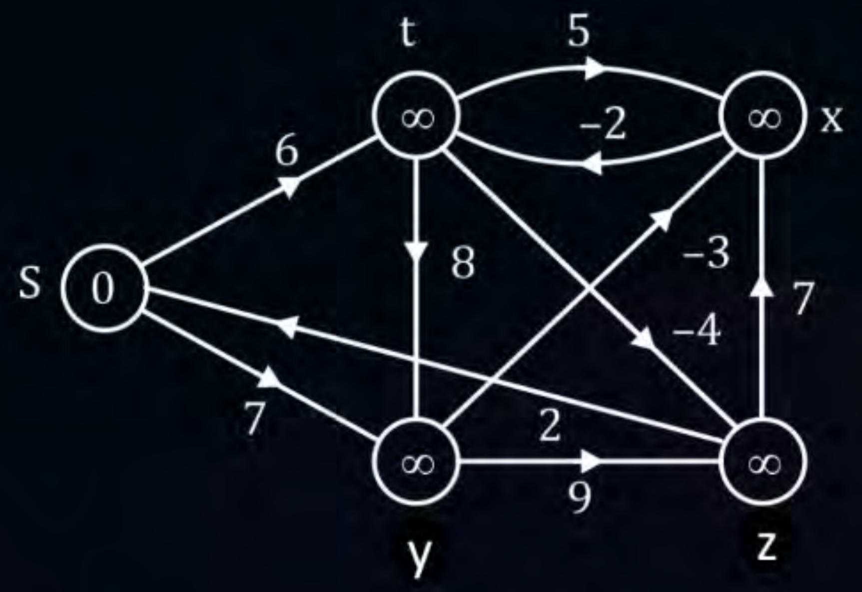
Topic: : Bellman ford Algorithm

Example:

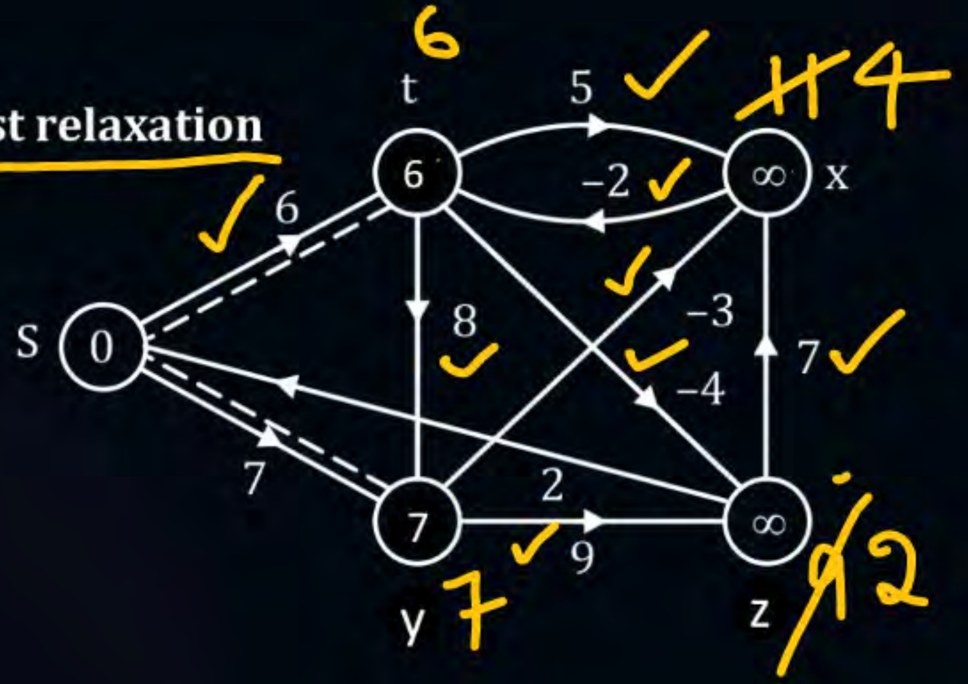


Topic: : Bellman ford Algorithm

Example:



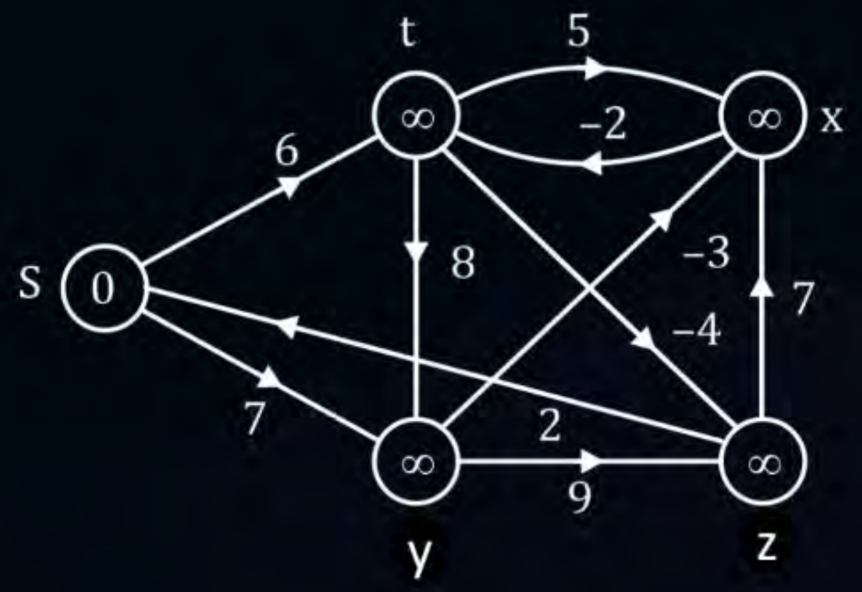
1st relaxation

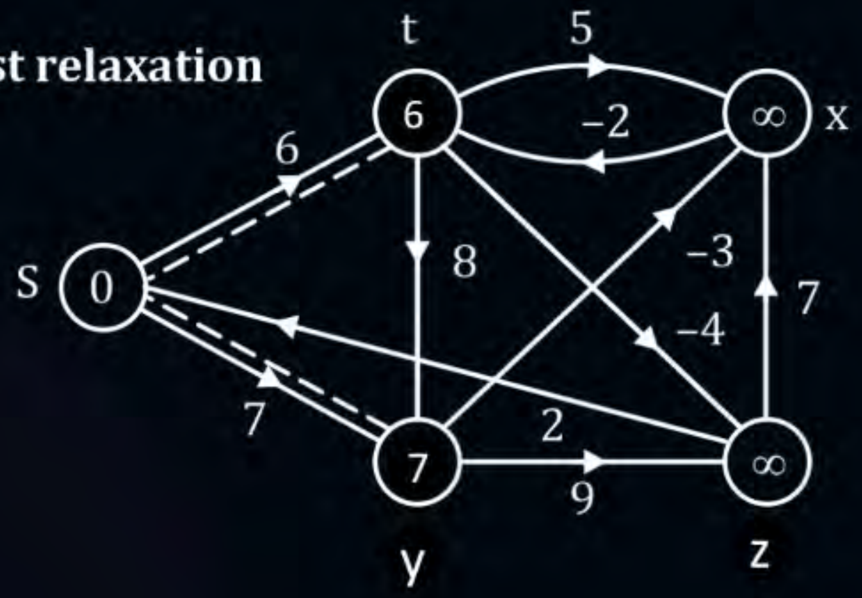


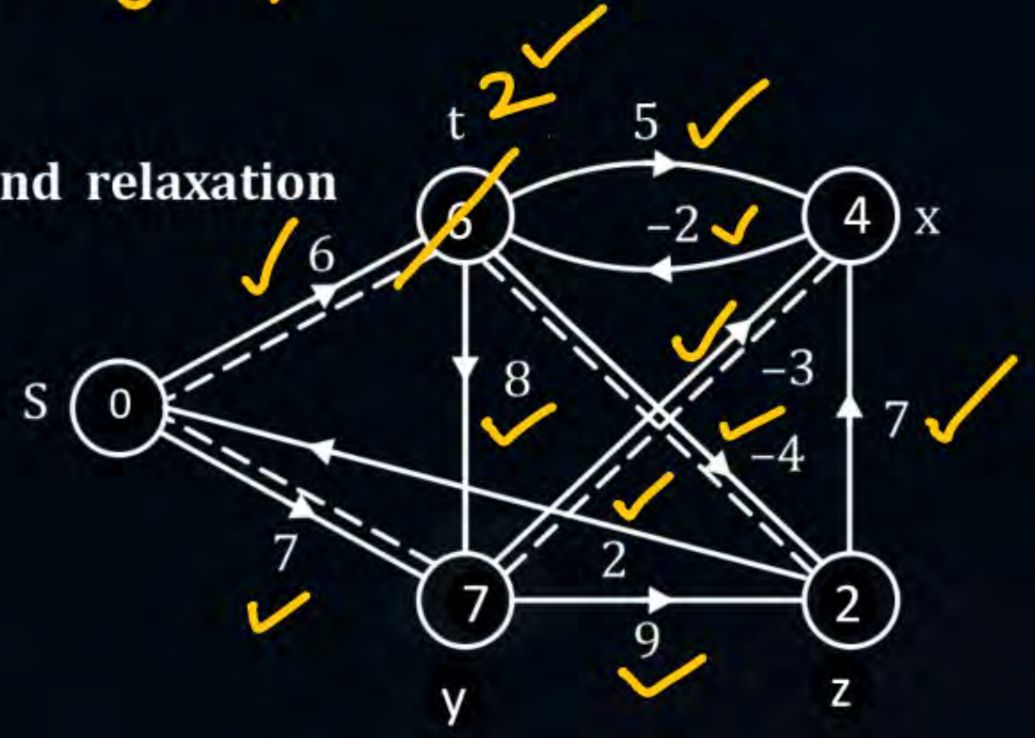
Topic: : Bellman ford Algorithm

Example:

4 times + 1 -ve weight cycle

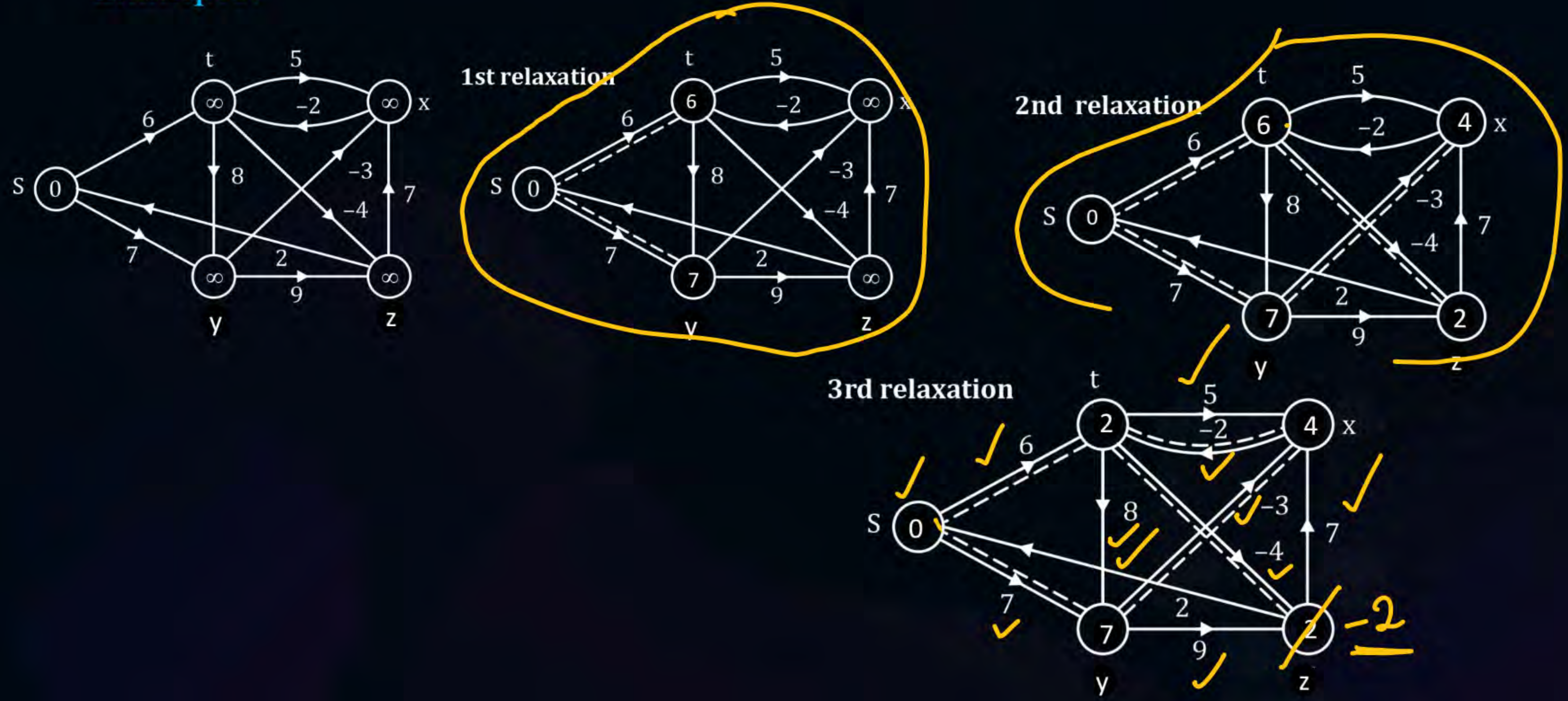




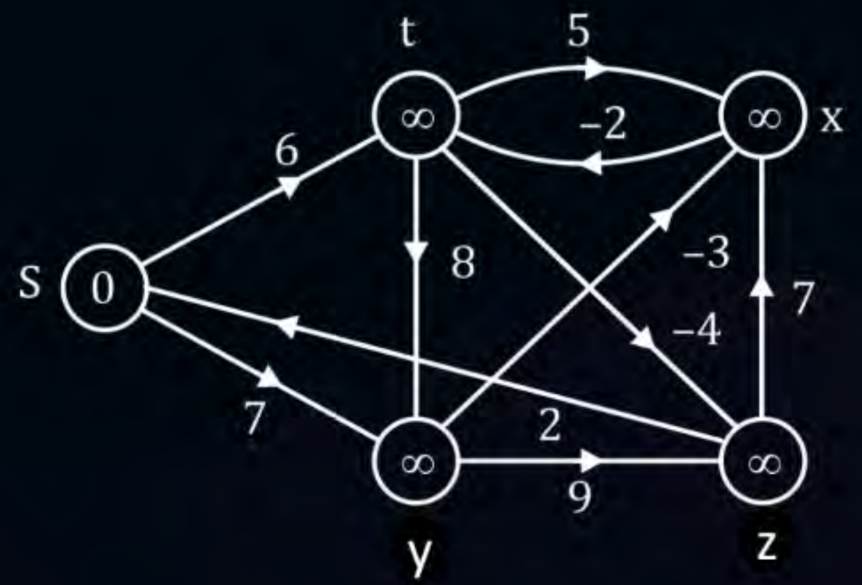


Topic: : Bellman ford Algorithm

Example:

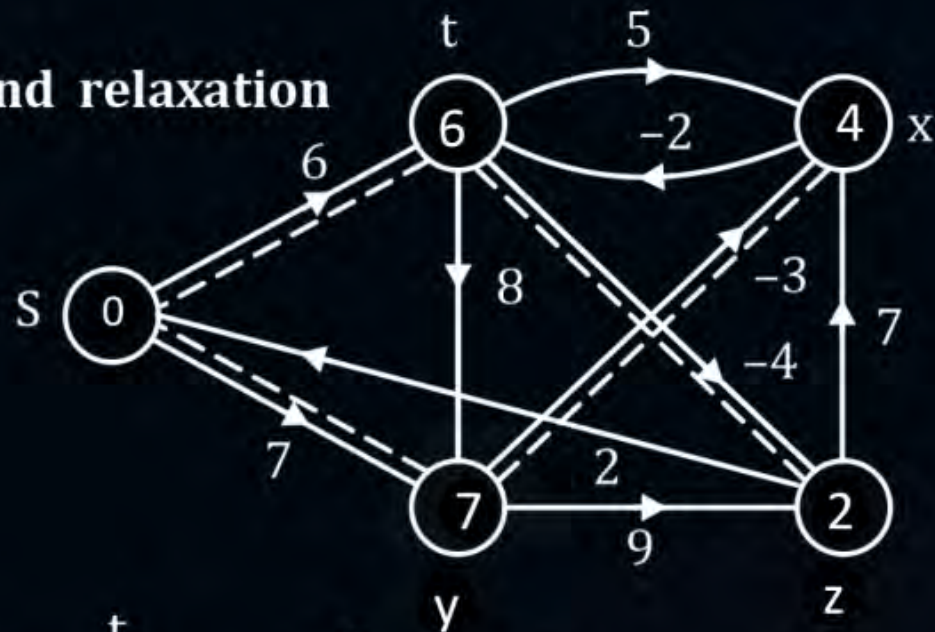


Example:

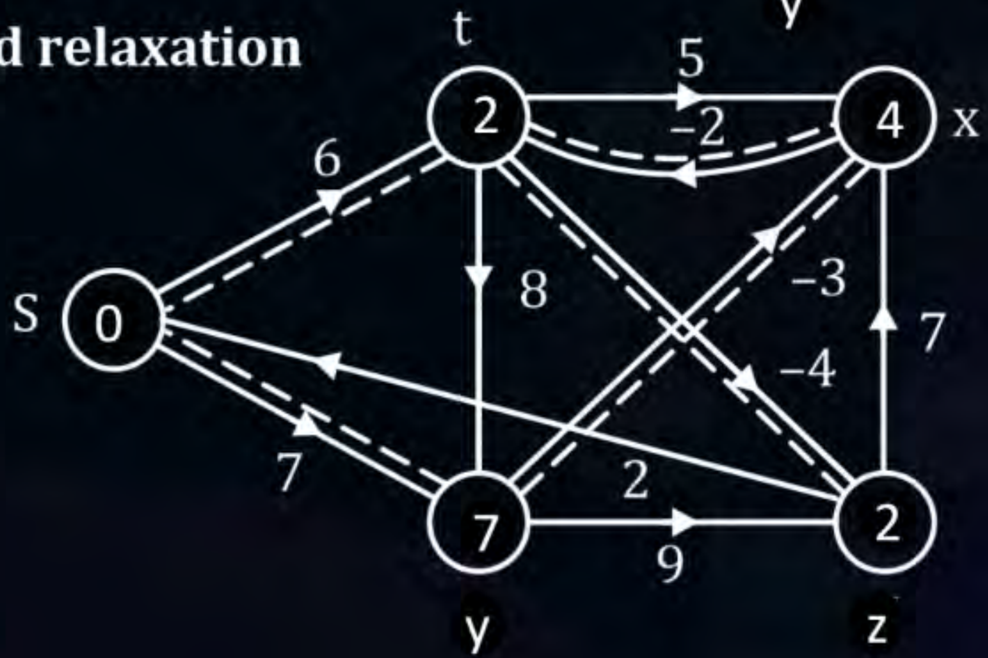


1st relaxation

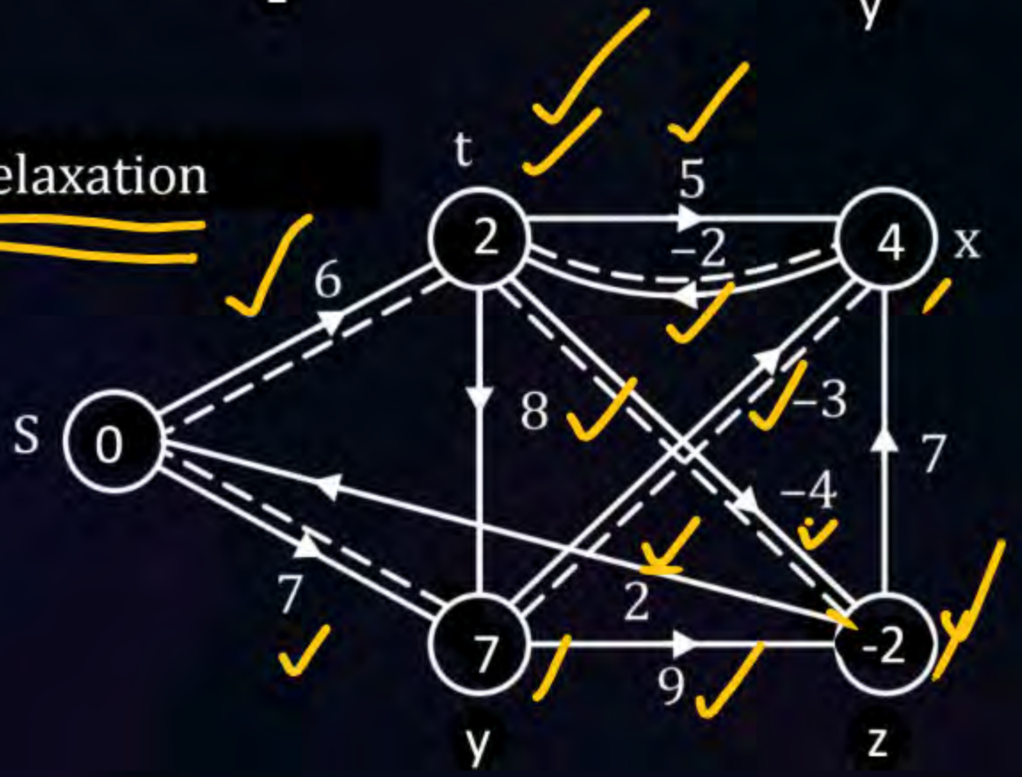
2nd relaxation



3rd relaxation



4th relaxation



Topic: : Bellman ford Algorithm

Bellman-Ford Algorithm

```
BELLMAN-FORD(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)          O(V)
    for i = 1 to |G.V| − 1
        for each edge (u, v) ∈ G.E
            RELAX(u, v, w)                  → O(1)
    for each edge (u, v) ∈ G.E
        if v.d > u.d + w(u, v)
            return FALSE
    return TRUE
```


Time complexity = $O(V) + O(VE) + O(E)$

$= O(VE)$

Shortest paths in DAG (Directed acyclic graph):

∞

Topic: : Shortest path in DAGS

Shortest paths in DAG (Directed acyclic graph):

- DAG is a graph with no cycles

∞

Topic: : Shortest path in DAGS

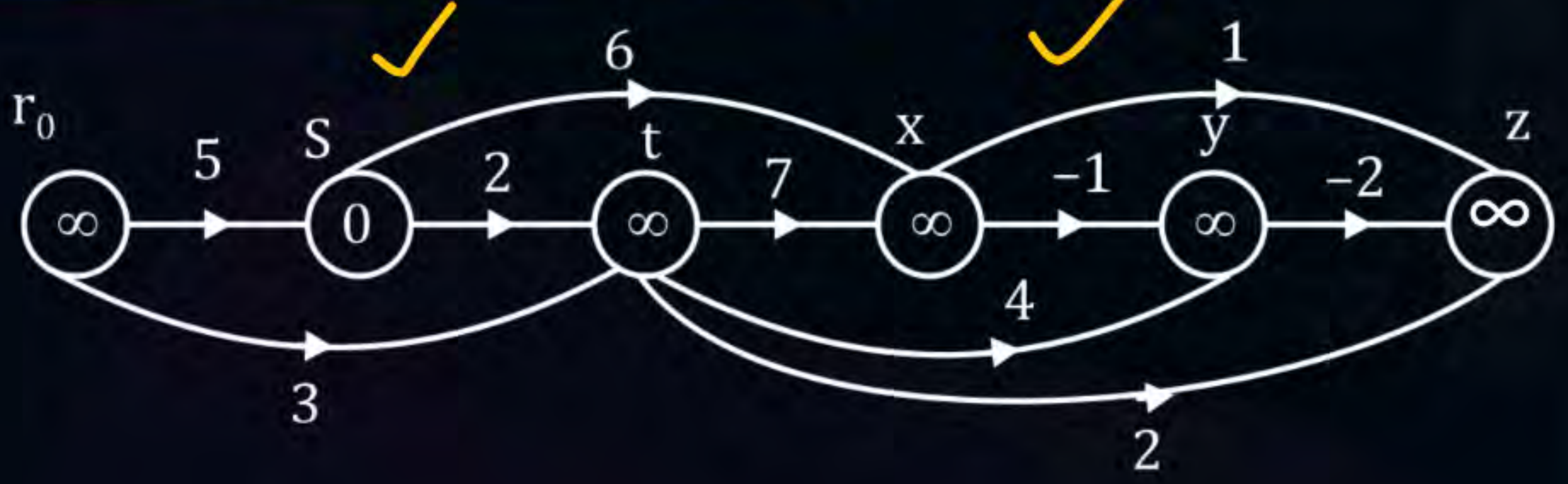
Shortest paths in DAG (Directed acyclic graph):

- DAG is a graph with no cycles
- O(V+E) time takes to in order to put a DAG in topological Order

∞

Shortest paths in DAG (Directed acyclic graph):

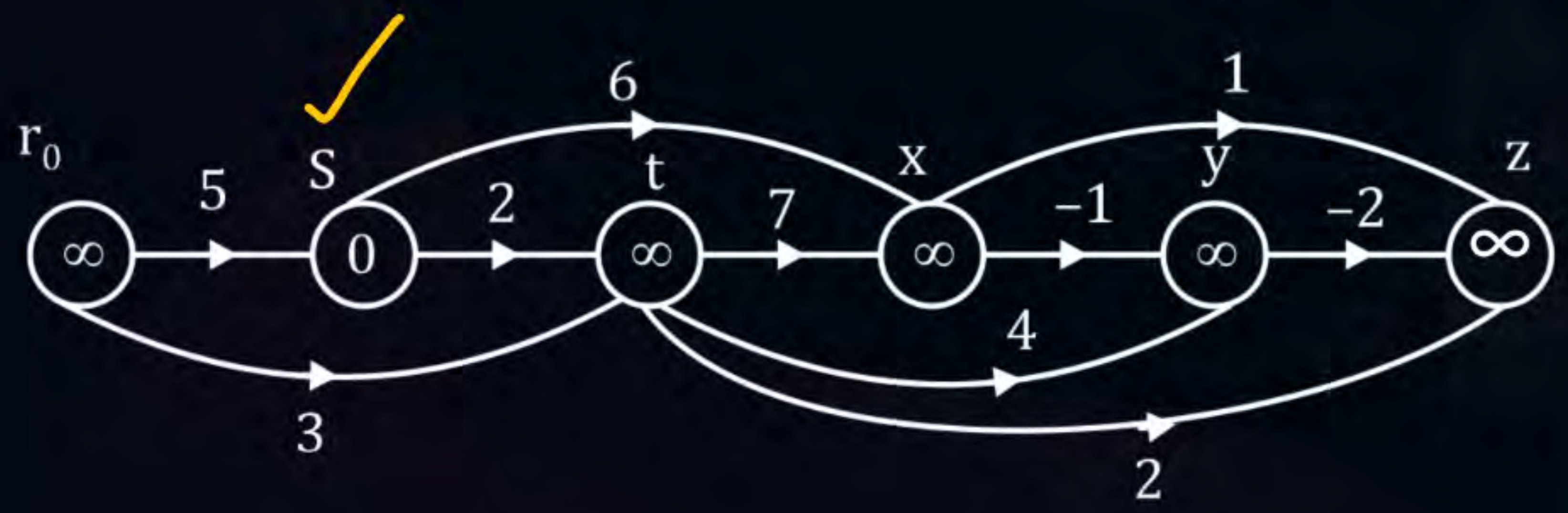
- DAG is a graph with no cycles
- O(V+E) time takes to in order to put a DAG in topological Order
- Take the vertices one by one in topological Order and then try to relax outgoing edges outgoing from them.



Topic: : Shortest path in DAGS

Topological order:

- Linear ordering of vertices in a DAG
- rstxyz
- Time complexity-O(V+E)

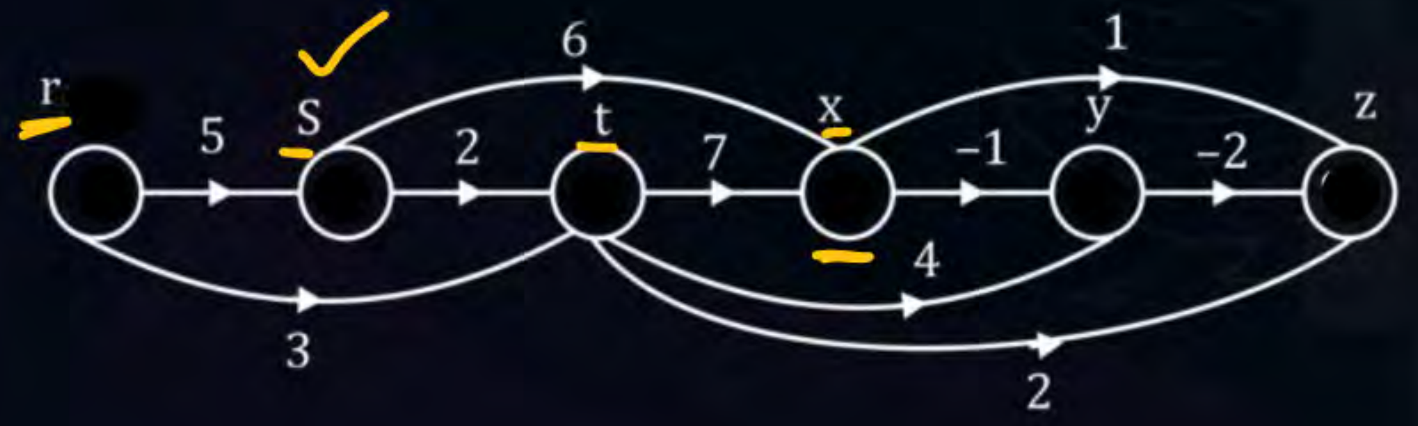


Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz ✓

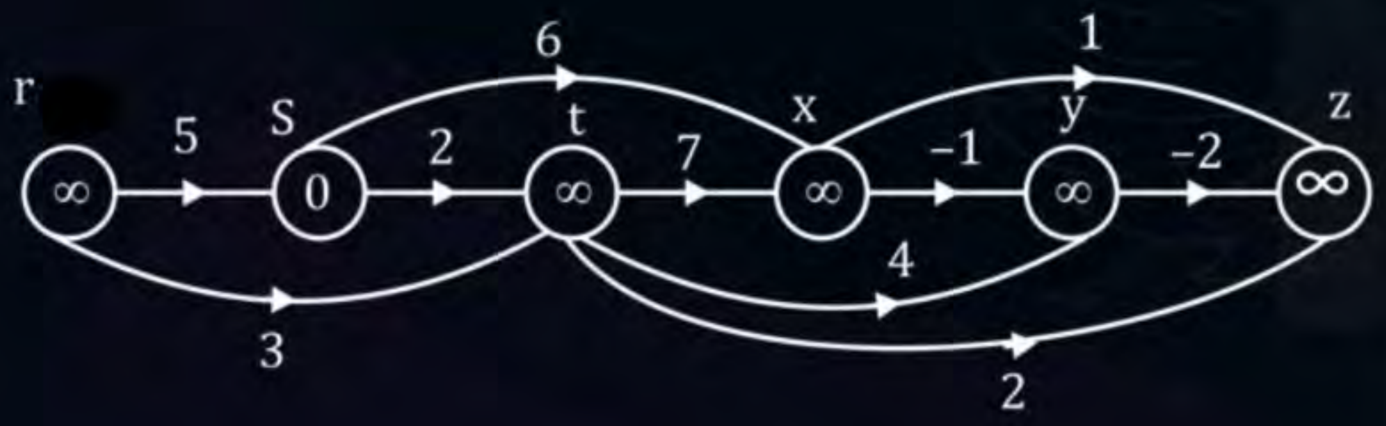


Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz



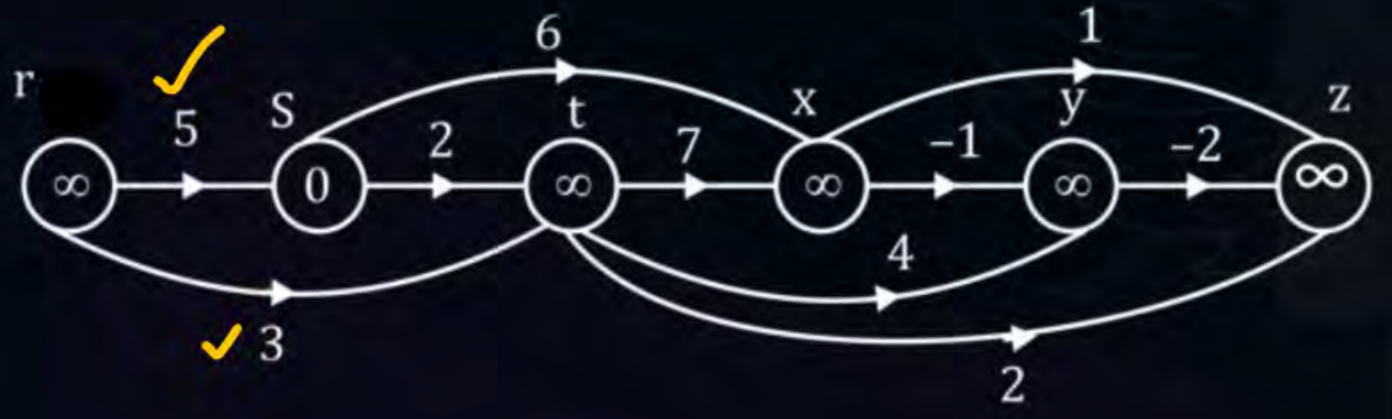
Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz

Relaxing from r



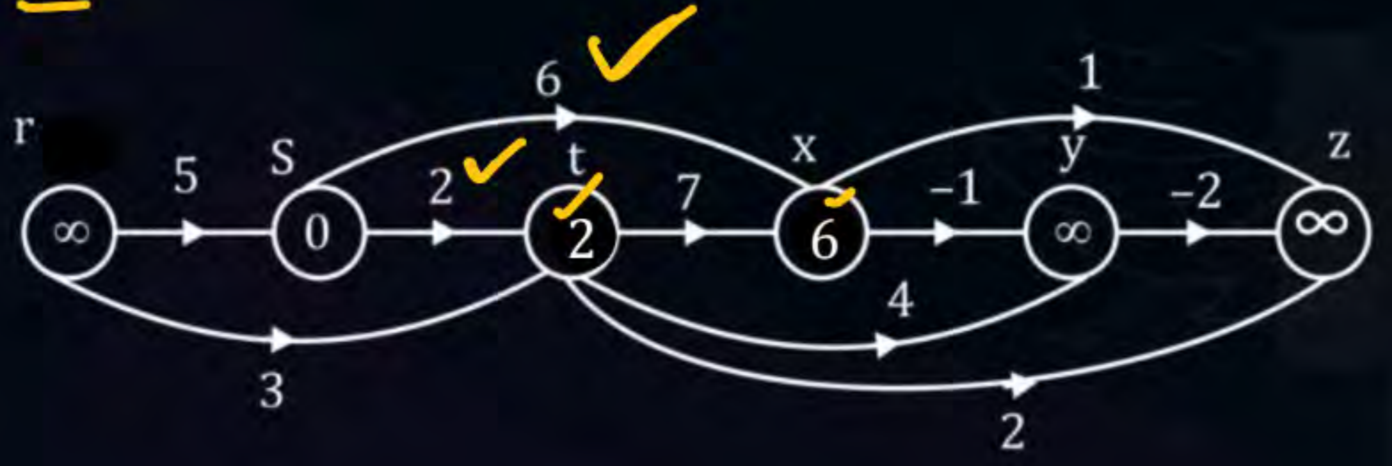
Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz

Relaxing edges from S



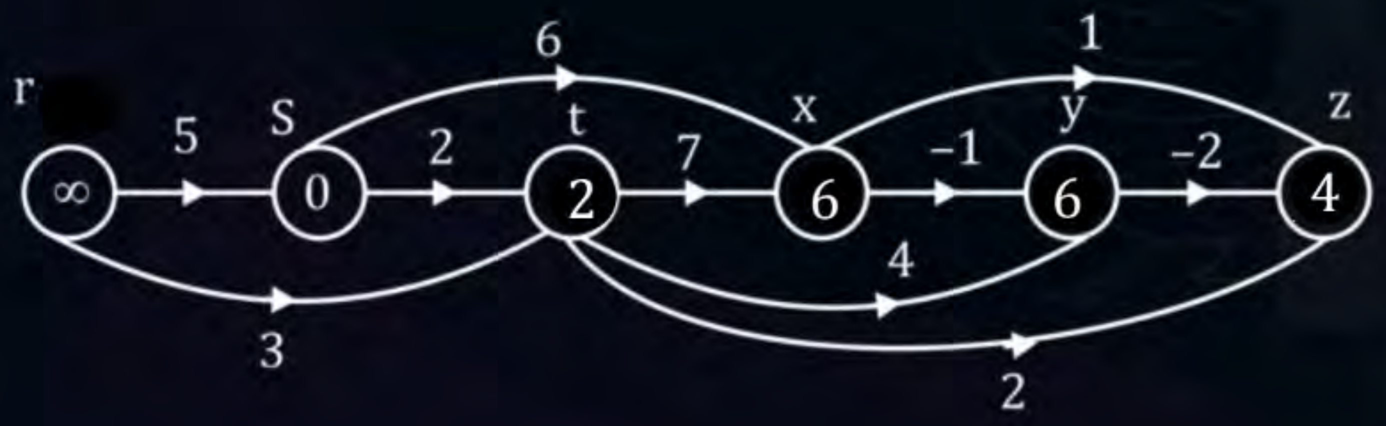
Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz

Relaxing edges from t



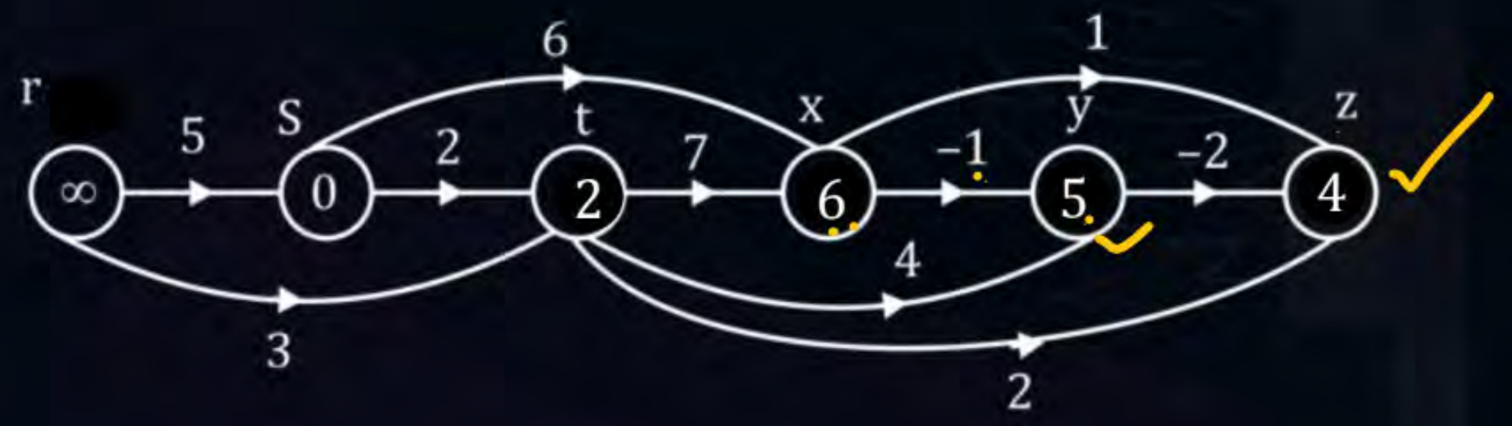
Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz

Relaxing edges from x



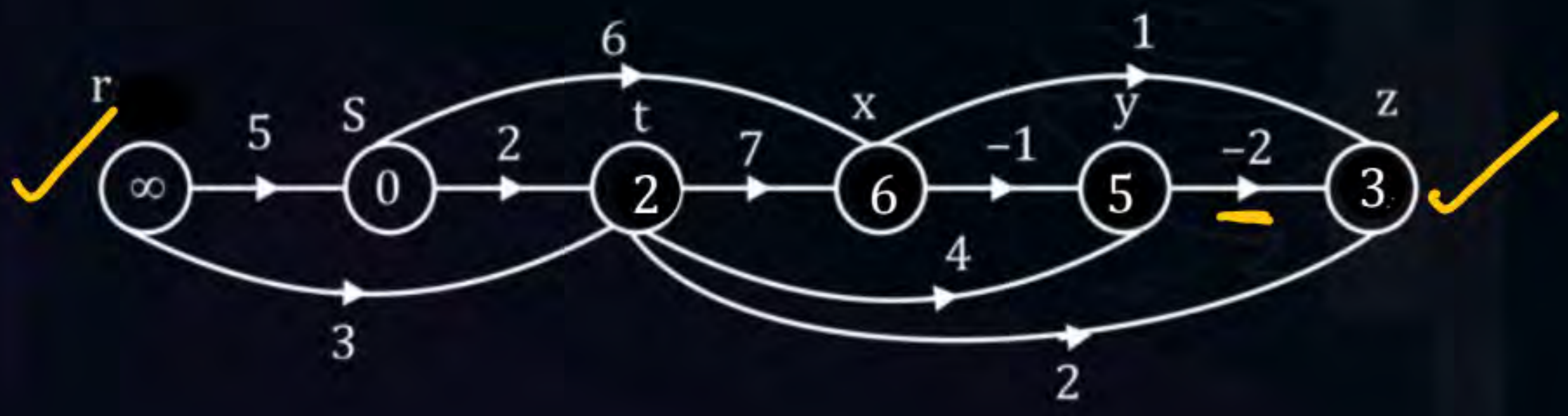
Topic: : Shortest path in DAGS

Example:-1

What is the shortest path from S to all the other vertices.

Topological order: rstxyz

Relaxing edges from y



Topic: : Shortest path in DAGS

Topological sort = d (V + E)

Total relaxion done = O(E)

$$\boxed{\text{Total time complexity} = O(V+E)}$$

Topic: : Shortest path in DAGS

DAG - shortest-paths (G, W, S)

{

1. Topologically sort the vertices of ‘G’. → $O(V+E)$
1. Initialize-single-source (G, S). → $O(V)$
3. for each vertex u, taken in topologically sorted order → $O(E)$
   4. for each vertex $v \in G.adj[u]$
   5. Relax (u, v, W)

}

Time complexity = $O(V+E)$.

Topic: : Dynamic programming

Example:

Fibonacci series –

f(n) = f(n-1) + f(n-2)

= 1; n = 1

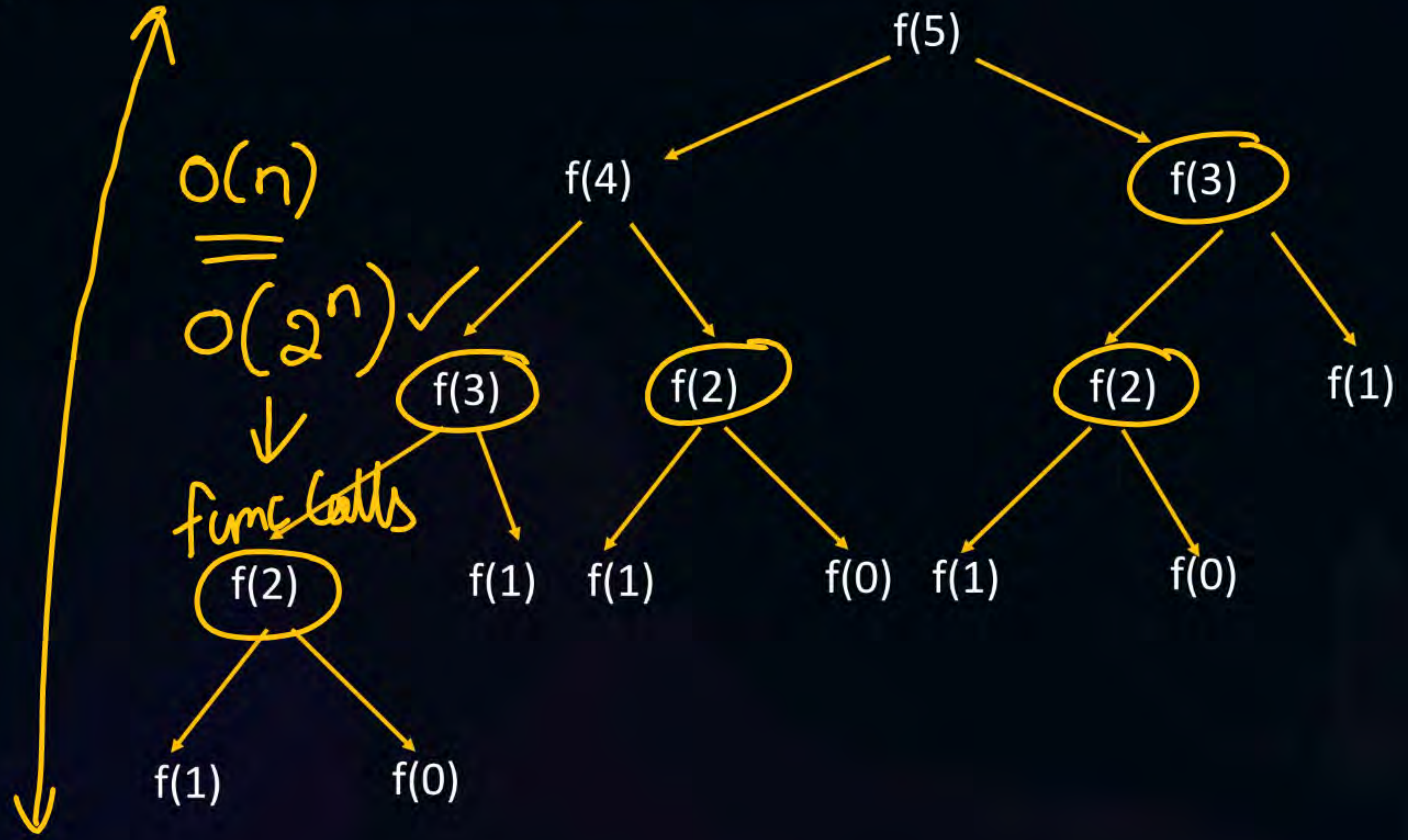
= 0; n = 0

n = 0, 2, 3,

f(n) = 1, 3, 5

```
f(n)
{
  if(n ==0)
     return 0;
  if (n ==1)
     return 1;
  return (f(n-1)+f(n-2));
}
```

Topic: : Dynamic programming



Topic: : Dynamic programming

Create a table instead of calling same function so many times

Find f(5)

0	1	2	3	4	5
0	1				

Topic: : Dynamic programming

Create a table instead of calling same function so many times

Find f(5)

0	1	2	3	4	5
0	1	1			

Topic: : Dynamic programming

Create a table instead of calling same function so many times

Find f(5)

0	1	2	3	4	5
0	1	1	2		

Topic: : Dynamic programming

Create a table instead of calling same function so many times

Find f(5)

0	1	2	3	4	5
0	1	1	2	3	

Topic: : Dynamic programming

Create a table instead of calling same function so many times

Find f(5)

0	1	2	3	4	5
0	1	1	2	3	5

$O(2^n) \rightarrow O(n)$ ✓

THANK - YOU