# Design and Analysis of Algorithms

## Lecture-29

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

# Shortest path algorithm

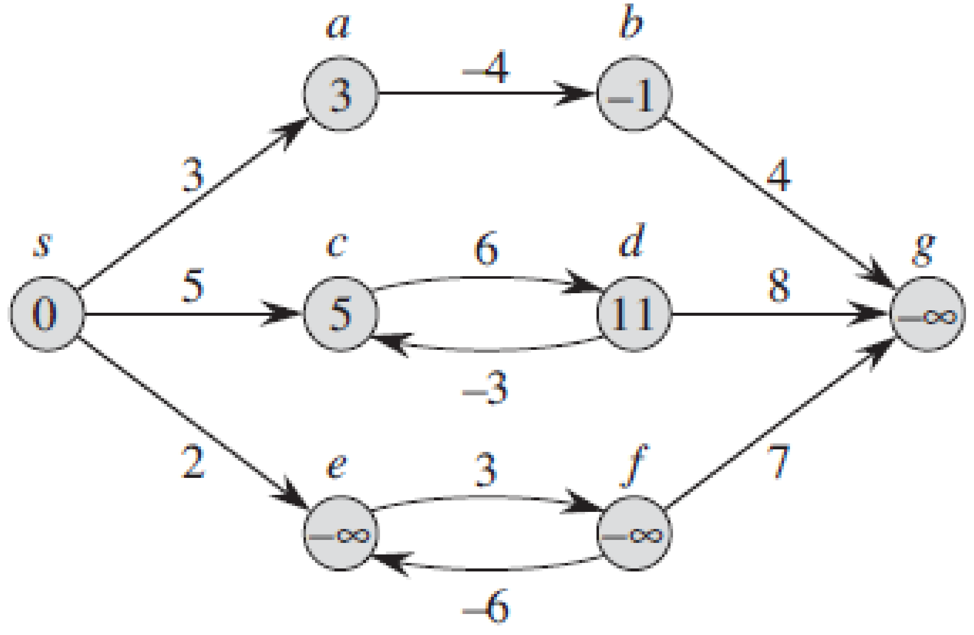
## Single-source shortest-paths problem

Given a graph G = (V, E), we want to find a shortest path from a given source vertex s ϵ V to each vertex v ϵ V .

## Negative weight cycles

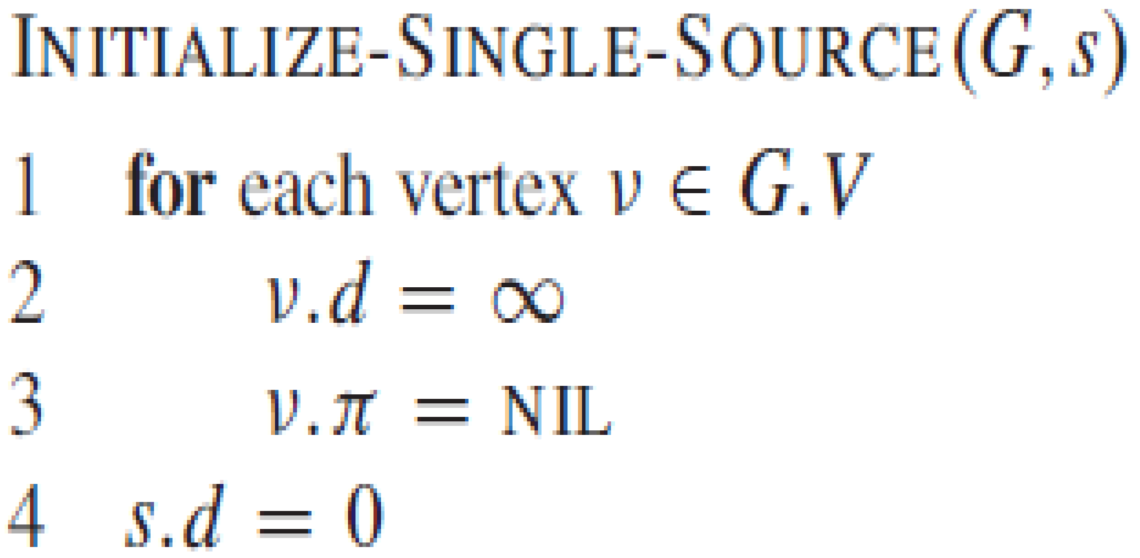
A cycle in the graph is said to be negative weight cycle if the sum of weights of all the edges in the cycle is negative value.

**Example:** Consider the following graph



# Shortest path algorithm

## Initialize the single source function

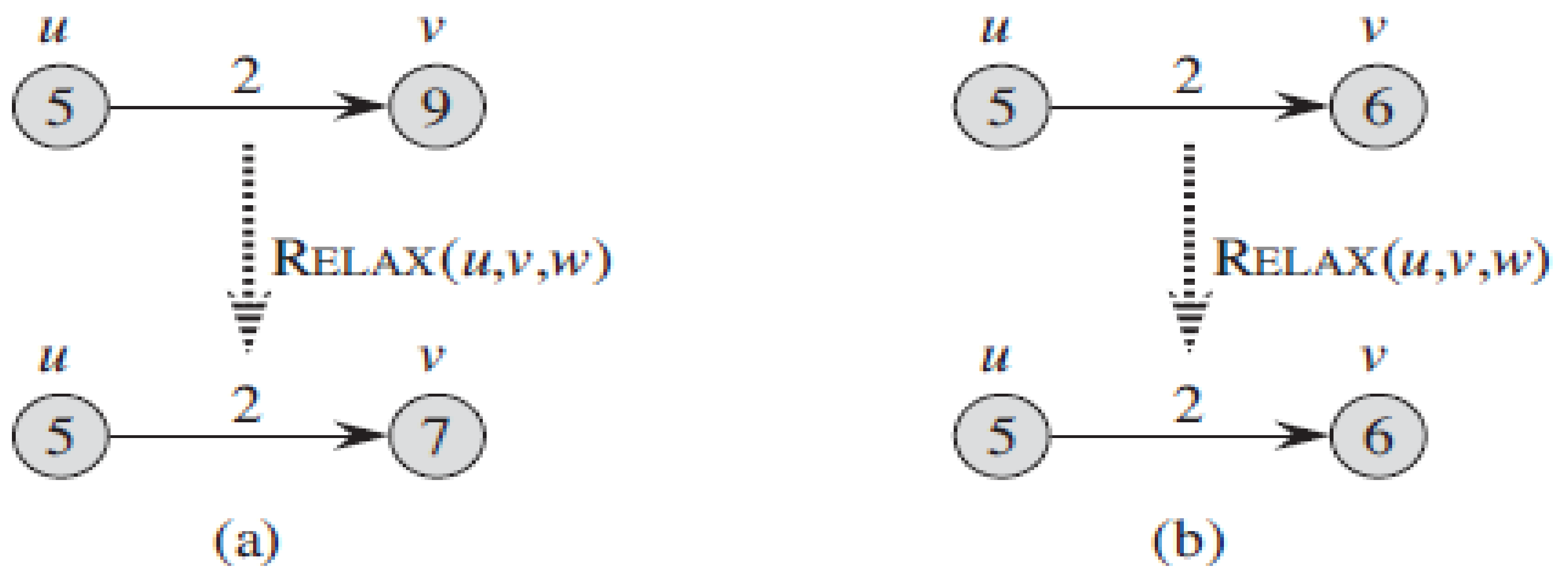


```
INITIALIZE-SINGLE-SOURCE$(G, s)$
1  for each vertex $v \in G.V$
2      $v.d = \infty$
3      $v.\pi = \text{NIL}$
4  $s.d = 0$
```

s is the source vertex.
v.d is the distance of vertex v from s.
v.π is the predecessor vertex of v.

# Shortest path algorithm

**Relaxation functior**



```
RELAX(u, v, w)
1  if $v.d > u.d + w(u, v)$
2      $v.d = u.d + w(u, v)$
3      $v.\pi = u$
```

# Shortest path algorithm

Here, we shall study two single source shortest path algorithms. These algorithms are the following:-

1. Bellman-Ford algorithm
2. Dijkastra algorithm

Both algorithms are based on greedy approach.

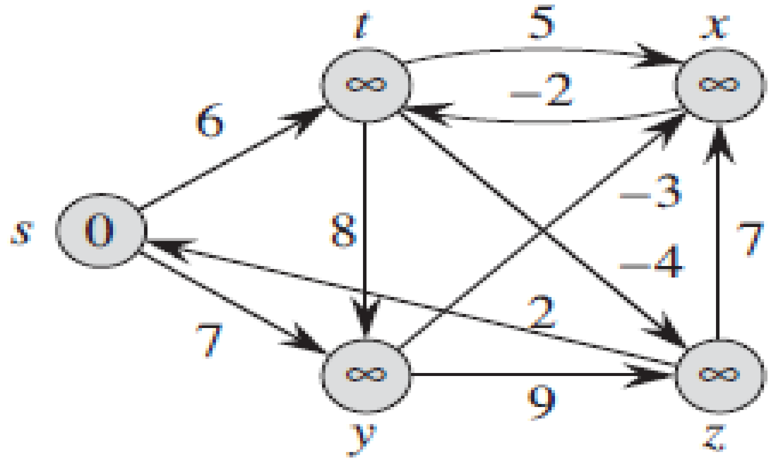
# Bellman-Ford algorithm

The **Bellman-Ford algorithm** solves the single-source shortest-paths problem in the general case in which edge weights may be negative. Given a weighted, directed graph G = (V, E) with source s and weight function w : E  R, the Bellman-Ford algorithm returns a boolean value indicating whether or not there is a negative-weight cycle that is reachable from the source. If there is such a cycle, the algorithm indicates that no solution exists. If there is no such cycle, the algorithm produces the shortest paths and their weights.

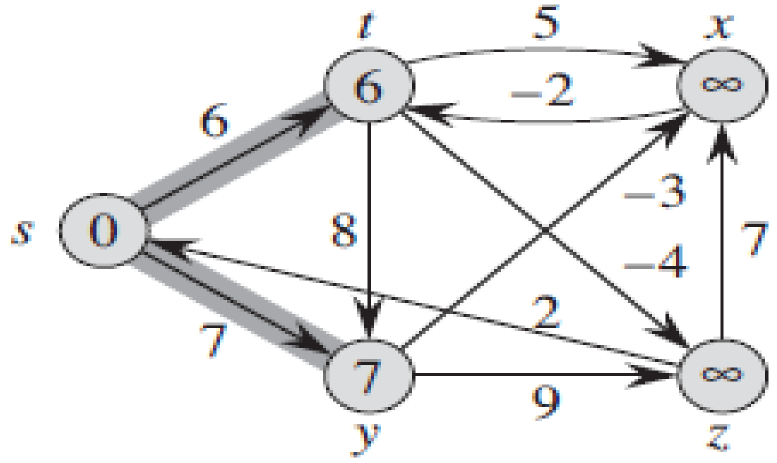
The algorithm returns TRUE if and only if the graph contains no negative-weight cycles that are reachable from the source.

# Bellman-Ford algorithm

**Example:**



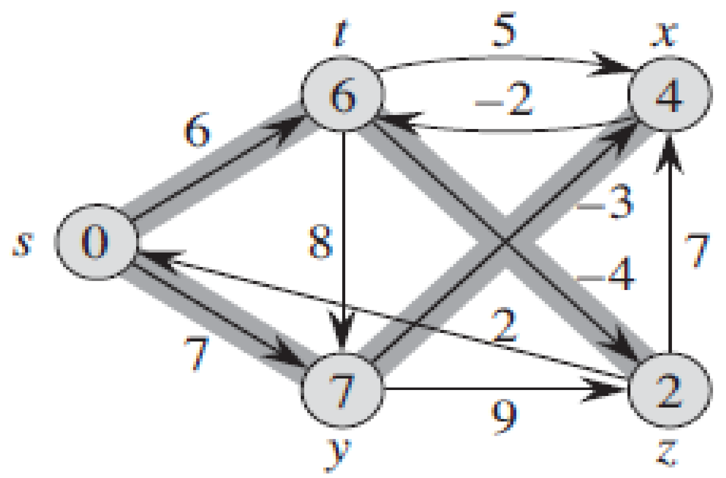
(a)



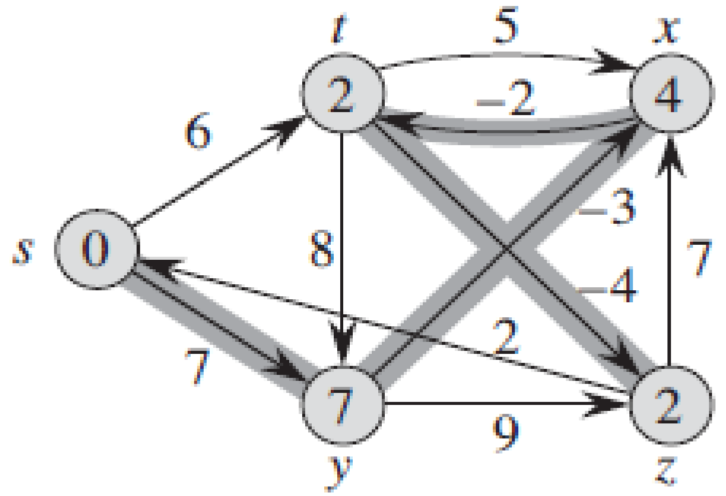
(b)

The execution of the Bellman-Ford algorithm. The source is vertex s. The d values appear within the vertices, and shaded edges indicate predecessor values: if edge (u, v) is shaded, then: v. = u. In this particular example, each pass relaxes the edges in the order (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), ( z, s), (s, t ), (s, y).

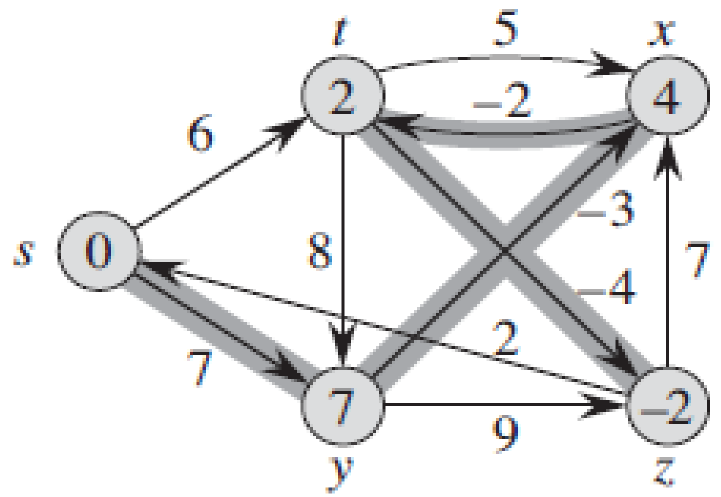
# Bellman-Ford algorithm



(c)



(d)



(e)

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

# Bellman-Ford algorithm

```
BELLMAN-FORD$(G, w, s)$
1  INITIALIZE-SINGLE-SOURCE$(G, s)$
2  for $i = 1$ to $|G.V| - 1$
3      for each edge $(u, v) \in G.E$
4          RELAX$(u, v, w)$
5  for each edge $(u, v) \in G.E$
6      if $v.d > u.d + w(u, v)$
7          return FALSE
8  return TRUE
```

**Time complexity of this algorithm is O(VE).**