Bellman-Ford Algorithm

Adapted from the CLRS book slides

THE BELLMAN-FORD ALGORITHM

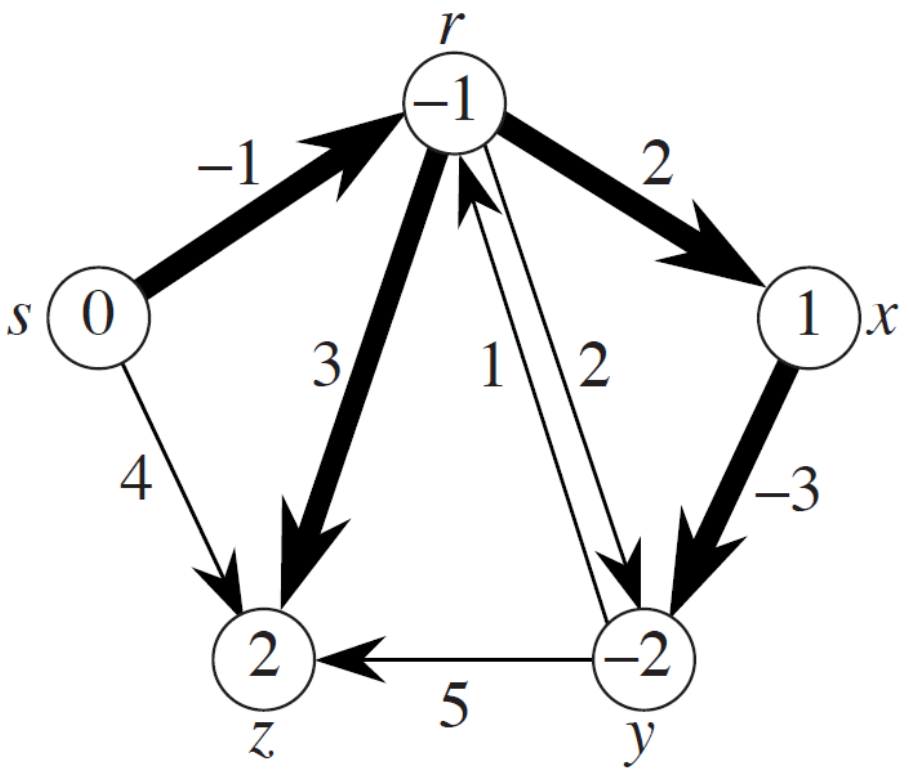
- Allows negative-weight edges.
- Computes $v.d$ and $v.\pi$ for all $v \in V$.
- Returns TRUE if no negative-weight cycles reachable from s, FALSE otherwise.

PSEUDO CODE AND RUNNING TIME

```
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) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if v.d > u.d + w(u, v)
7            return FALSE
8  return TRUE
```

Time: $O(V^2 + VE)$. The first **for** loop makes $|V| - 1$ passes over the edges, and each pass takes $\Theta(V + E)$ time. We use O rather than Θ because sometimes $< |V| - 1$ passes are enough

EXAMPLE

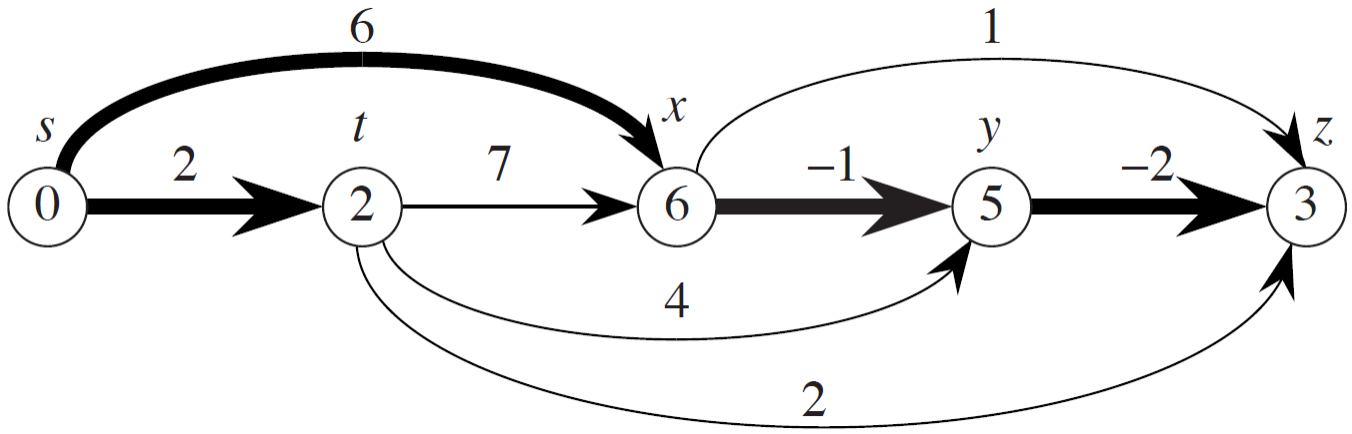


SINGLE-SOURCE SHORTEST PATHS IN A DIRECTED ACYCLIC GRAPH

Since a dag, we're guaranteed no negative-weight cycles.

```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u ∈ G.V, taken in topologically sorted order
4      for each vertex v in G.Adj[u]
5          RELAX(u, v, w)
```

EXAMPLE



Time

$\Theta(V + E)$.

Correctness

Because vertices are processed in topologically sorted order, edges of *any* path must be relaxed in order of appearance in the path.

$\Rightarrow$ Edges on any shortest path are relaxed in order.

$\Rightarrow$ By path-relaxation property, correct.