Shortest Path: Bellman-Ford Algorithm Shortest Path in a DAG

CSCI 3100

INITIALIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

Relaxation (compute estimates)

The Bellman-Ford algorithm idea

Given a graph G and a starting vertex s

Apply "relaxation" to each vertex, v, until v.d = δ(s, v)

Whenever a smaller v.d is found, update the predecessor of v, v.π

Question: how do we know when v.d = δ(s, v)?

Relaxation properties

Convergence property

- If p is a shortest path from s to v using an edge (u, v), and
- if u.d = δ(s, u) at any time prior to relaxing edge (u, v),
- then v.d= δ(s, v) after edge (u, v) has been relaxed.

Path relaxation property

- If p=<v_1, v_2, …, v_k> is a shortest path from v_1 to v_k, and we relax the edges of p in order (v_1, v_2), (v_2, v_3), …, (v_{k-1}, v_k), then v_k.d = δ(v_1, v_k)
- This property holds even if other relaxations are intermixed with the relaxation of edges of p

Bellman-Ford Algorithm

Line 1

$O(|V|)$

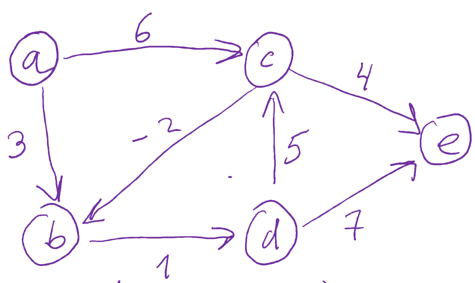
Lines 2 – 4

$O(|V| \cdot |E|)$

Lines 5 - 7

$O(|E|)$

```
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
```



$s = a$ $|V| = 5$

$a.d = 0$

$b.d = \infty$

$c.d = \infty$

$d.d = \infty$

$e.d = \infty$

(a,b): $b.d = a.d + w(a,b) = 3$; $b.\pi = a$

(a,c): $c.d = a.d + w(a,c) = 6$; $c.\pi = a$

(b,d): $d.d = b.d + w(b,d) = 4$; $d.\pi = b$

(c,b) ———

(c,e): $e.d = c.d + w(c,e) = 10$; $e.\pi = c$

(d,c): ———

(d,e): ———

Key element of Bellman-Ford Algorithm

How do we know that after |V|-1 iterations, each edge has been relaxed as much as possible:

- v.d = δ(s, v) OR
- Path from s to v has a negative weight cycle

If shortest path exists, it will use at most |V|-1 edges

Let *v* be any vertex reachable from *s* via a shortest path:
$(v_1, v_2, \ldots, v_k)$, where $s=v_1$ and $v=v_k$

Each of the |V|-1 iterations relaxes |E| edges

Among edges relaxed at iteration i is edge (v_{i-1}, v_i)

By the path relaxation property, *v.d = vk.d = δ(s, v)*

Shortest Path in a Directed Acyclic Graph (DAG)

Shortest path in a DAG is well defined (no cycles, so no negative weight cycles)

Relax edges of a DAG in the topological sort order

How do we get a topological sort order of vertices of a DAG?

A. Use Depth First Search

B. Use Prim's algorithm

C. Use Bellman-Ford algorithm

D. Use Kruskal's algorithm

E. Use priority queue

Shortest Path in a DAG

If DAG contains a path from vertex *v* to vertex *u* then

- *v* precedes *u* in a topological sort

If we make one path over the vertices in the topological order

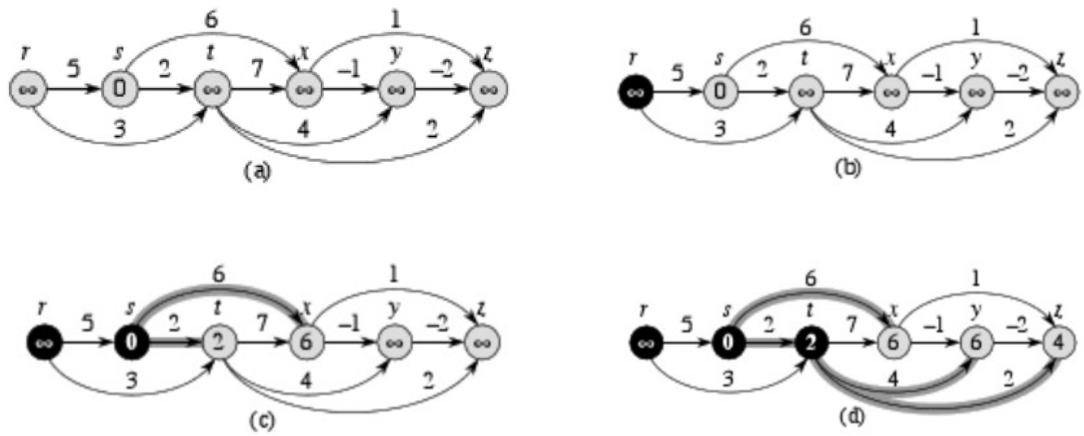
What is the running time of this algorithm?

$O(|V|+|E|)$

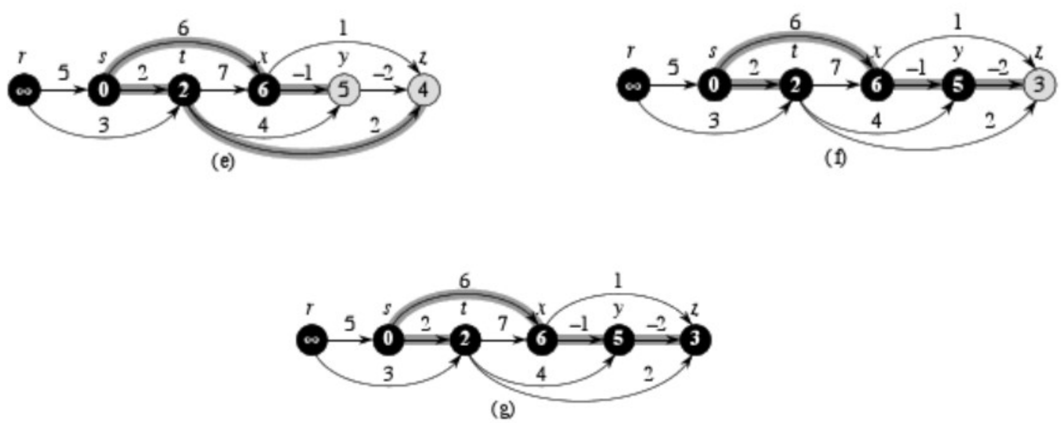
DAG-SHORTEST-PATHS(G, w, s)
1 topologically sort the vertices of G → $O(|V|+|E|)$
2 INITIALIZE-SINGLE-SOURCE(G, s) → $O(|V|)$
3 **for** each vertex u, taken in topologically sorted order → V
4   **for** each vertex $v \in G.Adj[u]$
5   RELAX(u, v, w) → E

$|V|+|E|$

Example



Example continued



If $p = (v_1, v_2, \ldots, v_k)$ is the shortest path from $s=v_1$ to $v= v_k$, produced by Shortest Path in DAG algorithm, then

A. Edges of p are relaxed in the order $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$

B. At iteration k, $v_k.d= \delta(s, v_k)$

C. When relaxing edges adjacent to v_i, $v_i.d= \delta(s, v_i) < \infty$

D. For any vertex v_i $(v_1, v_2, \ldots, v_i)$ is the shortest path from s to v_i

E. All of the above are true

Application of Shortest Path in DAG

Determine a critical path in a schedule

Represent jobs as edges

Edge weights – time to perform each job

If edge (u, v) enters vertex v and edge (v, x) leaves vertex v, then job (u, v) must be done before job (v, x)

A path through this DAG – a sequence of jobs that need to be performed in a particular order

In this context – a critical path is a longest path through the graph

Modify weights to be the negative of the time to perform each job and run Shortest Path in DAG algorithm