Advanced Algorithms Analysis and Design

By

Nazir Ahmad Zafar

Lecture No 34

Proof: Bellman-Ford Algorithm

and

Shortest Paths in Directed Acyclic Graphs

Today Covered

- Bellman-Ford Algorithm
  - Analysis
  - Proof
- Shortest path in Directed Acyclic Graphs
  - Assumptions
  - Algorithm
  - Analysis
  - Proof of correctness

Analysis: The Bellman-Ford Algorithm

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

Θ (V)

Θ (V.E)

O(E)

Total Running Time = O(V.E)

Contd..

Lemma 1

Statement: Let $G = (V, E)$ *be*

- directed, with source *s,*
- a weighted, with weight function $w : E \rightarrow \boldsymbol{R}$, and
- contains no negative-weight cycle reachable from *s.* Then, after the $|V| - 1$ iterations of the ***for*** loop of lines 2-4 of BELLMAN-FORD, we have $d[v] = \delta(s, v)$ for all vertices *v* that are reachable from *s.*

Path relaxation property

- *If* $p = \langle v_0, v_1, \ldots, v_k \rangle$, be a shortest path from $s = v_0$ to v_k and edges of p are relaxed in the order (v_0, v_1), $(v_1, v_2) \ldots (v_{k-1}, v_k)$, then $d(v_k) = \delta(s, v_k)$

Lemma 1 (Contd..)

Proof

- We prove it using path-relaxation property.
- Consider any vertex v that is reachable from s
- And let $p = \langle v_0, v_1, \ldots, v_k \rangle$, be any acyclic shortest path from s to v, where $v_0 = s$ and $v_k = v$,
- As there are k+1 vertices in the path p, hence there must be k edges in p.
- Because Graph has |V| vertices and path p contains no cycle, hence path p has at most $|V| - 1$ edges, and therefore, $k \leq |V| - 1$.
- Each of the |V| - 1 iterations of the for loop of lines 2-4 relaxes all E edges.

Lemma 1 (Contd..)

- At i = 1, edge (v_0, v_1) is relaxed, and $d[v_1] = \delta(s, v_1)$
- At i = 2, edge (v_1, v_2) is relaxed, and $d[v_2] = \delta(s, v_2)$
- By mathematical induction we can prove that
- At i = k, edge (v_{k-1}, v_k) is relaxed, $d[v_k] = \delta(s, v_k)$
- Hence all the edges (v_{i-1}, v_i) will be relaxed after the iterations, i = 1, 2,..., k.
- By the path-relaxation property, after k^{th} iteration,
  $d[v] = d[v_k] = \delta(s, v_k) = \delta(s, v)$.
- Hence we have proved the required result using path relaxation property.

Theorem : Correctness of Bellman-Ford algorithm

Let BELLMAN-FORD be run on weighted, directed graph G = (V, E), with source vertex s, and weight function $w : E \rightarrow R$.

- If G contains no negative-weight cycles that are reachable from s, then
  - $d[v] = \delta(s, v)$ for all vertices $v \in V$, and
  - the algorithm returns TRUE
  - the predecessor subgraph G_π is shortest-paths tree rooted at s.
- If G does contain a negative weight cycle reachable from s, then the algorithm returns FALSE.

Proof

Case 1

Suppose graph G contains no negative-weight cycles that are reachable from the source s.

- We first prove the claim that at termination, $d[v] = \delta(s, v)$ for all vertices $v \in V$.
  - If v is reachable from s, **Lemma above** proves it.
  - If v is not reachable from s, then claim follows from no-path property.
- The predecessor subgraph property, along with the claim, implies that G_π is a shortest-paths tree.

(Once d[v] = δ(s, v) for all v ∈ V, the predecessor sub-graph is a shortest paths tree rooted at s)

Contd..

- Now we use the claim to show that BELLMAN-FORD returns TRUE.
  - At termination, for all edges (u, v)
  - $d[v] = \delta(s, v) \leq \delta(s, u) + w(u, v) = d[u] + w(u, v)$,
  - It therefore returns TRUE

Case 2,

- Suppose that graph G contains a negative-weight cycle that is reachable from the source s
- Let this cycle be $c = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = v_k$,

Then, $$\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0 \qquad \text{(A)}$$

Contd..

- Assume for the purpose of contradiction that the Bellman-Ford algorithm returns TRUE.

- Thus, $d[v_i] \le d[v_{i-1}] + w(v_{i-1}, v_i)$ for $i = 1, 2, \ldots, k$.
- Summing the inequalities around cycle c gives us

$$\sum_{i=1}^{k} d[v_i] \le \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$

$$= \sum_{i=1}^{k} (d[v_{i-1}] + \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Contd..

- Since $v_0 = v_k$, each vertex in c appears exactly once in each of the summations and, and so

$$\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$$

- Of course $d[v_i]$ is finite for $i = 1, 2, \ldots, k$. Thus,

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- Which contradicts inequality (A). And hence it proves the theorem

Shortest Paths in Directed Acyclic Graphs

Shortest Paths in Directed Acyclic Graphs

- By relaxing edges of Directed Acyclic Graph (dag) $G = (V, E)$ according to topological sort of vertices single source shortest path can be computed in $\Theta (V + E)$ time
- Shortest paths are always well defined in a dag
  - Since even if there are negative-weight edges no negative weight cycle exists.
- It starts topologically sorting dag, to impose linear ordering of vertices.
  - If there is path from u to v then u precedes v.
- Each vertex and each edge that leaves the vertex is processed that is why this approach is well defined

Algorithm : Topological Sort

TOPOLOGICAL-SORT (*G*)

1. Call DFS(G) to compute $f[v]$ of each vertex $v \in V$.
2. Set an empty linked list $L = \emptyset$.
3. When a vertex v is colored black, assign it $f(v)$.
4. Insert v onto the front of the linked list, $L = \{v\}.L$.
5. **return** the linked list.
6. The rank of each node is its position in the linked list started from the head of the list.

Total Running Time = Θ (V + E)

Example

Algorithm : Shortest Path (dag)

DAG-SHORTEST-PATHS (*G*, *w*, *s*)

1 topologically sort the vertices of *G* — $\Theta(V+E)$

2 INITIALIZE-SINGLE-SOURCE (*G*, *s*) — $\Theta(V)$

3 **for** each vertex *u*, taken in topologically sorted order — $\Theta(V)$

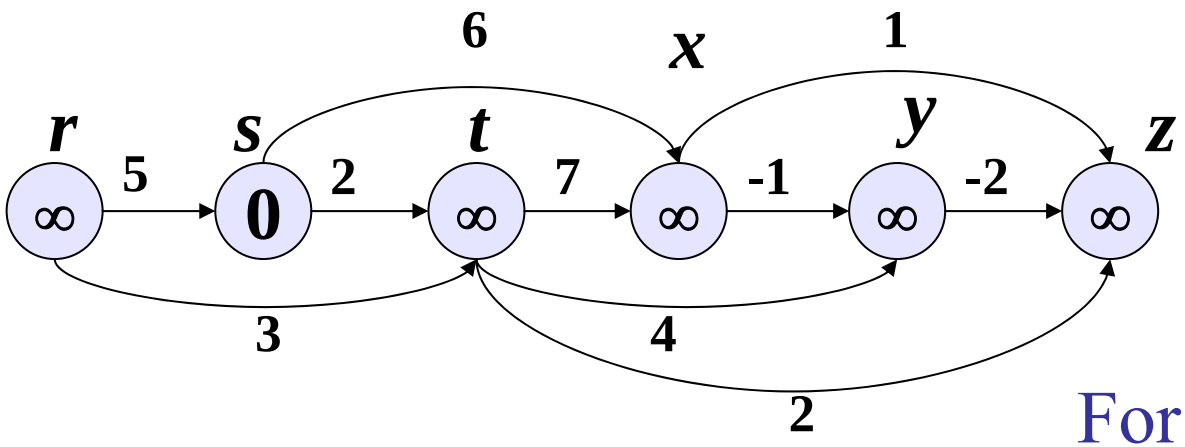
4 **do for** each vertex $v \in Adj[u]$ — $\Theta(E)$

5 **do** RELAX (*u*, *v*, *w*) — $\Theta(E)$

Each iteration of for loop takes $\Theta(1)$

Total Running Time = Θ (V+E)

SSSP in Directed Acyclic Graphs



For each vertex $v \in V(G)$

$d[v] \leftarrow \infty$

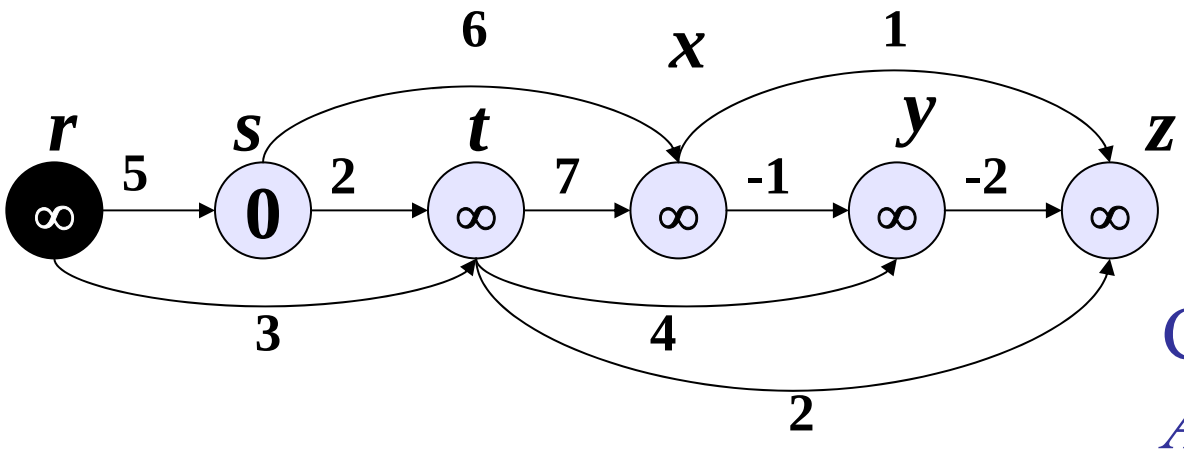
$\pi[v] \leftarrow \text{NIL}$

Considering s as root node

$d[s] \leftarrow 0$

The vertices are taken in topologically sorted order

SSSP in Directed Acyclic Graphs



Considering vertex r

$Adj[r] = s, t$

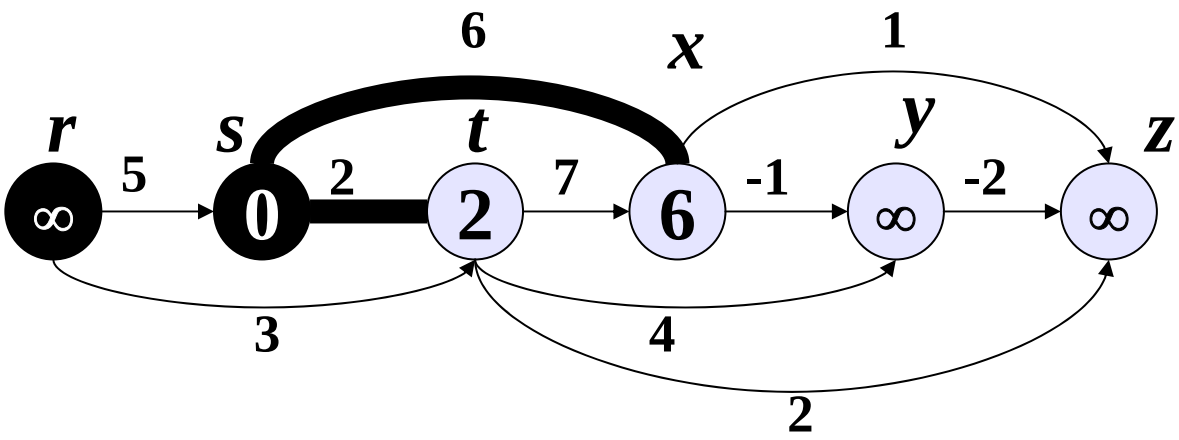
$d[s] > d[r] + w(r, s)$

But $(0 < \infty + 5)$

$d[t] > d[r] + w(r, t)$

But $(\infty < \infty + 3)$

SSSP in Directed Acyclic Graphs



Considering vertex s

$Adj[s] = t, x$

$d[t] > d[s] + w(s, t)$

$(\infty > 0 + 2)$

$d[t] \leftarrow d[s] + w(s, t)$

$0 + 2 = 2$

$\pi[t] \leftarrow s$

$d[x] > d[s] + w(s, x)$

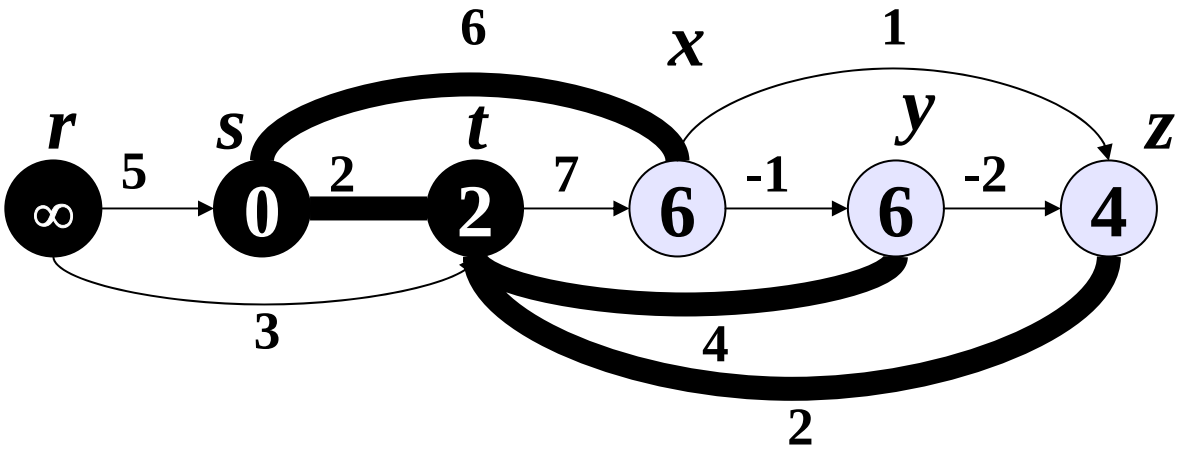
$(\infty > 0 + 6)$

$d[x] \leftarrow d[s] + w(s, x)$

$0 + 6 = 6$

$\pi[x] \leftarrow s$

SSSP in Directed Acyclic Graphs



Considering vertex t

$Adj[t] = x, y, z$

$d[x] > d[t] + w(t, x)$

But $(6 < 9)$

$d[y] > d[t] + w(t, y)$

$(\infty > 2 + 4)$

$d[y] \leftarrow d[t] + w(t, y)$

$2 + 4 = 6$

$\pi[y] \leftarrow t$

$d[z] > d[t] + w(t, z)$

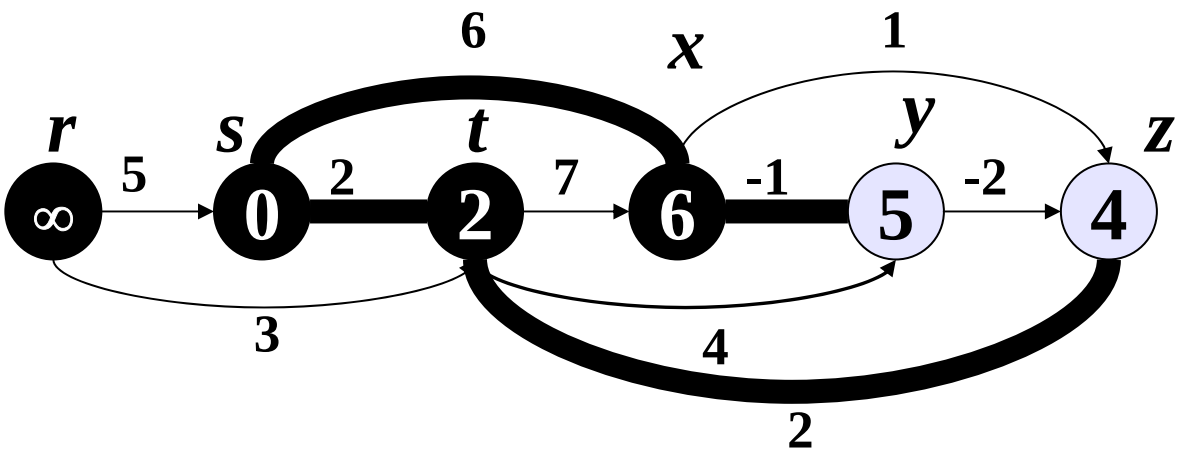
$(\infty > 2 + 2)$

$d[z] \leftarrow d[t] + w(t, z)$

$2 + 2 = 4$

$\pi[z] \leftarrow t$

SSSP in Directed Acyclic Graphs



Considering vertex x

$Adj[x] = y, z$

$d[y] > d[x] + w(x, y)$

$(6 > 6 + (-1))$

$d[y] \leftarrow d[x] + w(x, y)$

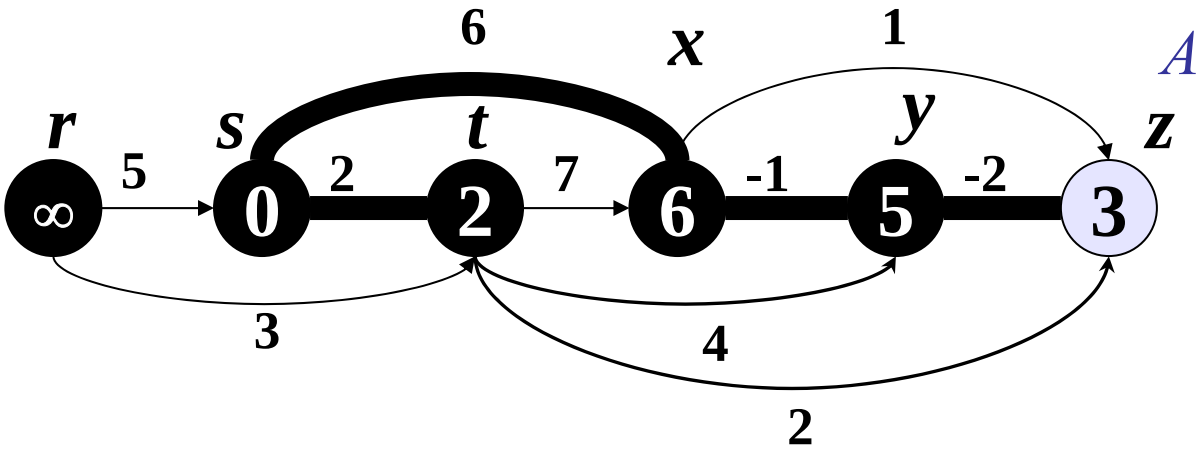
$6 + (-1) = 5$

$\pi[y] \leftarrow x$

$d[z] > d[x] + w(x, z)$

But $(4 < 6 + 1)$

SSSP in Directed Acyclic Graphs



Considering vertex y

$Adj[y] = z$

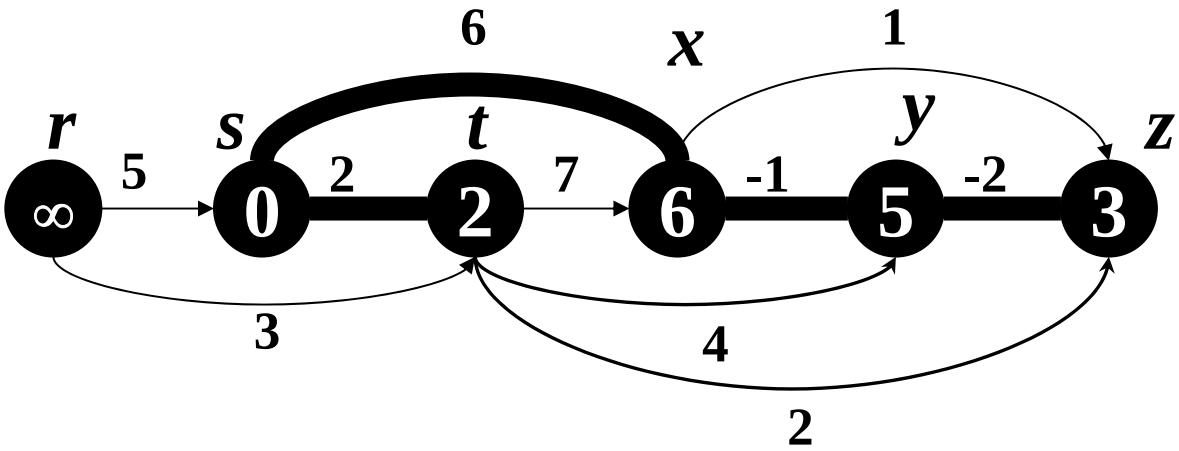
$d[z] > d[y] + w(y, z)$

$(4 > 5 + (-2))$

$d[z] \leftarrow d[y] + w(y, z)$

$5 + (-2) = 3$

$\pi[z] \leftarrow y$



Considering vertex z

$Adj[z] = \varnothing$

Theorem: Proof of Correctness

If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, $d[v] = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree.

Proof

- We first show that $d[v] = \delta(s, v)$ for all vertices $v \in V$ at termination.

Case 1

- If v is not reachable from s, then $d[v] = \delta(s, v) = \infty$ by the no-path property.

Theorem (Cont.)

Case 2

- Now, suppose that v is reachable from s, so that there is a shortest path $p = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = s$ and $v_k = v$.
- Because we process the vertices in topologically sorted order, the edges on p are relaxed in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$.
- The path-relaxation property implies that $d[v_i] = \delta(s, v_i)$ at termination for $i = 0, 1, \ldots, k$.
- Hence it proves the theorem