CSCI 470: Dijkstra’s algorithm, Bellman-Ford algorithm

Vijay Chaudhary

November 8, 2023

Department of Electrical Engineering and Computer Science
Howard University

Overview

1. Properties of shortest paths and relaxation
2. Correctness of Dijkstra's algorithm
3. Bellman-Ford algorithm
4. Single-source shortest paths in directed acyclic graphs

Properties of shortest paths and relaxation

Properties of shortest paths and relaxation

- **Triangle inequality** (Lemma 24.10)
  For any edge $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.
- **Upper-bound property** (Lemma 24.11)
  We always have $v.d \geq \delta(s, v)$ for all vertices $v \in V$, and once $v.d$ achieves the value $\delta(s, v)$, it never changes.
- **No-path property** (Lemma 24.12)
  If there is no path from s to v, then we always have $v.d = \delta(s, v) = \infty$.
- **Convergence property** (Lemma 24.14)
  If $s \rightsquigarrow u \rightarrow v$ is a shortest path in G for some $u, v \in V$, and if $u.d = \delta(s, u)$ at any time prior to relaxing edge (u, v), then $v.d = \delta(s, v)$ at all times afterward.

Properties of shortest paths and relaxation

- **Path-relaxation property** (Lemma 24.15)
  If $p = \langle v_0, v_1, ..., v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)$, then $v_k.d = \delta(s, v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.
- **Predecessor-subgraph property** (Lemma 24.17)
  Once $v.d = \delta(s, v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s.

Correctness of Dijkstra's algorithm

Dijkstra’s algorithm: pseudocode

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Correctness of Dijkstra's algorithm

We use the following loop invariant:

At the start of each iteration of the ***while*** *loop of lines 4-8, $v.d = \delta(s, v)$ for each vertex $v \in S$.*

Initialization / Base case

Initially, $S = \emptyset$, and so the invariant is trivially true.

A quick recap

- We define the ***shortest-path weights*** $\delta(u, v)$ from u to v by

$$\delta(u, v) = \begin{cases} min\{w(p) : u \leadsto v\} & \exists \text{ a path } u \text{ to } v, \\ \infty & \text{otherwise.} \end{cases}$$

- A ***shortest path*** from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.

Maintenance/Inductive step

- We wish to show that in each iteration, $u.d = \delta(s, u)$ for the vertex added to set S.

Maintenance/Inductive step

- We wish to show that in each iteration, $u.d = \delta(s, u)$ for the vertex added to set S.
- For the purpose of contradiction, let u be the first vertex for which $u.d \neq \delta(s, u)$ when it is added to set S.

Maintenance/Inductive step

- We wish to show that in each iteration, $u.d = \delta(s, u)$ for the vertex added to set S.
- For the purpose of contradiction, let u be the first vertex for which $u.d \neq \delta(s, u)$ when it is added to set S.
- We must have $u \neq s$ because s is the first vertex added to set S and $s.d = \delta(s, s) = 0$ at that time.

Maintenance/Inductive step

- We wish to show that in each iteration, $u.d = \delta(s, u)$ for the vertex added to set S.
- For the purpose of contradiction, let u be the first vertex for which $u.d \neq \delta(s, u)$ when it is added to set S.
- We must have $u \neq s$ because s is the first vertex added to set S and $s.d = \delta(s, s) = 0$ at that time.
- Because $u \neq s$, we have that $S \neq \emptyset$ just before u is added to S.

Maintenance/Inductive step

- We wish to show that in each iteration, $u.d = \delta(s, u)$ for the vertex added to set S.
- For the purpose of contradiction, let u be the first vertex for which $u.d \neq \delta(s, u)$ when it is added to set S.
- We must have $u \neq s$ because s is the first vertex added to set S and $s.d = \delta(s, s) = 0$ at that time.
- Because $u \neq s$, we have that $S \neq \emptyset$ just before u is added to S.
- There must be some path from s to u, for otherwise $u.d = \delta(s, u)$. Because there is at least one path, there is a shortest path p from s to u. Prior to adding u to S, path p connects a vertex in S, namely s, to a vertex in $V - S$, namely u.

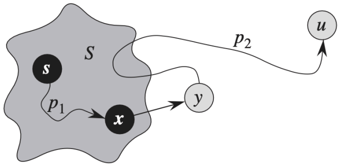


Figure 24.7 The proof of Theorem 24.6. Set S is nonempty just before vertex u is added to it. We decompose a shortest path p from source s to vertex u into $s \overset{p_1}{\leadsto} x \to y \overset{p_2}{\leadsto} u$, where y is the first vertex on the path that is not in S and $x \in S$ immediately precedes y. Vertices x and y are distinct, but we may have $s = x$ or $y = u$. Path p_2 may or may not reenter set S.

Maintenance (contd)

- Let us consider the first vertex y along p such that $y \in V - S$, and let $x \in S$ be y's predecessor along p. Thus, Figure 24.7 illustrates, we can decompose path p into $s \rightsquigarrow x \rightarrow y \rightsquigarrow u$. (Either of paths p_1 or p_2 may have no edges.)

Maintenance (contd)

- Let us consider the first vertex y along p such that $y \in V - S$, and let $x \in S$ be y's predecessor along p. Thus, Figure 24.7 illustrates, we can decompose path p into $s \rightsquigarrow x \to y \rightsquigarrow u$. (Either of paths p_1 or p_2 may have no edges.)
- We claim that $y.d = \delta(s, y)$ when u is added to S. To prove this claim, observe that $x \in S$. Then, because we chose u as the first vertex for which $u.d \neq \delta(s, u)$ when it is added to S, we had $x.d = \delta(s, x)$ when x was added to S. Edge (x, y) was relaxed at that time, the claim follows from the convergence property.

Maintenance (contd)

- We can now obtain a contradiction to prove that $u.d = \delta(s, u)$. Because y appears before u on a shortest path from s to u and all edge weights are non-negative (notably those on path p_2), we have $\delta(s, y) \leq \delta(s, u)$, and thus

$$
\begin{aligned}
y.d &= \delta(s, y) && (1)\\
&\leq \delta(s, u) && (2)\\
&\leq u.d \quad \text{by the upper-bound property} && (3)
\end{aligned}
$$

Maintenance (contd)

- We can now obtain a contradiction to prove that $u.d = \delta(s, u)$. Because y appears before u on a shortest path from s to u and all edge weights are non-negative (notably those on path p_2), we have $\delta(s, y) \leq \delta(s, u)$, and thus

$$
\begin{align}
y.d &= \delta(s, y) \tag{1}\\
&\leq \delta(s, u) \tag{2}\\
&\leq u.d \quad \text{by the upper-bound property} \tag{3}
\end{align}
$$

- But because both vertices u and y were in $V - S$ when u was chosen in line 5, we have $u.d \leq y.d$. Thus, the two inequalities in (2) and (3) are in fact equalities, giving $y.d = \delta(s, y) = \delta(s, u) = u.d$.

Maintenance (contd)

- Consequently, $u.d = \delta(s, u)$, which contradicts our choice of u. We conclude that $u.d = \delta(s, u)$ when u is added to S, and that this equality is maintained at all times thereafter.

Termination

At termination, $Q = \emptyset$ which, along with our earlier invariant that $Q = V - S$, implies that $S = V$. Thus, $u.d = \delta(s, u)$ for all vertices $u \in V$.

Bellman-Ford algorithm

Bellman-Ford algorithm

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

Lemma 24.2

Lemma 24.2
Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w : E \to \mathbb{R}$, and assume that G contains no negative-weight cycles that are reachable from s. Then, after the $|V| - 1$ iterations of the **for** loop of lines 2-4 of BELLMAN-FORD, we have $v.d = \delta(s, v)$ for all vertices v that are reachable from s.

Lemma 24.2: Proof

Proof We prove the lemma by appealing to the path-relaxation property.

Consider any vertex v that is reachable from s, and let $p = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = s$ and $v_k = v$, be any shortest path from s to v. Because shortest paths are simple, p has at most $|V| - 1$ edges, and so $k \leq |V| - 1$. Each of the $|V| - 1$ iterations of the **for** loop of lines 2-4 relaxes all $|E|$ edges. Among the edges relaxed in the i-th iteration, for $i = 1, 2, \ldots, k$, is (v_{i-1}, v_i). By the path-relaxation property, therefore, $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$. $\square$

Corollary 24.3

Let $G = (V, E)$ be a weighted, directed graph with source vertex s and weight function $w : E \rightarrow \mathbb{R}$, and assume that G contains no negative-weight cycles that are reachable from s. Then, for each vertex $v \in V$, there is a path from s to v **if and only if** BELLMAN-FORD terminates with $v.d < \infty$ when it is run on G.

Proof is left as an exercise for HW 5.

Correctness of Bellman-Ford algorithm

Let BELLMAN-FORD be run on a weighted, directed graph $G = (V, E)$ with source s and weight function $w : E \to \mathbb{R}$. If G contains no negative-weight cycles that are reachable from s, then the algorithm returns TRUE, we have $v.d = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree rooted at s. If G does contain a negative-weight cycles reachable from s, then the algorithm returns FALSE.

Proof: Case 1

Case 1:

Suppose that graph G contains no negative-weight cycles that are reachable from the source s. We first prove the claim that at termination, $v.d = \delta(s, v)$ for all vertices $v \in V$.

- If vertex v is reachable from s, then Lemma 24.2 proves this claim.
- If v is not reachable from s, then the claim follows from the no-path property. Thus, the claim is proven.

Proof: Case 1

- The predecessor-subgraph property, along with the claim, implies that G_π is a shortest-paths tree.
- Now we use the claim to show that Bellman-Ford returns TRUE. At termination, we have for all the edges $(u, v) \in E$,

$$\begin{aligned} v.d &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \\ &= u.d + w(u, v), \end{aligned}$$

and so none of the tests in line 6 causes Bellman-Ford to return FALSE. Therefore, it returns TRUE.

Case 2:

Now, suppose that graph G contains a negative-weight cycle that is reachable from the source s; let this cycle be $c = \langle v_0, v_1, ..., v_k \rangle$, where $v_0 = v_k$. Then,

$$\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0. \tag{4}$$

Proof: Case 2

- Assume for the purpose of contradiction that the BELLMAN-FORD algorithm returns TRUE.
- Thus, $v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$ for $i = 1, 2, ..., k$. Summing the inequalities around cycle c gives us

$$\begin{aligned}\sum_{i=1}^{k} v_i.d &\leq \sum_{i=1}^{k}(v_{i-1}.d + w(v_{i-1}, v_i)) \\ &= \sum_{i=1}^{k} v_{i-1}.d + \sum_{i=1}^{k} w(v_{i-1}, v_i)\end{aligned}$$

Proof: Case 2

- Since $v_0 = v_k$, each vertex in c appears exactly once in each of the summations $\sum_{i=1}^{k} v_i.d$ and $\sum_{i=1}^{k} v_{i-1}.d$, and so

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

- Moreover, by Corollary 24.3, $v_i.d$ is finite for $i = 1, 2, ..., k$. Thus,

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

which contradicts inequality (4).

Proof

We conclude that the BELLMAN-FORD algorithm returns TRUE if graph G contains no negative-weight cycles reachable from the source, and FALSE otherwise. $\square$

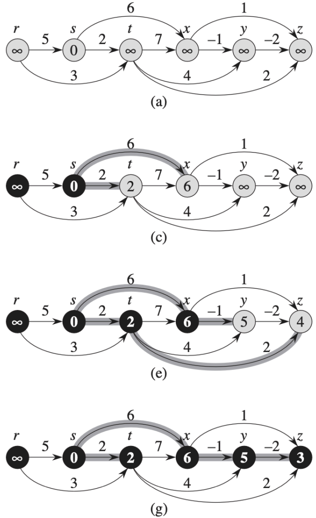
Single-source shortest paths in directed acyclic graphs

Shortest paths in DAGs

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

```
topologically sort the vertices in G
INITIALIZE-SINGLE-SOURCE(G)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Shortest paths in DAGs: Illustration



Runtime

- line 1 takes $\Theta(V + E)$.
- line 2 takes $\Theta(V)$.
- **for** loop of lines 3-5 makes one iteration each vertex.
- Altogether, the for loop of lines 4–5 relaxes each edge exactly once.
- Because each iteration of the inner for loop takes $\Theta(1)$ time, the total running time is $\Theta(V + E)$, which is linear in the size of an adjacency-list representation of the graph.