
24.1 The 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 $w : E \rightarrow \mathbb{R}$, 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 relaxes edges, progressively decreasing an estimate $v.d$ on the weight of a shortest path from the source s to each vertex $v \in V$ until it achieves the actual shortest-path weight $\delta(s, v)$. The algorithm returns TRUE if and only if the graph contains no negative-weight cycles that are reachable from the source.

```

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

```

Figure 24.4 shows the execution of the Bellman-Ford algorithm on a graph with 5 vertices. After initializing the d and π values of all vertices in line 1, the algorithm makes $|V| - 1$ passes over the edges of the graph. Each pass is one iteration of the **for** loop of lines 2–4 and consists of relaxing each edge of the graph once. Figures 24.4(b)–(e) show the state of the algorithm after each of the four passes over the edges. After making $|V| - 1$ passes, lines 5–8 check for a negative-weight cycle and return the appropriate boolean value. (We’ll see a little later why this check works.)

The Bellman-Ford algorithm runs in time $O(VE)$, since the initialization in line 1 takes $\Theta(V)$ time, each of the $|V| - 1$ passes over the edges in lines 2–4 takes $\Theta(E)$ time, and the **for** loop of lines 5–7 takes $O(E)$ time.

To prove the correctness of the Bellman-Ford algorithm, we start by showing that if there are no negative-weight cycles, the algorithm computes correct shortest-path weights for all vertices reachable from the source.

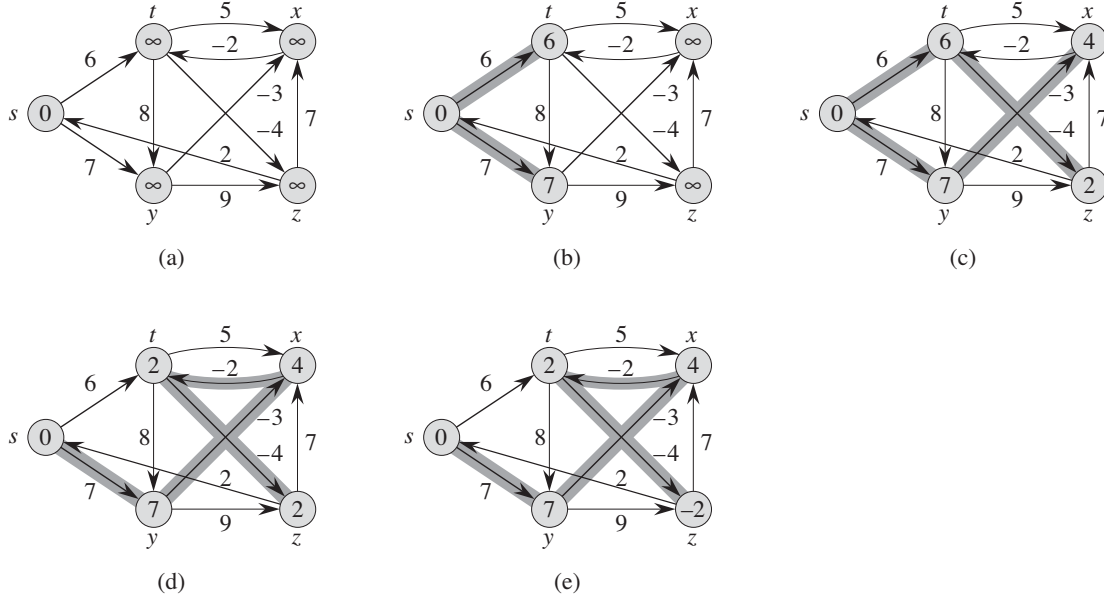


Figure 24.4 The execution of the Bellman-Ford algorithm. The source is vertex s . The d values appear within the vertices, and shaded edges indicate predecessor values: if edge (u, v) is shaded, then $v.\pi = u$. In this particular example, each pass relaxes the edges in the order (t, x) , (t, y) , (t, z) , (x, t) , (y, x) , (y, z) , (z, x) , (z, s) , (s, t) , (s, y) . (a) The situation just before the first pass over the edges. (b)–(e) The situation after each successive pass over the edges. The d and π values in part (e) are the final values. The Bellman-Ford algorithm returns TRUE in this example.

Lemma 24.2

Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w : E \rightarrow \mathbb{R}$, 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 .

Proof We prove the lemma by appealing to the path-relaxation property. Consider any vertex v that is reachable from s , and let $p = \langle v_0, v_1, \dots, v_k \rangle$, 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 $i = 1, 2, \dots, k$, is (v_{i-1}, v_i) . By the path-relaxation property, therefore, $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$. ■

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 The proof is left as Exercise 24.1-2. ■

Theorem 24.4 (Correctness of the Bellman-Ford algorithm)

Let BELLMAN-FORD be run on a weighted, directed graph $G = (V, E)$ with source s and weight function $w : E \rightarrow \mathbb{R}$. 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 cycle reachable from s , then the algorithm returns FALSE.

Proof 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. 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 edges $(u, v) \in E$,

$$\begin{aligned} v.d &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \quad (\text{by the triangle inequality}) \\ &= u.d + w(u, v), \end{aligned}$$

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

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

$$\sum_{i=1}^k w(v_{i-1}, v_i) < 0. \tag{24.1}$$

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 $i = 1, 2, \dots, k$. Summing the inequalities around cycle c gives us

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

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

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

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

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

which contradicts inequality (24.1). We conclude that the Bellman-Ford algorithm returns TRUE if graph G contains no negative-weight cycles reachable from the source, and FALSE otherwise. ■

Exercises

24.1-1

Run the Bellman-Ford algorithm on the directed graph of Figure 24.4, using vertex z as the source. In each pass, relax edges in the same order as in the figure, and show the d and π values after each pass. Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.

24.1-2

Prove Corollary 24.3.

24.1-3

Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v . (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

24.1-4

Modify the Bellman-Ford algorithm so that it sets $v \cdot d$ to $-\infty$ for all vertices v for which there is a negative-weight cycle on some path from the source to v .