Introduction To Algorithms

CLRS

May 25, 2022

Contents

1 Graph Algorithms

1.1 Elementary Graph Algorithms

1.1.1 Topological sort

```
procedure TOPOLOGICAL-SORT(G)
    call DFS(G) to compute finishing times v.f for each vertex v
    as each vertex is finished, insert it onto the front of a linked list
    return the linked list of vertices
end procedure
```

We can perform a topological sort in time $\Theta(V + E)$, since depth-first search takes $\Theta(V + E)$ time and it takes $O(1)$ time to insert each of the $|V|$ vertices onto the front of the linked list

Exercise 1.1.1 (22.4-3). Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a simple cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$

Proof. If the graph is acylic, then $|E| \leq |V| - 1$ and we can run DFS in $O(|V|)$. If there is a path going back, then at should end in $|V|$th step □

Exercise 1.1.2 (22.4-5). Another way to perform topological sorting on a directed acylic graph $G = (V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $O(V + E)$. What happens to this algorithm if G has cycles?

Proof. □

1.2 Single-Source Shortest Paths

procedure INITIALIZE-SINGLE-SOURCE(G, s)
    for $v \in G.V$ **do**
        $v.d = \infty$
        $v.\pi = nil$
    end for
    $s.d = 0$
end procedure

procedure RELAX(u, v, w)
    if v.d≥u.d+w(u,v) **then**
        $v.d = u.d + w(u, v)$
        $v.\pi = u$
    end if
end procedure

1.2.1 The Bellman-Ford algorithm

procedure INITIALIZE-SINGLE-SOURCE(G, s)
    for $i = 1$ to $|G, V| - 1$ **do**
        for $(u, v) \in G.E$ **do**
            RELAX(u, v, w)
        end for
    end for
    for each edge $(u, v) = G.E$ **do**
        if $v.d > u.d + w(u, v)$ **then**
            return False
        end if
    end for
end procedure

Lemma 1.1. *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*** *loops, we have $v.d = \delta(s, v)$ for all vertices v that are reachable from s*

Proof. 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 $v_k = v$ to 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 relaxes all $|E|$ edges. Among the edges relaxed in the ith iteration, for $i = 1, \ldots, k$, is (v_{i-1}, v_i). By the path-relaxation property, therefore $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$ □

Corollary 1.2. *Let $G = (V, E)$ be a weighted, directed graph with source vertex s and weight function $w : E \to \mathbb{R}$, 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 iff BELLMAN-FORD terminates with $v.d < \infty$ when it is run on G*

Theorem 1.3 (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 $w : E \to \mathbb{R}$. If G contains no negative-weight cycles that are reachable from s, then the algorithm return TRUE, we have $v.d = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-path tree rooted at s. If G does contain a negative-weight cycle reachable from s, then the algorithm returns FALSE*

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

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

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

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

But since $\sum_{i=1}^{k} v_i.d = \sum_{i=1}^{k} v_{i-1}.d$, we have

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

□

Exercise 1.2.1.

1.2.2 Single-source shortest paths in directed acyclic graphs

By relaxing the edges of a weighted dag $G = (V, E)$ according to a topological sort of its vertices, we can compute shortest paths from a single source in $\Theta(V + E)$ time. Shortest paths are always well defined in a dag

procedure DAG-SHORTEST-PATHS(G, w, s)
    topological sort the vertices of G
    INITIALIZE-SINGLE-SOURCE(G, s)
    for each vertex u, taken in topological sorted order **do**
        for each vertex $v \in G.Adj[u]$ **do** RELAX(u, v, w)
        end for
    end for
end procedure

Exercise 1.2.2 (24.2-4). Given an efficient algorithm to count the total number of paths in a directed acylic graph. Analyze your algorithm

1.2.3 Dijkstra's algorithm

Dijkstra's algorithm solves the single-source shortest-paths problem on a weighted, directed graph $G = (V, E)$ for the case in which all edge weights are nonnegative.

procedure DIJKSTRA(G, w, s)
    $S = \emptyset$
    $Q = G.V$
    while $Q \neq \emptyset$ **do**
        $u =$EXTRACT-MIN(Q)
        $S = S \cup \{u\}$
        for each vertex $v \in G.Adj[u]$ **do** RELAX(u, v, w)
        end for
    end while
end procedure

1.2.4 Proofs of shortest-paths properties

Lemma 1.4 (Triangle inequality)**.** *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$ and source vertex s. Then for all edges $u, v) \in E$ we have*

$$\delta(s, v) \leq \delta(s, u) + w(u, v)$$

Lemma 1.5 (Upper-bound property)**.** *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$. Let $s \in V$ be the source vertex, and let the graph be initialized by INITIALIZE-SINGLE-SOURCE(G, s). Then $v.d \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps on the edges of G. Moreover, once $v.d$ achieves its lower bound $\delta(s, v)$ it never changes*

Proof. By the inductive hypothesis, $x.d \geq \delta(s, x)$ for all $x \in V$ prior to the relaxation. The only d that may change is $v.d$. If it changes, we have

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

□

Corollary 1.6 (No-path property)**.** *Suppose that in a weighted, directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, no path connects a source vertex $s \in V$ to a given vertex $v \in V$. Then, after the graph is initialized by INITIALIZE-SINGLE-SOURCE(G, s), we have $v.d = \delta(s, v) = \infty$ and this equality is maintained as an invariant over any sequence of relaxation steps on the edges of G*

Proof. By the upper-bound property, we always have $\infty = \delta(s, v) \leq v.d$ □

Lemma 1.7. *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$, and let $(u, v) \in E$. Then immediately after relaxing edge (u, v) by executing RELAX(u, v, w), we have $v.d \leq u.d + w(u, v)$*

Proof. If prior to relaxing edge (u, v), we have $v.d > u.d + w(u, v)$, then $v.d = u.d + w(u, v)$ afterward. Otherwise $v.d$ doesn't change □

Lemma 1.8 (Convergence property)**.** *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$, let $s \in V$ be a source vertex, and let $s \rightsquigarrow u \to v$ be a shortest path in G for some vertices $u, v \in V$. Suppose G is initialized by INITIALIZE-SINGLE-SOURCE(G, s) and then a sequence of relaxation steps that includes the call RELAX(u, v, w) is executed on the edges of G. If $u.d = \delta(s, u)$ at any time prior to the call, then $v.d = \delta(s, v)$ at all times after the call*

Proof. □

Lemma 1.9 (Path-relaxation property). *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$, and let $s \in V$ be a source vertex. Consider any shortest path $p = \langle v_0, \dots, v_k \rangle$ from $s = v_0$ to v_k. If G is initialized by INITIALIZE-SINGLE-SOURCE(G, s) and then a sequence of relaxation steps occurs that includes, in order, relaxing the edges $(v_0, v_1), \dots, (v_{k-1}, v_k)$ then $v_k.d = \delta(s, v_k)$ after these relaxations and at all times after wards.*