

Introduction To Algorithms

CLRS

December 28, 2022

Contents

1	Graph Algorithms	1
1.1	Elementary Graph Algorithms	1
1.1.1	Topological sort	1
1.2	Minimum Spanning Trees	2
1.2.1	Growing a minimum spanning tree	2
1.2.2	The algorithms of Kruskal and Prim	3
1.3	Single-Source Shortest Paths	4
1.3.1	The Bellman-Ford algorithm	4
1.3.2	Single-source shortest paths in directed acyclic graphs	6
1.3.3	Dijkstra's algorithm	6
1.3.4	Proofs of shortest-paths properties	7

1 Graph Algorithms

1.1 Elementary Graph Algorithms

1.1.1 Topological sort

- 1: **procedure** TOPOLOGICAL-SORT(G)
- 2: call DFS(G) to compute finishing times $v.f$ for each vertex v
- 3: as each vertex is finished, insert it onto the front of a linked list
- 4: **return** the linked list of vertices
- 5: **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 acyclic, then $|E| \leq |V| - 1$ and we can run DFS in $O(|V|)$. If there is a path going back, then it should end in $|V|$ th step \square

Exercise 1.1.2 (22.4-5). Another way to perform topological sorting on a directed acyclic 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. \square

1.2 Minimum Spanning Trees

1.2.1 Growing a minimum spanning tree

```

1: procedure GENERIC-MST( $G, w$ )
2:    $A = \emptyset$ 
3:   while  $A$  does not form a spanning tree do
4:     find an edge  $(u, v)$  that is safe for  $A$ 
5:      $A = A \cup \{(u, v)\}$ 
6:   end while
7:   return  $A$ 
8: end procedure

```

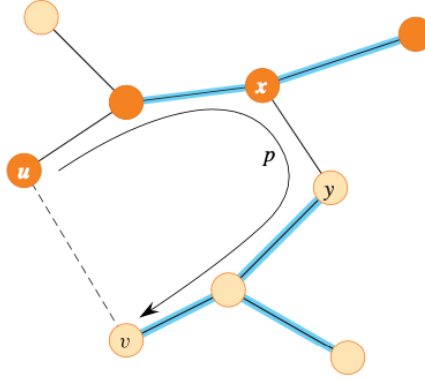
We call an edge **safe** if it can be added safely to A while maintaining the invariant

A **cut** $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V . Edge $(u, v) \in E$ **crosses** the cut $(S, V - S)$ if one of its endpoints belongs to S and the other belongs to $V - S$. A cut **respects** a set A of edges if no edge in A crosses the cut. A edge is a **light edge** crossing a cut if its weight is the minimum of any edge crossing the cut.

Theorem 1.1. Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E . Let A be a subset of E that is included in some minimum spanning tree for G , let $(S, V - S)$ be any cut of G that respects A , and let (u, v) be a light edge crossing $(S, V - S)$. Then edge (u, v) is safe for A .

Proof. Let T be a minimum spanning that includes A , and assume that T does not contain the light edge (u, v) , since if it does, we are done. We'll

construct another minimum spanning tree T' that includes $A \cup \{(u, v)\}$ by using a cut-and-paste technique, thereby showing that (u, v) is a safe edge for A .



The edge (u, v) forms a cycle with the edges on the single path p from u to v in T . Since u and v are on opposite sides of the cut $(S, V - S)$, at least one edge in T lies on the simple path p and also crosses the cut. Let (x, y) be any such edge. The edge (x, y) is not in A , because the cut respects A . Since (x, y) is on the unique simple path from u to v in T , removing (x, y) breaks T into two components. Adding (u, v) reconnected them to form a new spanning tree $T' = (T - \{(x, y)\}) \cup \{(u, v)\}$.

We next show that T' is a minimum spanning tree. Since (u, v) is a light edge crossing $(S, V - S)$ and (x, y) also crosses this cut, $w(u, v) \leq w(x, y)$. Therefore $w(T') \leq w(T)$. \square

Corollary 1.2. *Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E . Let A be a subset of E that is included in some minimum spanning tree for G , and let $C = (V_C, E_C)$ be a connected component in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting C to some other component in G_A , then (u, v) is safe for A .*

1.2.2 The algorithms of Kruskal and Prim

1. Kruskal's algorithm

- 1: **procedure** MST-KRUSKAL(G, w)
- 2: $A = \emptyset$
- 3: **for** each vertex $v \in G.V$ **do**

```

4:      Make-Set( $v$ )
5:  end for
6:  create a single list of the edges in  $G.E$ 
7:  sort the list of edges into monotonically increasing order by
    weight  $w$ 
8:  for each edge  $(u, v)$  taken from the sorted list in order do
9:      if Find-Set( $u$ )  $\neq$  Find-Set( $v$ ) then
10:          $A = A \cup \{(u, v)\}$ 
11:         Union( $u, v$ )
12:      end if
13:  end for
14:  Return  $A$ 
15: end procedure

```

1.3 Single-Source Shortest Paths

```

1: procedure INITIALIZE-SINGLE-SOURCE( $G, s$ )
2:   for  $v \in G.V$  do
3:      $v.d = \infty$ 
4:      $v.\pi = nil$ 
5:   end for
6:    $s.d = 0$ 
7: end procedure

1: procedure RELAX( $u, v, w$ )
2:   if  $v.d \geq u.d + w(u, v)$  then
3:      $v.d = u.d + w(u, v)$ 
4:      $v.\pi = u$ 
5:   end if
6: end procedure

```

1.3.1 The Bellman-Ford algorithm

```

1: procedure INITIALIZE-SINGLE-SOURCE( $G, s$ )
2:   for  $i = 1$  to  $|G, V| - 1$  do
3:     for  $(u, v) \in G.E$  do
4:       RELAX( $u, v, w$ )
5:     end for
6:   end for
7:   for each edge  $(u, v) \in G.E$  do
8:     if  $v.d > u.d + w(u, v)$  then

```

```

9:         return False
10:     end if
11: end for
12: end procedure

```

Lemma 1.3. *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** 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 $p = \langle v_0, v_1, \dots, v_k \rangle$ 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 i th iteration, for $i = 1, \dots, k$, is (v_{i-1}, v_i) . By the path-relaxation property, therefore $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$ \square

Corollary 1.4. *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 iff BELLMAN-FORD terminates with $v.d < \infty$ when it is run on G*

Theorem 1.5 (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 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 $c = \langle v_0, \dots, v_k \rangle$, 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 $i = 1, \dots, k$. Summing

the inequalities around cycle c gives us

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

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

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

□

Exercise 1.3.1.

1.3.2 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

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

Exercise 1.3.2 (24.2-4). Given an efficient algorithm to count the total number of paths in a directed acyclic graph. Analyze your algorithm

1.3.3 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.

- 1: **procedure** DIJKSTRA(G, w, s)
- 2: $S = \emptyset$

```

3:    $Q = G.V$ 
4:   while  $Q \neq \emptyset$  do
5:        $u = \text{EXTRACT-MIN}(Q)$ 
6:        $S = S \cup \{u\}$ 
7:       for each vertex  $v \in G.Adj[u]$  do  $\text{RELAX}(u, v, w)$ 
8:       end for
9:   end while
10: end procedure

```

Theorem 1.6 (Correctness of Dijkstra’s algorithm). *Dijkstra’s algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s , terminates with $u.d = \delta(s, u)$ for all vertices $u \in V$*

Proof. Let u be the first vertex for which $u.d \neq \delta(s, u)$ when it is added to set S . Then $u \neq s$ and $\delta(s, u) \neq \infty$. 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 . Let us consider the first vertex y along p s.t. $y \in V - S$, and let $x \in S$ be y ’s predecessor along p . We can decompose path p into $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} u$

We claim that $y.d = \delta(s, y)$ when u is added to S . But y should be chosen after x □

Exercise 1.3.3.

1.3.4 Proofs of shortest-paths properties

Lemma 1.7 (Triangle inequality). *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \mathbb{R}$ 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 1.8 (Upper-bound property). *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \mathbb{R}$. Let $s \in V$ be the source vertex, and let the graph be initialized by $\text{INITIALIZE-SINGLE-SOURCE}(G, s)$. 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 1.9 (No-path property). *Suppose that in a weighted, directed graph $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{R}$, 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 1.10. *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \mathbb{R}$, 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 1.11 (Convergence property). *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \mathbb{R}$, let $s \in V$ be a source vertex, and let $s \rightsquigarrow u \rightarrow v$ 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 1.12 (Path-relaxation property). *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \mathbb{R}$, 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.*