PROOFS OF SHORTEST-PATHS PROPERTIES

Lemma 1. Subpaths of shortest paths are shortest paths.

Theorem 2. 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 $\delta(s, v) \leq \delta(s, u) + w((u, v))$

Proof. Let $p = \langle s, \ldots, v \rangle$ be the shortest path from s to v. Then p has length $\delta(s, v)$ and since p is the shortes path $\delta(s, v)$ is less than or equal to the length of any other path from s to v. In particular the path that includes the shortest path from s to u plus the edge from u to v. If there is no path from s to v then define $\delta(s, v) \equiv -\infty$. □

Lemma 3. Upper-bound property. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$ and source vertex s and initialize the graph with

Algorithm 1 *Initialize-Single-Source*

$Initialize-Single-Source(G, s)$
for $each\ v \in V$
    $v.d = \infty$
    $v.\pi = NIL$
$s.d = 0$

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. Proof by induction on the number of relaxation steps:

Algorithm 2 Relaxation

Relax(u, v, w)
if $v.d > u.d + w((u, v))$
    $v.d = u.d + w((u, v))$
    $v.\pi = u$

For the base case $v.d \geq \delta(s, v)$ for all $v \in V$ after the initialization step since $v.d = \infty$ for all $v \in V$ after the initialization step. The inductive hypothesis is $x.d \geq \delta(s, x)$ for $x \in V$ after the previous relaxation step. During the relaxation of (u, v) only $v.d$ might change, and if it does then it's set to $u.d + w((u, v))$. But $u.d \geq \delta(s, u)$ so

$$v.d \geq \delta(s, u) + w((u, v)) \geq \delta(s, v)$$

by the triangle inequality.

Note that once $v.d = \delta(s, v)$ it cannot decrease further by the above inequality and it cannot increase because relaxation only decreases $v.d$. □

Corollary 4. If there is not path from s to v then $\delta(s, v) = v.d = \infty$ always.

Lemma 5. Immediately after relaxing edge (u, v) by executing Relax(u, v, w), $v.d \leq u.d + w((u, v))$. Basically after at least one relaxation $v.d$ will never be larger than the distance to u plus the distance from $\underline{u}$ to v.

Proof. If $v.d > u.d + w((u, v))$ before relaxation then $v.d = u.d + w((u, v))$ after relaxation. Otherwise if $v.d \leq u.d + w((u, v))$ before relaxation then Relax(u, v, w) is nilpotent and so $v.d \leq u.d + w((u, v))$ remains. □

Theorem. ***Convergence property***. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$ and source vertex s, and $s \rightsquigarrow u \to v$ be a shortest path for some vertices $u, v \in V$. Suppose G is initialized using Initialize-Single-Source(G, s) and then a sequence of relaxation steps is performed. If $u.d = \delta(s, u)$ at any time prior calling Relax(u, v, w) then $v.d = \delta(s, v)$ at all times afterwards.

Proof. By the upper-bound property if $u.d = \delta(s, u)$ prior to relaxing (u, v) then it continues to hold thereafter. Hence after relaxing (u, v) we have $v.d \leq u.d + w((u, v))$ by Lemma 5 and

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

by subpaths of shortest paths are shortest paths[1]. Again by Lemma 5 $v.d \geq \delta(s, v)$ and so $v.d = \delta(s, v)$, and the inequality persists thereafter. □

Theorem. ***Path-relaxation property***. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$ and source vertex s. Consider any shortest path $p = \langle v_0 = s, \ldots, v_k = v\rangle$. Suppose G is initialized using Initialize-Single-Source(G, s) and then a sequence of relaxation steps is performed that includes relaxing $(v_0, v_1), \ldots, (v_1, v_2), \ldots, (v_{k-1}, v_k)$ (i.e. the edges (v_i, v_j) are relaxed in order). Then $v.d = \delta(s, v)$ after the relaxations and persists.

Proof. The proof is by induction on the ith edge in p to be relaxed. For the base case $v_0.d = s.d = 0 = \delta(s, s)$. By the upper-bound property $s.d$ never changes.

For the inductive step, assume that $v_{i-1}.d = \delta(s, v_{i-1})$. By the convergence property, after relaxing (v_{i-1}, v_i) it's the case that $v_i.d = \delta(s, v_i)$. □

Theorem 6. ***Shortest-paths trees***. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}$ and source vertex s and assume G contains no negative-weight cycles reachable from s. Then after initialization the predecessor subgraph G_π forms a rooted tree with root s, and any sequence of relaxation steps on edges of G maintain this property as an invariant.

Summary:

- Triange inequality: $\delta(s, v) \leq \delta(s, u) + w((u, v))$
- Upper-bound property: $v.d \geq \delta(s, v)$

[1] Recall that by assumption $s \rightsquigarrow u \to v$ is a shortest path from s to v.

- Convergence property: $s \rightsquigarrow u \to v$ a shortest path and $u.d = \delta(s,u)$ prior to relaxing (u,v) then $v.d = \delta(s,v)$ afterward
- Path-relaxation property: $p = \langle s = v_0, \ldots, v_k = v\rangle$ a shortest path and edges are relaxing in order then $v.d = \delta(s,v)$.
- Predecessor-subgraph property: once $v.d = \delta(s,v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s.

Exercise 7 (24.4-9)**.** This is a dumb problem but for completeness I'm writing it up[2] . First of all $\max\{x_i\} = 0$. Why? Let $p = \langle s, v_1, \ldots, v_k\rangle$ be the shortest path from the source to any vertex v_k returned by Bellman-Ford. By optimal substructure $p' = \langle s, v_1\rangle$ is the shortest path from the source to v_1. But by construction that edge has weight 0 and therefore $x_1 = 0$. Finally since the $x_i \leq 0$ it's the case that $\max\{x_i\} = 0$. Now all that remains is to show that Bellman-Ford maximizes $\min\{x_i\}$.

Let $x_k = \min\{x_i\}$ and consider $p = \langle s, v_1, \ldots, v_k\rangle$. The weight of the path is just the sum of the constraints along the path

$$\begin{aligned} x_1 - x_0 &\leq 0 \\ x_2 - x_1 &\leq C_{1,2} \\ &\vdots \\ x_k - x_{k-1} &\leq C_{k-1,k} \end{aligned}$$

Summing we get $x_k \leq \sum_{i=1}^{k-1} C_{i,i+1} = w(p)$ the distance from source to x_k. Therefore any solution x_k that satisfies the constraints cannot be greater than $w(p)$, the shortest distance from source to v_k. Since Bellman-Ford sets $x_k = w(p)$ it does indeed maximize x_k. I'm guessing that this same reasoning can be applied to argue that Bellman-Ford maximizes $\sum x_k$.

Problem 8 (24-1)**.** Yen's improvement to Bellman-Ford

(a) Towards a contradiction assume that G_f has a cycle. Then there exists a sequence of edges

$$\{(v_a, v_b), (v_c, v_d), \ldots, (v_j, v_a)\} \subset E_f$$

but by definition $(v_j, v_a) \in E_b$ not in E_f. Similarly for G_b. Topological sort follows from definition too: only edges that "flow" in the corresponding direction exist in each of E_f, E_b.

(a) Stolen from[3]. In the first part of any iteration, any path in the shortest path tree that starts at a finished vertex and consists of edges only in E_f becomes correctly labeled (shortest path in DAG is easy - just relax in order of topological sort), i.e. all vertices become finished. In the second part of any iteration, any path in the shortest path tree that starts at a finished vertex and consists of edges only in E_b becomes correctly labeled. The number of iterations needed then is the maximum alternations between G_f and G_b, which is $(|V|+1)/2$, since G_f and G_b partition G in half. Another optimization is you don't need to relax edges coming out of a

[2]Answer stolen from https://dspace.mit.edu/bitstream/handle/1721.1/37150/6-046JFall-2004/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-046JFall-2004/EC0B786D-A562-4952-B52A-7A4DCEB60908/0/ps6sol.pdf

[3]http://11011110.livejournal.com/215330.html

vertex that wasn't updated in the previous iteration. A final optimization is randomizing the linear ordering to prevent shortest paths that alternate between E_f and E_b, bringing down the number of iterations to $|V|/3$ on average.

Problem 9 (24-2)**.** Nesting boxes

(a) Well duh. If box A can contain box B and box B can contain box C then obviously A can contain C.
(b) Sort both sets of dimensions and if one set is entry-wise prior then it fits.
(c) Do the n^2 comparisons to figure out all (B_i, B_j), where B_i can contain B_j. Then do a topological sort on the graph because it's a DAG (because what would it mean for a cycle of box containments). Then find the longest path using DP:

Algorithm 3 Longest Path

```
Longest-Path(G)
T = Top-Sort(G)
for v ∈ T
      v.d = max_{(u,v)∈E} {u.d + 1}
return max_{v∈V} {v.d}
```

Problem 10 (24-3)**.** Arbitrage

(a) Take log. Then Bellman-Ford will find negative cycles.

Problem 11 (24-6)**.** Bitonic shortest paths. Sort the edges by weight. Then relax increasing order, decreasing order, then increasing again and you're done