

Lecture #11:

0.0.1 Graph Algorithms: Shortest Paths (Proofs)

Single Origin Problems: Representation:

For each node $v \in V$, a parent (predecessor) $\pi[v]$ (which is either another node or NIL) is maintained. The algorithms mentioned set these π values so that we have a chain of predecessors that go to the origin for each node for which $\pi \neq \text{NIL}$. This gives us the required shortest path. This creates a predecessor (directed) subgraph $G_\pi = (V_\pi, E_\pi)$ defined as follows:

$$\begin{aligned} V_\pi &= \{v \in V : \pi[v] \neq \text{NIL}\} \\ E_\pi &= \{(\pi[v], v) \in E : v \in V - \{s\}\} \end{aligned}$$

We will prove that this graph is tree rooted at s and is called a **shortest-path-tree**. There may be more than one of these.

- Its nodes V_π are those that are "reachable" from s
- Its root is s
- For all nodes $v \in V_\pi$, the unique path from s to v in G_π is a shortest path from s to v .

Lemma 1 (25.1) Let $p = \langle v_0, v_1, \dots, v_k \rangle$ be a shortest path from v_0 to v_k in $G = [V, E]$. For any i and j such that $1 \leq i \leq j \leq k$, let $p_{i,j} = \langle v_i, \dots, v_j \rangle$ be a sub-path from v_i to v_j . Then $p_{i,j}$ is a shortest path between these vertices.

Proof by contradiction. If $p_{i,j}$ is not a shortest path between i and j , let $p'_{i,j}$ be a shorter path. Then, $p' = \langle v_0, v_1, \dots, v_{i-1} \rangle \cdot p'_{i,j} \cdot \langle v_{j+1}, \dots, v_j \rangle$ is a shorter path between v_0 and v_k .

Corollary 2 (25.2) Let a shortest path p between s and v be broken into $p_{s,u}$ and the edge (u, v) . Then $\delta(s, v) = \delta(s, u) + w(u, v)$.

Follows directly from the above lemma.

Lemma 3 (25.3) For all edges $(u, v) \in E$, $\delta(s, v) \leq \delta(s, u) + w(u, v)$

Proof: One path from s to v is $p_{s,u} \cdot (u, v)$ where $p_{s,u}$ is a shortest path from s to u . Hence the result follows.

Properties of Relaxation:

Lemma 4 (25.4) Immediately after relaxing edge (u, v) by executing $\text{RELAX}(u, v, w)$, we have $d[v] \leq d[u] + w(u, v)$.

Lemma 5 (25.5) *Assume that we initialize as shown above. Then, the relaxation algorithm maintains the invariant that $d[v] \geq \delta(s, v)$ for all $v \in V$. Moreover, if equality is achieved at any step, it does not change.*

Proof: Since $d[s] = 0 \geq \delta(s, s)$ (if there is a negative cycle with s , then $\delta(s, s) = -\infty$) and $d[v] = \infty$ at the initial step, the above condition is satisfied in this step. If the condition does not hold at some point, let the first time this violation happens be after edge (u, v) is relaxed and the values before this relaxation are represented by $d[u]$ and $d[v]$ for these two nodes. Thus, $d^{after}[v] < \delta(s, v)$.

$$\begin{aligned} d^{after}[v] &= d^{before}[u] + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \quad \text{by lemma 25.3} \end{aligned}$$

This implies, $d^{before}[u] < \delta(s, u)$ – a contradiction to the assumption that the first violation happens with v . Hence, the invariant holds.

For the rest of the lemma, we observe that the values $d[v]$ never increase. So once equality is achieved in the relation it is maintained thereafter.

Corollary 6 (25.6) *If a node v can not be reached from s , then $d[v] = \delta(s, v)$ throughout the algorithm.*

Lemma 7 (25.7) *If at any time before relaxing edge (u, v) , $d[u] = \delta(s, u)$, and there is a shortest path from s to v , then after this relaxation, $d[v] = \delta(s, v)$.*

Dijkstra Algorithm:

Claim: For each vertex $v \in V$, at the time when v is inserted into S , we have $d[v] = \delta(s, v)$ and this equality is maintained after this point.

Proof: Suppose not; let u be the **first** vertex (in the order of insertion into S) for which $d[u] \neq \delta(s, u)$.

1. $u \neq s$: because when s is inserted into S (at the initial step), $d[s] = 0 = \delta(s, s)$
2. Just prior to the time of insertion of u into S , $S \neq \phi$
3. Since at the time of insertion of u into S , $d[u] \neq \delta(s, u)$, $d[u]$ is finite and hence there is a path from s to u in G . Hence, there is a shortest path $p_{s,u}$ from s to u in G . (This is because the number of relevant paths is finite)
4. $p_{s,u}$ connects $s \in S$ with $u \notin S$ (this is just before u is inserted into S).
5. Let the first vertex in $p_{s,u}$ not in S (this is just before u is inserted into S) be y and let the vertex in $p_{s,u}$ before y be x . Hence $x \in S$ and the part of the path $p_{s,u}$ from s to x is entirely in S . Actually this path looks like:

$$s \rightsquigarrow x \longrightarrow y \rightsquigarrow u$$

6. Claim#2: Just prior to inserting u into S , $d[y] = \delta(s, y)$

Proof: At this time, $x \in S$ and since u is the first vertex for which $d[u] \neq \delta(s, u)$ at the time of insertion into S , and x was inserted prior to this time, $d[x] = \delta(s, x)$

Edge (x, y) was relaxed at the time x was inserted into S (which is prior to when u was inserted into S). Hence, $d[y] = \delta(s, y)$ by lemma 25.7 at the time this edge was relaxed. Hence the claim.

7. Since y occurs before u in the path $p_{s,u}$, we have

$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$$

The last of inequalities by lemma 25.5. Since the algorithm chose to insert u and not y at this point in time, we have $d[u] \leq d[y]$. Therefore, $d[u] = d[y] = \delta(s, y) = \delta(s, u)$. But this contradicts the main assumption regarding the choice of u . Hence claim #1.

8. Equality holds thereafter by lemma 25.7.

Bellman-Ford Algorithm:

Lemma 8 (25.12) *If G contains no negative cycles that are reachable from s , then at termination of this algorithm $d[v] = \delta(s, v)$ for all nodes $v \in V$ that are reachable from s .*

Proof: Since there are no negative cycles reachable from s , there is a (simple) shortest path from s to all nodes v that can be reached from s . Let v be such a node and let $p_{s,v} = \langle s = v_0, v_1, \dots, v_k = v \rangle$ be a shortest path from s to v . Since this is a simple path, $k \leq |V| - 1$. We prove the lemma by induction on the number of passes that the algorithm makes. In particular, that $d[v_i] = \delta(s, v_i)$ after the i^{th} pass of the algorithm for $i = 0, 1, \dots, k$. Since there are $|V| - 1$ passes in all, we have the lemma if we prove the above statement.

$d[s] = d[v_0] = \delta(s, s)$ after initialization (=0th step) if there are no negative cycles. Hence, this equality is maintained thereafter.

Assume by IH that $d[v_{i-1}] = \delta(s, v_{i-1})$ after $(i-1)^{\text{st}}$ pass. Edge (v_{i-1}, v_i) is relaxed in the i^{th} pass (as is every edge) and by lemma 25.7, we have $d[v_i] = \delta(s, v_i)$ after this relaxation and this is maintained thereafter. This completes the proof that this algorithm works.

This also shows that if one more pass changes some of these values, then there must be a negative cycle that can be reached from s .