Single-Source Shortest Paths

Problem: To find the shortest path from point A to point B on a map. You are given a map with distances between adjacent nodes already marked.

Solution using brute force: Enumerate the total distance using all paths (eliminating cycles), and select the shortest. Leads to a combinatorial explosion of possibilities.

Shortest-path problem

- Given a weighted directed graph $G = (V, E)$
- Weight function $w = E \rightarrow \Re$ to map edges to real-valued weights
- Weight of path $p = \langle v_0, v_1, \ldots, v_k \rangle$

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

- Shortest path weight from u to v

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\leadsto} v\} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- A *shortest path* from vertex u to vertex v is defined as any path p with weight $w(p) = \delta(u, v)$.
  - Breadth-first search algorithm is an example of shortest path algorithm that works on unweighted graphs
    * Each edge is considered to be of unit weight
- Representing shortest paths
  - For each vertex $v \in V$, maintain a predecessor $\pi[v]$
  - Find *predecessor subgraph* $G_\pi = (V_\pi, E_\pi)$ induced by π values
  - A *shortest-paths tree* rooted at s is a directed subgraph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, such that
    1. V' is the set of vertices reachable from s in G
    2. G' forms a rooted tree with s
    3. For all $v \in V'$, the unique simple path from s to v in G' is a shortest path from s to v in G

Shortest Paths and Relaxation

- Repeatedly decrease an upper bound on the actual shortest-path weight of each vertex until the upper bound equals the shortest-path weight
- Optimal substructure of a shortest path
  - A shortest path between two vertices contains other shortest paths within it

    Lemma 1 (Subpaths of shortest paths are shortest paths.) *Given a weighted, directed graph $G = (V, E)$ with weight function $w : E \rightarrow \Re$, let $p = \langle v_1, v_2, \ldots, v_k \rangle$ be a shortest path from vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the subpath of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.*

    Corollary 1 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$. Suppose that a shortest path p from a source s to a vertex v can be decomposed into $s \overset{p'}{\leadsto} u \rightarrow v$ for some vertex u and path p'. Then, the weight of a shortest path from s to v is $\delta(s, v) = \delta(s, u) + w(u, v)$.*

    Lemma 2 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$ and source vertex s. Then, for all edges $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.*

- Relaxation
  - For each vertex $v \in V$, maintain an attribute $d[v]$
  - $d[v]$ – upper bound on the weight of a shortest path from source s to v – *shortest path estimate*
  - Initialization procedure

```
initialize_single_source (G,s)
   for each vertex v ∈ V[G] do
      d[v] ← ∞
      π[v] ← nil
   d[s] ← 0
```

  - Relaxation

```
relax (u,v,w)
   if d[v] > d[u] + w(u,v) then
      d[v] ← d[u] + w(u,v)
      π[v] ← u
```

- Properties of relaxation

Lemma 3 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$, and let $(u, v) \in E$. Then, immediately after relaxing edge (u, v) by executing* `relax`(u, v, w)*, we have $d[v] \leq d[u] + w(u, v)$.*

Lemma 4 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$. Let $s \in V$ be the source vertex, and let the graph be initialized by* `initialize_single_source`(G, s)*. Then, $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps on the edges of G. Moreover, once $d[v]$ achieves its lower bound $\delta[s, v]$, it never changes.*

Corollary 2 *Suppose that in a weighted, directed graph $G = (V, E)$ with weight function $w : E \rightarrow \Re$, 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 $d[v] = \delta(s, v)$, and this equality is maintained as an invariant over any sequence of relaxation steps on the edges of G.*

Lemma 5 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$, 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 that 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 $d[u] = \delta(s, u)$ at any time prior to the call, the $d[v] = \delta(s, v)$ at all times after the call.*

- Shortest-paths trees

Lemma 6 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$ and source vertex $s \in V$, and assume that G contains no negative-weight cycles that are reachable from s. Then, after the graph is initialized by* `initialize_single_source`(G, s)*, 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.*

Lemma 7 *Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \rightarrow \Re$ and source vertex $s \in V$, and assume that G contains no negative-weight cycles that are reachable from s. Let us call* `initialize_single_source`(G, s) *and then execute any sequence of relaxation steps on edges of G that produces $d[v] = \delta(s, v)$ for all $v \in V$. Then, the predecessor subgraph G_π is a shortest-paths tree rooted at s.*

Dijkstra's Algorithm

- All edge weights are assumed to be non-negative

- Maintain a set S of vertices whose final shortest-path weights from the source s have already been determined, or for all vertices $v \in S$, we have $d[v] = \delta(s, v)$.
- Select the vertex $u \in V - S$ with the minimum shortest-path estimate, using a priority queue Q
- Insert u into S
- Relax all edges leaving u

```
Dijkstra (G,w,s)
   initialize_single_source(G,s)
   S ← ∅
   Q ← V[G]
   while Q ≠ ∅ do
      u ← extract_min(Q)
      S ← S ∪ {u}
      for each vertex v ∈ Adj[u] do
         relax(u,v,w)
```

- Analysis
  - Each `extract_min` – $O(V)$
  - Total time for `extract_min` – $O(V^2)$
  - $|E|$ iterations of `for` loop with $O(1)$ for each iteration
  - Total run time – $O(V^2 + E) = O(V^2)$

Bellman-Ford Algorithm

-
```
bellman_ford (G,w,s)
   initialize_single_source (G,s)
   for i ← 1 to |V[G]| - 1 do
      for each edge (u,v) ∈ E[G] do
         relax (u, v, w)
   for each edge (u,v) ∈ E[G] do
      if d[v] > d[u] + w(u,v) then
         return false
   return true
```

Lemma 8 *Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w : E \to \Re$, and assume that G contains no negative weight cycles that are reachable from s. Then, at the termination of* BELLMAN-FORD, *we have $d[v] = \delta(s, v)$ for all vertices v that are reachable from s.*

Corollary 3 *Let $G = (V, E)$ be a weighted, directed graph with source vertex s and weight function $w : E \to \Re$. Then for each vertex $v \in V$, there is a path from s to v if and only if* BELLMAN-FORD *terminates with $d[v] < \infty$ when it is run on G.*