**Single-Source Shortest Paths**

Def. Given a weighted directed graph \((G(V,E),w)\), we define the **weight of a path**

\[ p = (v_0, v_1, \ldots, v_k) \]

as the sum of the weights of its constituent edges, i.e.,

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

We define the **shortest-path weight** \(\delta(u,v)\) of from \(u\) to \(v\) as

\[ \delta(u,v) = \min \{ w(p) : u \rightarrow \rightarrow \rightarrow \rightarrow v \} \]

if \(\exists\) a path from \(u\) to \(v\)

otherwise

A **shortest path** is one of shortest path weight.

**Optimal Substructure**

Lemma 24.1 (Page 582). Given a weighted directed graph \((G(V,E),w)\). Let \(p = (v_1, \ldots, v_k)\) be a shortest path from \(v_1\) to \(v_k\), then for every \(i, j\) s.t. \(1 \leq i \leq j \leq k\), the subpath \(p_{ij} = (v_i, \ldots, v_j)\) is a shortest path from \(v_i\) to \(v_j\).

**Single-Source Shortest Paths Problem**

- **Single-destination shortest-paths problem**: Find a shortest path to a given destination \(t\) for each vertex \(v\).
- **Single-pair shortest-path problem**: For given vertices \(u\) and \(v\), find a shortest path from \(u\) to \(v\).
- **All-pairs shortest-paths problem**: For all pairs of vertices \((u,v)\), find a shortest path from \(u\) to \(v\).

**Other variants of this problem**

- **Single-source shortest paths problem**: Given a weighted graph \((G=(V,E),w)\), find a shortest path from a given source vertex \(s \in V\) to each vertex \(v \in V\).

**A shortest path exists** iff there are no reachable negative cycles. Each node \(v\) contains a field \(d[v]\) which is an upper bound of \(\delta(s,v)\).

**Shortest Path Representation**

For each vertex \(v \in V\), we maintain a predecessor \(\pi[v]\) that is either a vertex or \(NIL\). The algorithms given in this chapter will set the predecessor \(\pi[v]\) so that \(\pi[v]\) points to the previous vertex on the shortest path from source \(s\) to \(v\). In this way, the shortest path is represented.

We thus have a predecessor subgraph

\[ G_{\pi} = (V_{\pi}, E_{\pi}) \]

where

\[ V_{\pi} = \{ v \in V : \pi[v] \neq NIL \} \cup \{ s \} \]

\[ E_{\pi} = \{ (\pi[v], v) \in E : v \in V_{\pi} - \{ s \} \} \]

The shortest path is not unique.
Shortest Path Tree

The algorithms we will consider will compute a shortest path tree
\[ G_s = (V_s, E_s) \]
where
\[ V_s = \{ v \in V : \pi[v] \neq \text{NIL} \} \cup \{ s \} \]
\[ E_s = \{ (\pi[v], v) \in E : v \in V_s \} \]

Relaxation

Relaxation Algorithm

1. Initialize-Single-Source(G, s)
   1. for each vertex \( v \in V \)
      2. \( d[v] \leftarrow \infty \)
      3. \( \pi[v] \leftarrow \text{NIL} \)
      4. \( d[s] \leftarrow 0 \)

   Relax(u, v, w)
   1. if \( d[v] > d[u] + w(u, v) \)
      2. then \( d[v] \leftarrow d[u] + w(u, v) \)
      3. \( \pi[v] \leftarrow u \)

Properties

Triangle inequality (Lemma 24.10)
For any edge \((u, v) \in E\), we have \( d(u, v) \leq d(u, v') + d(v, v') \).

Upper-bound property (Lemma 24.11)
We always have \( d(v) \leq \delta(u, v) \) for all vertices \( v \in V \), and once \( d[v] \) achieves the value \( \delta(u, v) \), it never changes.

No-path property (Corollary 24.12)
If there is no path from \( u \) to \( v \), then we always have \( d[v] = \infty \).

Convergence property (Lemma 24.14)
If \( u \rightarrow v \rightarrow w \) is a path in \( G \) for some \( u, v, w \in V \), and if \( d[u] = \delta(u, v) \) at any time prior to relaxing edge \((u, v)\), then \( d[w] = \delta(u, w) \) at all times afterward.

Path-relaxation property (Lemma 24.15)
If \( p = (x_0, x_1, \ldots, x_k) \) is a shortest path from \( x = x_0 \) to \( x_k \), and the edges of \( p \) are relaxed in the order \((x_0, x_1), (x_1, x_2), \ldots, (x_{k-1}, x_k)\), then \( d[x_k] = \delta(x, x_k) \).

Predecessor-subgraph property (Lemma 24.17)
Once \( d[v] = \delta(u, v) \) for all \( v \in V \), the predecessor subgraph is a shortest paths tree rooted at \( u \).

Two Single-Source Shortest Paths

- Bellman-Ford – A dynamic programming algorithm
- Dijkstra – A greedy algorithm which works only when weight function is non-negative

Single-Source Shortest Paths in DAGs

DAG-SHORTEST-PATHS(G, w, s)
1. topologically sort the vertices of \( G \)
2. INITIALIZE-SINGLE-SOURCE(G, s)
3. for each vertex \( u \), taken in topologically sorted order
   4. do for each vertex \( v \in \text{Adj}[u] \)
      5. do RELAX(u, v, w)
5. return TRUE
**Dijkstra’s Algorithm**

\[
\text{DIJKSTRA}(G, w, s) \\
1 \quad \text{INITIALIZE-SINGLE-SOURCE}(G, s) \\
2 \quad S \leftarrow \emptyset \\
3 \quad Q \leftarrow V[G] \\
4 \quad \text{while } Q \neq \emptyset \\
5 \quad \text{do } u \leftarrow \text{EXTRACT-MIN}(Q) \\
6 \quad S \leftarrow S \cup \{u\} \\
7 \quad \text{for each vertex } v \in \text{Adj}[u] \\
8 \quad \text{do RELAX}(u, v, w)
\]

**Linear System of Difference Constraints**

\[
\begin{pmatrix}
1 & -1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 \\
-1 & 0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 \\
0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & -1 & 1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{pmatrix} \\
\begin{pmatrix}
0 \\
-1 \\
1 \\
-1 \\
0 \\
0
\end{pmatrix}
\]

**Weighted Directed Graph**

Associated with a Linear System of Difference Constraints

- \(x_i \leftrightarrow v_i \in V\)
- \(x_j - x_i \leq b_k \leftrightarrow \text{edge } (v_i, v_j) \text{ of weight } b_k\)

We add an additional vertex \(v_0\), and the edges \((v_0, v_i) \forall x_i\)
Weighted Directed Graph Associated with a Linear System of Difference Constraints