Shortest Paths

Given a directed graph $G$

Edges have costs $\text{cost}(u \to v)$

Path length = sum of individual edge costs

Want to find shortest paths in $G$.

Shortest paths from a source $S$

Form a tree:

![Graph diagram]

[If shortest $S \to t$ path is
$s = u_1 \to u_2 \to \ldots \to u_{k-1} \to u_k = t$

then shortest $S \to u_{k-1}$ path

is also $s = u_1 \to u_2 \to \ldots \to u_{k-2} \to u_{k-1}$.]
Single-Source Shortest Paths (SSSP):
Find shortest path tree from S.

Point-to-Point:
Find shortest S\rightarrow T Path
Algorithm = run SSSP from S
Find t in the tree
[Nothing better known in general!]

All Pairs Shortest Paths (APSP):
Find all shortest paths.
Algorithm = run SSSP for all S.

So how to solve SSSP?

Let $c^*(u) = \text{true shortest path length to u}$

Triangle Inequality says:
For all $(u\rightarrow v)$ edges,
$c^*(v) \leq c^*(u) + \text{Cost}(u\rightarrow v)$.

[Can get to v by taking this edge.]

Generic algorithm:
Start with upper bound $c(\cdot)$ on $c^*(\cdot)$
Repeatedly pick edges somehow
and apply triangle inequality.
More formally:

Generic \((G, s)\):

- Set \(c(s) = 0\), \(c(u) = \infty\) \(\forall u \neq s\)
- Repeatedly pick edges \((u, v)\) somehow:
  \[c(v) \leq \min(c(u), \ c(u) + \text{cost}(u \to v))\]

"Relax" \((u, v)\)

or "Tighten" \((u, v)\)

Lemma: No matter how edges are picked, \(c(v) \geq c^*(v)\) \(\forall v\) at all times.

PE starts true.

If it's true at any point, updates have

\[c(v) \leq \min(c(u), c(u) + \text{cost}(u \to v))\]

\[\geq c^*(v),\]

\[\geq \min(c^*(v), c^*(u) + \text{cost}(u \to v))\]

\[\geq c^*(v)\]

by the triangle inequality.

Hence it remains true.

When does it get to the true answer?
Lemma

If \( S = u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_K \) is true shortest \( S \rightarrow u_K \) path, and tighten is called on \( (u_1, u_2), (u_2, u_3), \ldots, (u_{K-1}, u_K) \) in order — possibly with intervening calls before between and after — then \( c(u_K) = c^*(u_K) \).

Proof

We induct on \( K \).

\( K = 1 \Rightarrow u_1 = S \), so \( c(u_1) = 0 = c^*(u_1) \) to start, and it never increases.

Otherwise, by induction
\[ c(u_{K-1}) = c^*(u_{K-1}) \]
when tighten is called on \( (u_{K-1}, u_K) \).

Then this call sets
\[
\begin{align*}
c(u_K) &= \min (c(u_K), c(u_{K-1}) + \text{cost}(u_{K-1} \rightarrow u_K)) \\
&\leq c(u_{K-1}) + \text{cost}(u_{K-1} \rightarrow u_K) \\
&= c^*(u_{K-1}) + \text{cost}(u_{K-1} \rightarrow u_K) \\
&= c^*(u_K).
\end{align*}
\]
and later calls cannot increase it. \( \Box \)
So we need to tighten every edge of the path in order.

Bellman-Ford Algorithm:
N-1 times:
tighten every edge.

The i-th iteration tightens every edge
⇒ tightens the i-th edge on path
Paths have ≤ n-1 edges
(Assuming no negative cycles!)
⇒ tightened all edges in order after n-1 iterations
⇒ correct.

Running time O(mn)
Best known algorithm for general graphs!

Can do better if edge lengths nonnegative
by Dijkstra's algorithm.