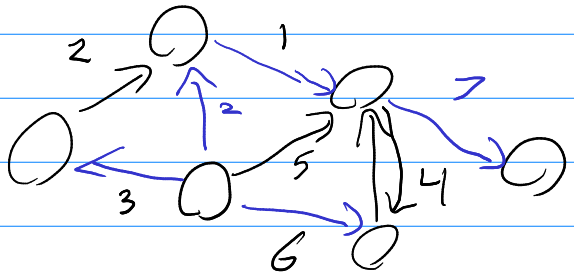
Shortest Paths

Given a directed graph G
edges have costs $cost(u \to v)$

Path length = sum of individual edge costs

Want to find shortest paths in G.

Shortest paths from a source s form a tree:



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

then shortest $s \to u_{k-1}$ path
is also $s = u_1 \to u_2 \to \cdots \to u_{k-2} \to u_{k-1}$.]

Single-Source Shortest Paths (SSSP):

Find shortest path tree from s.

Point-to-point:

Find shortest $s \to t$ 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 $c^*(u)$ = true shortest path length to u.

Triangle Inequality says:

For all $(u \to v)$ edges,

$$c^*(v) \leq c^*(u) + \text{cost}(u \to v).$$

[Can get to v by taking this edge.]

Generic algorithm:

Start with upper bound $c()$ on $c^*()$

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:

$c(v) \leftarrow \min(c(v), c(u) + \text{cost}(u \to v))$

"Relax" (u, v)

or "Tighten" (u, v)

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

Pf Starts true.

If it's true at any point, updates have

$$c(v) \leftarrow \min(\underbrace{c(v)}_{\geq c^*(v)}, \underbrace{c(u)}_{\geq c^*(u)} + \text{cost}(u \to v))$$

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

$$\geq c^*(v)$$

by the triangle inequality.

Hence it remains true. ∎

When does it get to the true answer?

Lemma

If $S = u_1 \to u_2 \to \cdots \to u_k$ is true shortest $s \to u_k$ path, and tighten is called on $(u_1, u_2), (u_2, u_3), \ldots (u_{k-1}, u_k)$ in order — possibly w/ intervening calls, before, between and after — then $c(u_k) = c^*(u_k)$.

Pf We induct on k.

$k = 1 \Rightarrow u_k = s$, so $c(u_k) = 0 = c^*(u_k)$ 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

$$\begin{aligned} c(u_k) &= \min\left(c(u_k),\ c(u_{k-1}) + \text{cost}(u_{k-1} \to u_k)\right) \\ &\le c(u_{k-1}) + \text{cost}(u_{k-1} \to u_k) \\ &= c^*(u_{k-1}) + \text{cost}(u_{k-1} \to u_k) \\ &= c^*(u_k). \end{aligned}$$

and later calls cannot increase it. ∎

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
$\Rightarrow$ tightens the i^{th} edge on path
Paths have $\leq n-1$ edges
(Assuming no negative cycles!)
$\Rightarrow$ tightened all edges in order after $n-1$ iterations
$\Rightarrow$ correct.

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

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