Shortest Paths

Adapted from the CLRS book slides

SHORTEST PATHS

How to find the shortest route between two points on a map.

Input:

- Directed graph $G = (V, E)$
- Weight function $w : E \to \mathbb{R}$

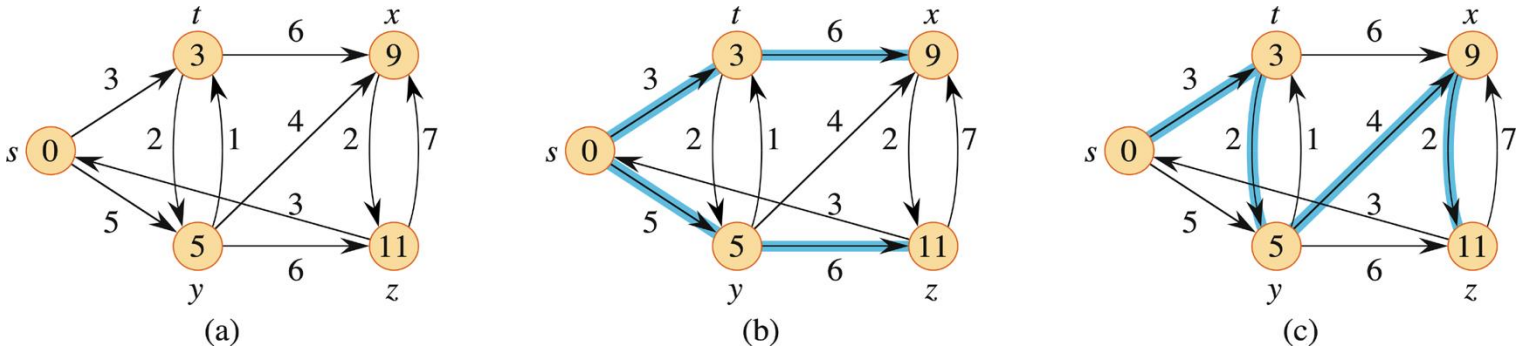
Shortest-path weight u to v:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there exists a path } u \rightsquigarrow v \text{ ,} \\ \infty & \text{otherwise .} \end{cases}$$

Shortest path u to v is any path p such that $w(p) = \delta(u, v)$.

EXAMPLE

shortest paths from s



This example shows that a shortest path might not be unique.

It also shows that when we look at shortest paths from one vertex to all other vertices, the shortest paths are organized as a tree.

NEGATIVE-WEIGHT EDGES

OK, as long as no negative-weight cycles are reachable from the source.

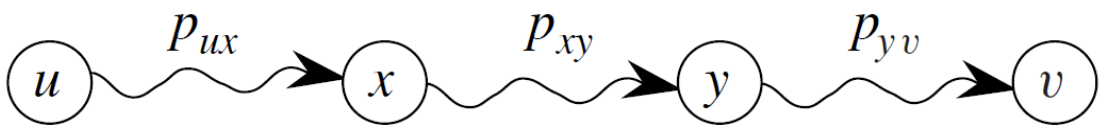
- If we have a negative-weight cycle, we can just keep going around it, and get $w(s, v) = -\infty$ for all v on the cycle.
- But OK if the negative-weight cycle is not reachable from the source.
- Some algorithms work only if there are no negative-weight edges in the graph. We'll be clear when they're allowed and not allowed.

OPTIMAL SUBSTRUCTURE

Lemma

Any subpath of a shortest path is a shortest path.

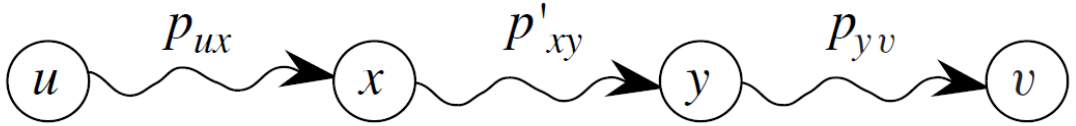
Proof Cut-and-paste.



Now suppose there exists a shorter path $x \overset{p'_{xy}}{\rightsquigarrow} y$.

Then $w(p'_{xy}) < w(p_{xy})$.

Construct p':



Contradicts the assumption that p is a shortest path.

CYCLES

Shortest paths can't contain cycles:

- Already ruled out negative-weight cycles.
- Positive-weight $\Rightarrow$ we can get a shorter path by omitting the cycle.
- 0-weight: no reason to use them $\Rightarrow$ assume that our solutions won't use them.

OUTPUT OF SINGLE-SOURCE SHORTEST-PATH ALGORITHM

For each vertex $v \in V$:

- $v.d = \delta(s, v)$.
  - Initially, $v.d = \infty$.
  - Reduces as algorithms progress. But always maintain $v.d \geq \delta(s, v)$.
  - Call $v.d$ a ***shortest-path estimate***.
- $v.\pi$ = predecessor of v on a shortest path from s.
  - If no predecessor, $v.\pi = \text{NIL}$.
  - π induces a tree—***shortest-path tree***.

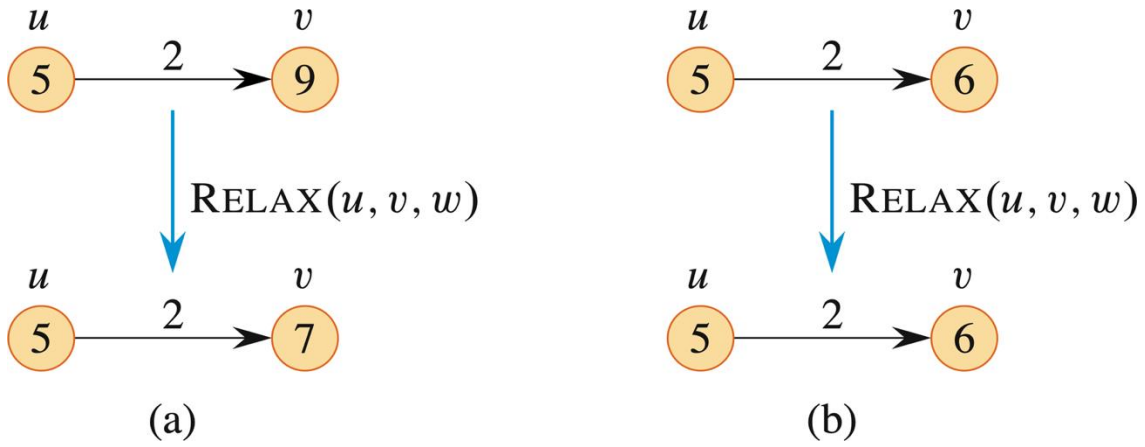
INITIALIZATION

All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE.

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

Can the shortest-path estimate for v be improved by going through u and taking (u, v)?

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2      v.d = u.d + w(u, v)
3      v.π = u
```



RELAXING AN EDGE (u,v)

For all the single-source shortest-paths algorithms we'll look at,

- start by calling INITIALIZE-SINGLE-SOURCE,
- then relax edges.

The algorithms differ in the order and how many times they relax each edge.

RELAXING AN EDGE (continued)

SHORTEST-PATHS PROPERTIES

Based on calling INITIALIZE-SINGLE-SOURCE once and then calling RELAX zero or more times.

Triangle inequality: For all $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.

Upper-bound property: Always have $v.d \geq \delta(s, v)$ for all v. Once $v.d$ gets down to $\delta(s, v)$, it never changes.

No-path property: If $\delta(s, v) = \infty$, then $v.d = \infty$ always.

Convergence property: If $s \rightsquigarrow u \rightarrow v$ is a shortest path, $u.d = \delta(s, u)$, and edge (u, v) is relaxed, then $v.d = \delta(s, v)$ afterward.

Path-relaxation property: Let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a shortest path from $s = v_0$ to v_k. If the edges of p are relaxed, *in the order*, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, even intermixed with other relaxations, then $v_k.d = \delta(s, v_k)$.