Single-Source Shortest Paths

謝仁偉 教授
jenwei@mail.ntust.edu.tw
國立台灣科技大學 資訊工程系
2017 Spring

Outline

- ***Shortest Paths***
- Shortest-Paths Properties
- The Bellman-Ford Algorithm
- Single-Source Shortest Paths in Directed Acyclic Graphs
- Dijkstra's Algorithm
- Difference Constraints and Shortest Paths

Overview (1/2)

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

- Input:
  - Directed graph $G = (V, E)$
  - Weight function $w : E \rightarrow \mathbb{R}$
- Weight of path $p = \langle v_0, v_1, \ldots, v_k \rangle$

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

  = sum of edge weights on path p.

Overview (2/2)

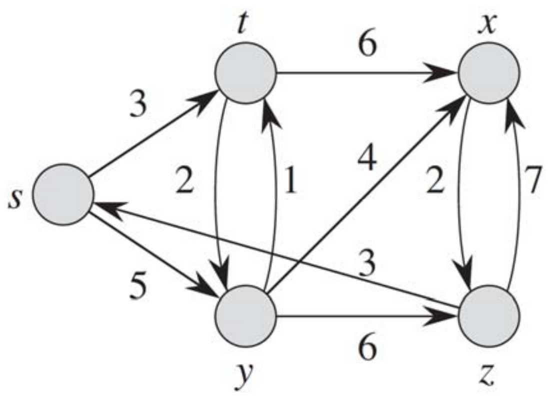
- Shortest-path weight u to v:

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

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

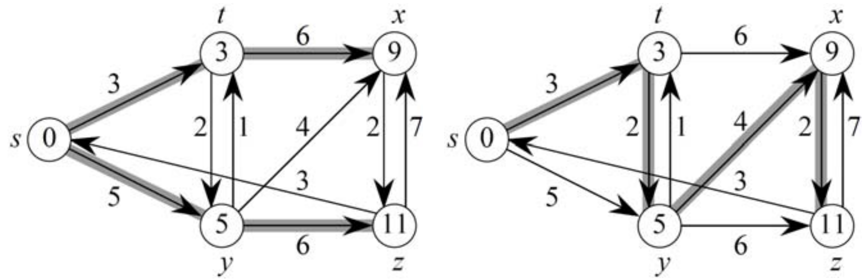
Example (1/2)

- Find shortest paths from s



Example (2/2)

- Shortest paths from s



- This example shows that the 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.

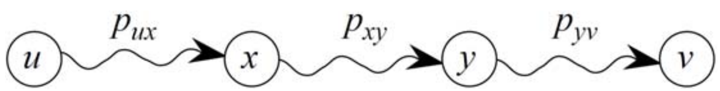
Variants

- Single-source: Find shortest paths from a given source vertex $s \in V$ to every vertex $v \in V$.
- Single-destination: Find shortest paths to a given destination vertex.
- Single-pair: Find shortest path from u to v. If we solve the single-source problem with source vertex u, we solve this problem also.
- All-pairs: Find shortest path from u to v for all $u, v \in V$. We'll see algorithms for all-pairs in the next chapter.

Optimal Substructure (1/2)

- **Lemma 24.1** Any subpath of a shortest path is a shortest path.

  Proof. Cut-and-paste.

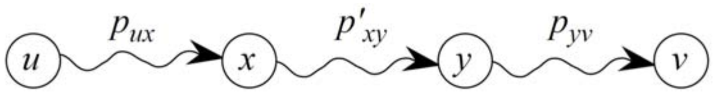


Suppose this path p is a shortest path from u to v. Then $\delta(u, v) = w(p) = w(p_{ux}) + w(p_{xy}) + w(p_{yv})$.

Now suppose there exists a shorter path $x \overset{p'_{xy}}{\leadsto} y$. Then $w(p'_{xy}) < w(p_{xy})$.

Optimal Substructure (2/2)

Construct p':



Then

$$\begin{aligned} w(p') &= w(p_{ux}) + w(p'_{xy}) + w(p_{yv}) \\ &< w(p_{ux}) + w(p_{xy}) + w(p_{yv}) \\ &= w(p)\,. \end{aligned}$$

Contradicts the assumption that p is a shortest path.

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.

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.
- Zero-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, $v.\pi$ = NIL.
  - π induces a tree – shortest-path tree.
  - We won't prove properties of π in lecture – see text.

Initialization

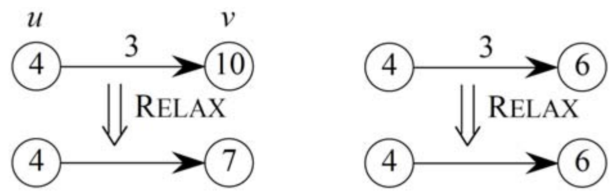
- All the shortest-paths algorithms start with INIT-SINGLE-SOURCE.

INITIALIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```


Relaxation (1/2)

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



RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```


Relaxation (2/2)

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

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

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

Outline

- Shortest Paths
- ***Shortest-Paths Properties***
- The Bellman-Ford Algorithm
- Single-Source Shortest Paths in Directed Acyclic Graphs
- Dijkstra's Algorithm
- Difference Constraints and Shortest Paths

Shortest-Paths Properties (1/2)

- **Lemma 24.10** (Triangle inequality)
  For any edge $(u, v) \in E$, we have $\delta(s, v) \le \delta(s, u) + w(u, v)$.
- **Lemma 24.11** (Upper-bound property)
  Always have $v.d \ge \delta(s, v)$ for all v. Once $v.d = \delta(s, v)$, it never changes.
- **Corollary 24.12** (No-path property)
  If there is no path from s to v, then we always have $v.d = \delta(s, v) = \infty$.

Shortest-Paths Properties (2/2)

- **Lemma 24.14** (Convergence property)
  If $s \rightsquigarrow u \rightarrow v$ is a shortest path in G for some $u, v \in V$, and if $u.d = \delta(s, u)$ at any time prior to relaxing edge (u, v), then $v.d = \delta(s, v)$ at all times afterward.
- **Lemma 24.15** (Path-relaxation property)
  If $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = \delta(s, v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.

Outline

- Shortest Paths
- Shortest-Paths Properties
- ***The Bellman-Ford Algorithm***
- Single-Source Shortest Paths in Directed Acyclic Graphs
- Dijkstra's Algorithm
- Difference Constraints and Shortest Paths

The Bellman-Ford Algorithm (1/3)