CS 314 - Shortest paths

Note Title 2/17/2010

Announcements

- Oral grading tomorrow

Shortest paths in a graph. (4.4)

Suppose we have $G=(V,E)$ and each edge $e \in E$ has a length l_e.

Here, we'll assume G is directed: $u \xrightarrow[l_{uv}]{10} v$.

Goal: Given two veritces, find shortest path between them.

We'll actually do something harder:

→ Given a source vertex s, compute <u>shortest path</u> from s to <u>every</u> other vertex.

Greedy idea:

Start with a set S.
(initially $S = \{s\}$)

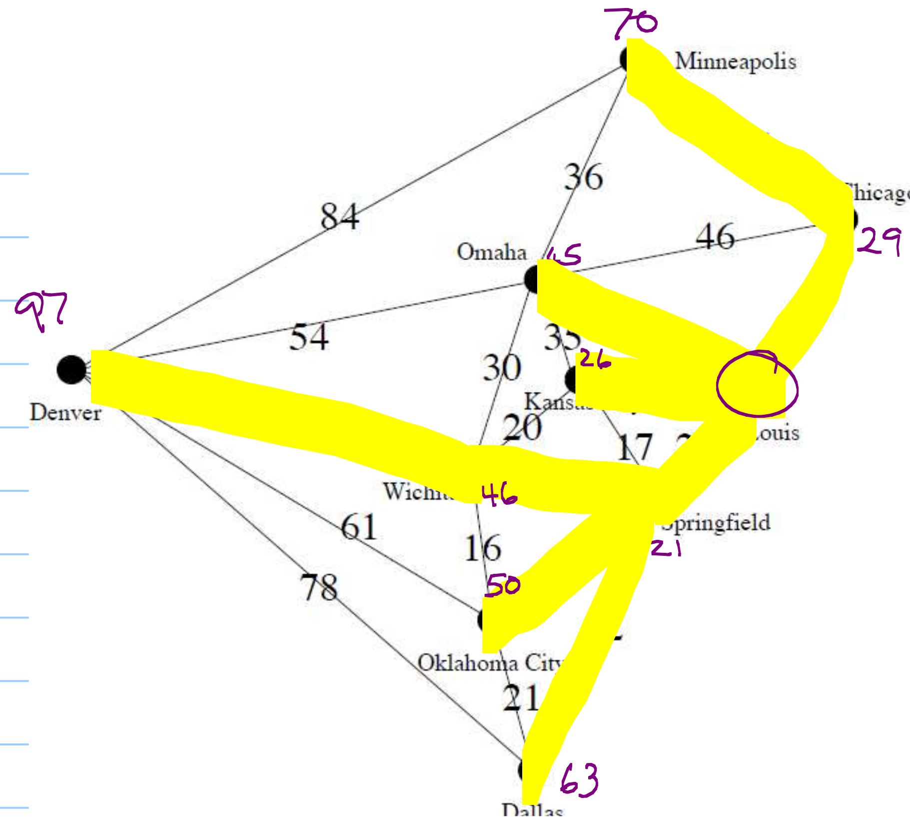
At each step, grow out from s, taking next shortest path from s to a new vertex & adding that to S.

Greedy idea:

Start with source
(here, St. Louis)

Let $S = \{s\}$

Consider edges
going out from S.



At each step, grow out from S,
taking next shortest path from s to
a new vertex & adding that to S.

Pseudo code: Dijkstra's algorithm

(actually Leyzorek, Gray, Johnson, Ladew, Meaker, Petry + Seitz)

SPtree(G, s):
    $S \leftarrow \{s\}$
    $D[s] \leftarrow 0$
    $T \leftarrow \phi$
    while $S \neq V$
        select node v with at least one edge into S where $d'(v) = \min_{(u,v) \in E, u \in S} D[u] + \ell_{uv}$ is minimized
        $S \leftarrow S \cup \{v\}$
        $D[v] \leftarrow d'(v)$
        $T \leftarrow T \cup \{(u,v)\}$

more later

Claim: At each stage, T is a set of shortest paths from s to S.

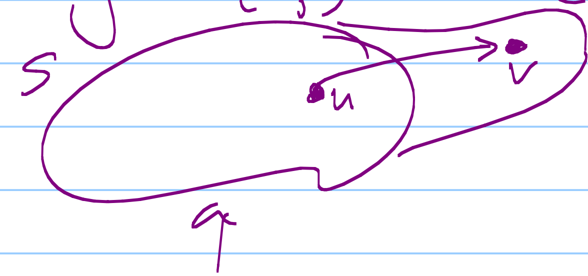
pf: induction on $|S|$

Base case: $|S| = 1$ so $S = \{s\}$, $T = \emptyset$ empty set

IH: Suppose true if $|S| < k$.

IS: Consider next edge (u,v) added.

T contains shortest path to u by IH.



(pf continued)

Suppose the path to v through u <u>isn't</u> the shortest path.

→ Then shortest path must use a different route, P.

We know P must leave S somewhere; let (x, y) be the first edge on P leaving S

Then distance from s to y is <u>less</u> than distance from s to v

But then algorithm would have added (x, y) instead of (u, v) ↯

∎

Improved Pseudocode

Dijkstra(G, S):

(this just calculates distances, not paths)

Create array D[v], initially all ∞

$S \leftarrow \{s\}$

$D[s] \leftarrow 0$

for every edge (s,u)

  set $D[u] \leftarrow l_{su}$

n times

While $S \neq V$

$O(n)$ select node $v \notin S$ with D[v] minimized

  $S \leftarrow S \cup \{v\}$

$d(v)$ for each edge (v,u)

    if $D[v] + l(u,v) < D[u]$

      $D[u] \leftarrow D[v] + l(u,v)$

heap

$\leftarrow O(n)$

$\leftarrow O(\log n)$

$$n(n-1) = m = O(n^2)$$

Runtime:

If D is just an array:

$$O\Big(n^2 + n\underbrace{\Big(\sum_{v \in V} d(v)\Big)}\Big)$$

$$= O(n^2 + n \cdot m) = O(n \cdot m)$$

If we use a heap to store distances, $O(\log n)$ time each time we extractmin or update priority.

$$O(n \log n + m \log n) = O(m \log n)$$