CS 314 - Shortest paths

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: $\overrightarrow{luv}$.

Goal: Given two vertices, find shortest path between them.
We'll actually do something harder:

- Given a source vertex $s$, compute shortest path from $s$ to every other vertex.

Greedy idea:

Start with a set $S$
(initially $S = \emptyset$)

At each step, grow out from $s$, taking next shortest path from $s$ to a new vertex and adding that to $S$. 
Greedy Idea:
Start with source (here, St. Louis)
Let $S = \emptyset$
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 ley 2002, Gray, Johnson, Ladew, Meaker, Petry + Sel'kov)

SPtree(G, s):
S ← \{s\}
D[s] ← 0
T ← Ø

while S ≠ V
    select node v with at least one edge into S
    where d'(v) = \min_{u \in S} \{ D[u] + lw \} is minimized
    S ← S ∪ \{v\}
    D[v] ← d'(v)
    T ← T ∪ \{(u,v)\}
Claim: At each stage, \( T \) is a set of shortest paths from \( s \) to \( S \).

pf: Induction on \(|S|\).

Base case: \(|S| = 1\) so \( S = \{s\} \), \( T = \emptyset \).

Ii: Suppose true if \(|S| < k\).

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

\( T \) contains shortest path to \( u \) by Iii.
(PF continued)

Suppose the path to $v$ through $u$ isn't 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 less than distance from $s$ to $v$.

But then algorithm would have added $(x, y)$ instead of $(u, v)$.
Improved Pseudo code

Dijkstra(G, s):

Create array D[v], initially all ∞
S ← {s}
D[s] ← 0
for every edge (s, u)
   set D[u] ← ∞

while S ≠ V
   select node v ∈ S with D[v] minimized
   S ← S ∪ {v}
   for each edge (v, u)
      if D[v] + ℓ(u,v) ≤ D[u]
         D[u] ← D[v] + ℓ(u,v)
Runtime:

If $D$ is just an array:
\[
O(n^2 + \sum_{\text{ev}} d(u))
\]
\[
= O(n^2 + n.m) = O(n.m)
\]

If we use a heap to store distances \(O(\log n)\) time each time we extract min or update priority:
\[
O(n \log n + m \log n) = O(m \log n)
\]