CSCI 3100

SSSP (cont)
Today

- No office hours today
- Monday: sign up for Friday HW slot
Next problem: Shortest paths

Goal: Find shortest path from $s$ to $v$.

We'll think directed, but really could do undirected w/ no negative edges:

Motivation:
- maps
- routing

Usually, to solve this, need to solve a more general problem:
Find shortest paths from $s$ to every other vertex.

Called the Single-Source Shortest Path Tree.

$\text{SSSP}$
Some notes:

- Why a tree?

If $s \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow v$ and $s \rightarrow a \rightarrow x \rightarrow y \rightarrow d \rightarrow u$ are shortest paths, then $s \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow u$ is also a shortest path.

- Negative edges?
  (Initially, assume none)

There is no shortest path from $s$ to $t$. 
Important to realize: \( \text{MST} \neq \text{SSSP} \)

Why? \( \text{SSSP} \) is rooted & directed
- \( \text{SSSP} \) for every vertex (these are different!)
Computing a SSSP:

(Ford 1956 & Dantzig 1957)

Each vertex will store 2 values.

(Think of these as tentative shortest paths)

- \( \text{dist}(v) \) is length of tentative shortest \( s \to v \) path (or \( \infty \) if don't have an option yet)

- \( \text{pred}(v) \) is the predecessor of \( v \) on that tentative path \( s \to v \) (or NULL if none)

Initially:

- \( \text{dist}(s) = 0 \)
- \( \text{pred}(s) = \text{NULL} \)
We say an edge $\overrightarrow{uv}$ is tense if $\text{dist}(u) + w(u\rightarrow v) < \text{dist}(v)$.

If $u\rightarrow v$ is tense:

use the better path!

So, relax:

\[
\text{RELAX}(u\rightarrow v):
\]
\[
\text{dist}(v) \leftarrow \text{dist}(u) + w(u\rightarrow v)
\]
\[
\text{pred}(v) \leftarrow u
\]
Algorithm: Dantzig

Repeatedly find tense edges and relax them.

When none remain, the \( \text{pred}(v) \) edges form the SSSP tree.

**InitSSSP(s):**

- \( \text{dist}(s) \leftarrow 0 \)
- \( \text{pred}(s) \leftarrow \text{Null} \)
- For all vertices \( v \neq s \)
  - \( \text{dist}(v) \leftarrow \infty \)
  - \( \text{pred}(v) \leftarrow \text{Null} \)

**GenericSSSP(s):**

- \( \text{InitSSSP}(s) \)
  - Put \( s \) in the bag
  - While the bag is not empty
    - Take \( u \) from the bag
    - For all edges \( u \rightarrow v \)
      - If \( u \rightarrow v \) is tense
        - \( \text{Relax}(u \rightarrow v) \)
        - Put \( v \) in the bag

To do: Which "bag"?
Dijkstra (59)

(actualy, Leyzorek et al '57, Dantzig '58)

Make the bag a priority queue.

Keep "explored" part of the graph, $S$.

Initially, $S = \{s\}$ + dist$(s)=0$

(all others NULL + $\infty$)

While $S \neq V$:

Select node $v \notin S$ with one edge from $S$ to $v$ with:

$$\min_{e=(u,v), u \in S} \text{dist}(u) + w(u \to v)$$

Add $v$ to $S$, set dist$(v)+\text{pred}(v)$
Four phases of Dijkstra’s algorithm run on a graph with no negative edges. At each phase, the shaded vertices are in the heap, and the bold vertex has just been scanned. The bold edges describe the evolving shortest path tree.
Correctness

**Thm.** Consider the set $S$ at any point in the algorithm. For each $u \in S$, the distance $\text{dist}(u)$ is the shortest path distance (so $\text{pred}(u)$ traces a shortest path).

**Pf:** Induction on $|S|$: \\

**Base case:** $|S| = 1$ \hspace{1cm} \checkmark \\
\[d(S) = 0.\]

**IH:** Claim holds when $|S| = k-1$.

**IS:** Consider when $|S| = k$.

Let $e = u \rightarrow v$ into $v$ getting us to $v$. 
**Claim:** any other $s \rightarrow x$ path where $x \notin S-v$ is longer than $s \rightarrow u \rightarrow v$

On any $s \rightarrow x$ path, some edge (left set $S$).
Portion of $s \rightarrow x$ path up to 1st edge leaving was considered when I added $u \rightarrow v$.

That portion of $s \rightarrow x$ wasn't chosen, so it was $> \text{dist}(u) + \text{w}(e)$

So no other paths to $v$ could be shorter $\Rightarrow S$ is best.
Back to implementation + run time:

For each \( v \in S \), could check each edge + compute \( \text{dist}(v) + w(e) \) runtime? \[ O(mn) \] \( \text{(for worse)} \)
Better: a heap! of vertices

When \( v \) is added to \( S \):
- Look at \( v \)'s edges and either insert \( w \) with key \( \text{dist}(v) + w(v \rightarrow w) \)
  or update \( w \)'s key if \( \text{dist}(v) + w(v \rightarrow w) \) beats current one \( O(\log n) \)

Runtime:
- At most \( m \) ChangeKey operators in heap
- At most \( n \) inserts/removes
  \( O(m \log n) \)
What about negative edges?

There is no shortest path from $s$ to $t$.

**Bellman-Ford (’58)**  
(Actually, Shiple ’55)

Key: use dynamic programming to force a path to use each edge at most once.

$$
\text{dist}_i(v) = \begin{cases} 
0 & \text{if } i = 0 \text{ and } v = s \\
\infty & \text{if } i = 0 \text{ and } v \neq s \\
\min \left\{ \text{dist}_{i-1}(v), \min_{u \rightarrow v \in E} (\text{dist}_{i-1}(u) + w(u \rightarrow v)) \right\} & \text{otherwise}
\end{cases}
$$
Notes cover 2 ways to formalize this:

**SHIMBELSSSP(s)**

**INITSSSP(s)**

repeat $V$ times:

for every edge $u \rightarrow v$

if $u \rightarrow v$ is tense

RELAX($u \rightarrow v$)

for every edge $u \rightarrow v$

if $u \rightarrow v$ is tense

return “Negative cycle!”

$\leftarrow n \leftarrow mO(mn)$

$\perp$
detects (but doesn’t work) negative cycles

$\Rightarrow$ defects, (but doesn’t work) negative cycles

$\Rightarrow$ work w/ negative cycles

**SHIMBELDP(s)**

$\text{dist}[0, s] \leftarrow 0$

for every vertex $v \neq s$

$\text{dist}[0, v] \leftarrow \infty$

for $i \leftarrow 1$ to $V - 1$

for every vertex $v$

$\text{dist}[i, v] \leftarrow \text{dist}[i - 1, v]$

for every edge $u \rightarrow v$

if $\text{dist}[i, v] > \text{dist}[i - 1, u] + w(u \rightarrow v)$

$\text{dist}[i, v] \leftarrow \text{dist}[i - 1, u] + w(u \rightarrow v)$

(more in notes...