6.046	Lecture 11	Mar. 17, 2015

TODAY: All-pairs shortest paths
- dynamic programming
- matrix multiplication
- Floyd-Warshall algorithm
- Johnson's algorithm
- difference constraints

Recall: single-source shortest paths [6.006]
- given directed graph $G=(V,E)$, vertex $s \in V$, & edge weights $w: E \to \mathbb{R}$
- find $\delta(s,v)$ = shortest-path weight $s \to v$ $\forall v \in V$
  (or $-\infty$ if neg-weight cycle along the way, or ∞ if no path)

situation	algorithm	time
unweighted ($w=1$)	BFS	$O(V+E)$
nonneg. edge weights	Dijkstra	$O(E+V \lg V)$
general	Bellman-Ford	$O(VE)$
acyclic graph (DAG)	topological sort + 1 pass Bellman-Ford	$O(V+E)$

using Fibonacci heaps

all of these results are the best known

All-pairs shortest paths:

given edge-weighted graph $G=(V,E,w)$,
find $\delta(u,v)$ for all $u,v \in V$

situation	algorithm	time (obvious)	$E=\Theta(V^2)$
unweighted	$\|V\| \times$ BFS	$O(VE)$	$O(V^3)$
nonneg. weights	$\|V\| \times$ Dijkstra	$O(VE+V^2 \lg V)$	$O(V^3)$
general	$\|V\| \times$ B-F	$O(V^2 E)$	$O(V^4)$
general	Johnson's (TODAY)	$O(VE+V^2 \lg V)$	$O(V^3)$

these results (except third) are also best known — don't know how to beat $|V| \times$ Dijkstra

Application: Google Maps preprocessing (between waypoints)
Internet routing

- define $w(u,v)=\infty$ for $(u,v) \notin E$

Dynamic program (#1):

1. subproblems: $d_{uv}^{(m)}$ = weight of shortest path $u \to v$ using $\le m$ edges
2. guessing: what's the last edge (x, v)?
3. recurrence: $d_{uv}^{(m)} = \min(d_{ux}^{(m-1)} + w(x,v) \text{ for } x \text{ in } V)$

   $$d_{uv}^{(0)} = \begin{cases} 0 & \text{if } u=v \\ \infty & \text{else} \end{cases}$$
4. topolog. order: for $m = 0, 1, \ldots, n-1$: for u & v in V:
   ↳ $|V|$
5. original problem:
   if no neg.-weight cycles then (by B-F analysis) shortest path is simple $\Rightarrow \delta(u,v) = d_{uv}^{(n-1)} = d_{uv}^{(n)} = \cdots$
   (neg.-weight cycle $\Leftrightarrow d_{vv}^{(n-1)} < 0$ for some $v \in V$)

Time: V^3 subproblems $\cdot$ V choices $\cdot$ $O(1)$ time/choice $= \boxed{O(V^4)}$ — no better than $|V| \times$ Bellman-Ford

Bottom-up via relaxation steps: (like Dijkstra & Bellman-Ford)

```
for m in range(1, n):
    for u in V:
        for v in V:
            for x in V:
                if duv > dux + dxv:
                    duv = dux + dxv
```

d_{xv} instead of $w(x,v)$ — only helps

relaxation step (Δ inequality)

omit superscripts because more relaxation never hurts

OR: $d_{uv}^{(m)} = \min(d_{ux}^{\lceil m/2 \rceil} + d_{xv}^{\lceil m/2 \rceil} \text{ for } x \in V) \Rightarrow O(n^3 \lg n)$ time! (student suggest.)

Matrix multiplication: (recall)

given $n \times n$ matrices A & B,
compute $C = A \cdot B$: $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$

- $O(n^3)$ via standard algorithm
- $O(n^{2.807})$ via Strassen's algorithm
- $O(n^{2.376})$ via Coppersmith-Winograd algorithm
- $O(n^{2.3728})$ via Vassilevska Williams algorithm

Connection to shortest paths:

- define $\oplus = \min$ & $\odot = +$
- then $C = A \odot B$ is $c_{ij} = \min_k (a_{ik} + b_{kj})$
- define $D^{(m)} = (d_{ij}^{(m)})$, $W = (w(i,j))$, $V = \{1, 2, \ldots, n\}$

$\Rightarrow D^{(m)} = D^{(m-1)} \odot W$ (by ③ above)

$= W^{ⓜ}$ where $W^{⓪} = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$

[$W^{ⓜ}$ makes sense because $\odot$ is associative, which follows from $(\mathbb{R}, \min, +)$ being closed semiring]

Matrix multiplication algorithm:

- $n-2$ multiplications $\Rightarrow O(n^4)$ time (still no better)
- repeated squaring: $((W^2)^2)^{2\cdots} = W^{2^{\lceil \lg n \rceil}} = W^{n-1}$
  $= (\delta(i,j))$ if no negative-weight cycles
  - time: $\boxed{O(n^3 \lg n)}$
- neg.-weight cycles $\Leftrightarrow$ neg. diagonal entries in W
- can't use Strassen etc. ☹ (no negation)

Transitive closure: $t_{ij} = \begin{cases} 1 & \text{if there's a path } i \to j \\ 0 & \text{else} \end{cases}$

$= [\text{is } \delta(i,j) < \infty ?]$ ⇒ special case of APSP

- $(\{0,1\}, \text{or}, \text{and})$ is a ring ⇒ can use Strassen etc.
⇒ $O(n^{2.3728} \lg n)$ time

Floyd-Warshall algorithm: faster dynamic program

① subproblem $c_{uv}^{(k)}$ = weight of shortest path $u \to v$ whose intermediate vertices $\in \{1, 2, \ldots, k\}$

$$i \to \leq k \to \leq k \to \cdots \to \leq k \to j \quad (V = \{1, 2, \ldots, n\})$$

② guessing = does shortest path use vertex k?

③ $c_{uv}^{(k)} = \min\{c_{uv}^{(k-1)}, c_{uk}^{(k-1)} + c_{kv}^{(k-1)}\}$

no neg.-weight cycles ⇒ use vertex k only once

$c_{uv}^{(0)} = w(u,v)$

④ for k: for u, v:

⑤ $\delta(u,v) = c_{uv}^{(n)}$, neg.-weight cycle ⇔ neg. $c_{uu}^{(n)}$

Time: $O(V^3)$ subproblems · 2 choices · $O(1)$
$= \boxed{O(V^3)}$

Bottom up via relaxation:

simple & efficient in practice

```
C = (w(u,v))
for k = 1, 2, ..., n:
    for u in V:
        for v in V:
            if c_uv > c_uk + c_kv:
                c_uv = c_uk + c_kv
```

relaxation again

again OK to omit subscripts

Johnson's algorithm:

1. find function $h: V \to \mathbb{R}$ such that
   $w_h(u,v) = w(u,v) + h(u) - h(v) \geq 0$ for all $u,v \in V$
   or determine that a negative-weight cycle exists
2. run Dijkstra's algorithm on (V, E, w_h)
   from every source vertex $s \in V$
   $\Rightarrow$ get $\delta_h(u,v)$ for all $u,v \in V$
3. claim $\delta(u,v) = \delta_h(u,v) - h(u) + h(v)$

Proof of claim:

- look at any $u \to v$ path p in G
- say p is $\underset{\underset{u}{\shortparallel}}{v_0} \to v_1 \to v_2 \to \cdots \to \underset{\underset{v}{\shortparallel}}{v_k}$

$$\begin{aligned}
\Rightarrow w_h(p) &= \sum_{i=1}^{k} w_h(v_{i-1}, v_i) \\
&= \sum_{i=1}^{k} \left[w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i) \right] \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \quad \text{telescoping} \\
&= w(p) + h(u) - h(v)
\end{aligned}$$

- so all $u \to v$ paths change in weight by the same offset $+h(u) - h(v)$

$\Rightarrow$ shortest path is preserved (but offset) $\square$

How to find h? (1)

$$w_h(u,v) = w(u,v) + h(u) - h(v) \geq 0$$

$$\Leftrightarrow h(v) - h(u) \leq w(u,v) \quad \text{for all } u,v \in V$$

↳ SYSTEM OF DIFFERENCE CONSTRAINTS

Theorem: if (V,E,w) has a negative-weight cycle then no solution to difference constraints

Proof: say $v_0 \to v_1 \to \cdots \to v_k \to v_0$ is neg. weight

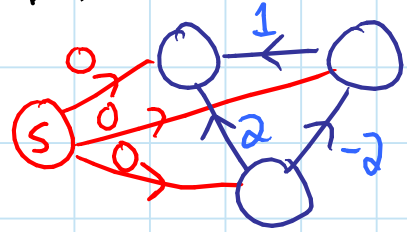
Good Will Hunting

$$\begin{aligned}
\text{if } \quad & h(v_1) - h(v_0) \leq w(v_0,v_1) \\
\& \quad & h(v_2) - h(v_1) \leq w(v_1,v_2) \\
& \quad \vdots \qquad \vdots \qquad \vdots \\
\& \quad & h(v_k) - h(v_{k-1}) \leq w(v_{k-1},v_k) \\
\& \quad & h(v_0) - h(v_k) \leq w(v_k,v_0) \\
\hline
\text{then sum:} \quad & 0 \leq w(\text{cycle}) < 0 \quad \text{※} \quad \square
\end{aligned}$$

Theorem: if (V,E,w) has no negative-weight cycle then can solve difference constraints

Proof: add to G a new vertex s & add weight-0 edges (s,v) for all $v \in V$ }*

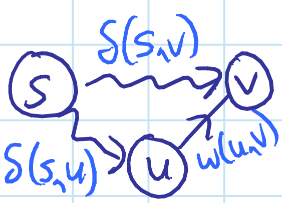
- introduce no (negative-weight) cycles



- $s \to v$ path now exists

$\Rightarrow \delta(s,v)$ is finite for all $v \in V$

- assign $h(v) = \delta(s,v)$
- $h(v) - h(u) \leq w(u,v) \Leftrightarrow \delta(s,v) - \delta(s,u) \leq w(u,v)$



$\Leftrightarrow \delta(s,v) \leq \delta(s,u) + w(u,v)$ TRIANGLE INEQUALITY $\square$

* Alternate reduction: for every $(u,v) \notin E$, add (u,v) with weight $M = |V| \cdot (\text{largest } |w|)$.
$\Rightarrow$ Strongly connected, still no neg.-weight cycles

Analysis:

① = Bellman-Ford from s	$O(VE)$
[+ reweight all edges	$O(E)$]
② = $\|V\| \times$ Dijkstra	$O(VE + V^2 \lg V)$
③ = reweight all pairs	$O(V^2)$
	$\boxed{O(VE + V^2 \lg V)}$

Also: Bellman-Ford can solve any system of difference constraints $\{x - y \le c\}$ (or report unsolvable) in $O(VE)$ where V=variables, E=constraints

Exercise: Bellman-Ford minimizes $\max_i x_i - \min_i x_i$

Applications to real-time programming
multimedia scheduling
temporal reasoning

e.g.

	bounds on:
$LB \le t_{end} - t_{start} \le UB$	duration
$0 \le t_{start2} - t_{end1} \le \varepsilon$	gap
$\|t_{start1} - t_{start2}\| \le \varepsilon$ or 0	synchrony