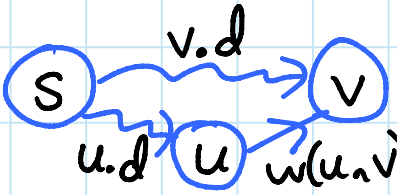
6.006	Lecture 15	Nov. 3, 2016

TODAY: Shortest Paths via
Dijkstra → sphere-growing intuition
→ algorithm, correctness, runtime
- bidirectional search
- A* & landmarks

Recall:

- $\delta(u,v) = \begin{cases} \inf\{w(p) \mid \text{path } p \text{ from } u \text{ to } v\} \\ \infty \text{ if no path from } u \text{ to } v \end{cases}$ ($\to -\infty$ if neg.-weight cycle on the way)
- SSSP: given edge-weighted directed graph $G=(V,E,w)$ & source $s \in V$, compute $\delta(s,v)$ for all $v \in V$, and shortest-path tree containing a shortest path from s to each $v \in V$
- Relaxation Algorithm:

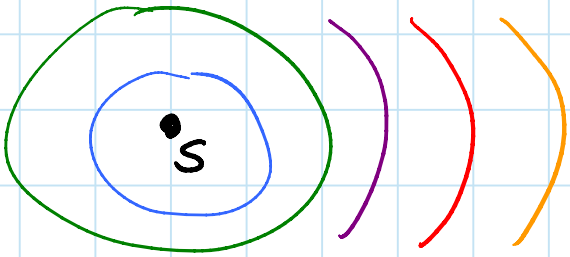
```
initialization(V):
    for v in V:
        v.d = ∞
        v.parent = None
    s.d = 0
while some edge (u,v) has v.d > u.d + w(u,v):
    pick some edge (u,v)
    relax(u,v):
        if v.d > u.d + w(u,v):
            v.d = u.d + w(u,v)
            v.parent = u
```



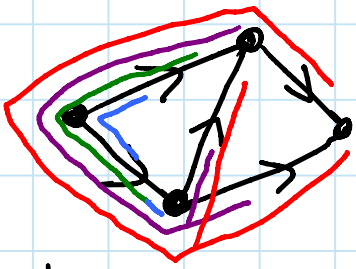
- Safety Lemma: $v.d \geq \delta(s,v)$ for all $v \in V$

SSSP algorithms so far:

setting	algorithm	time
DAG	topo. sort + 1 B-F	$O(V+E)$
general	Bellman-Ford	$O(VE)$
unweighted ($w=1$)	BFS	$O(V+E)$
TODAY: nonnegative weights	Dijkstra	$O(V \lg V + E)$



or really



Idea: grow sphere centered at source s
- this is exactly BFS when weights = 1
  i.e. takes unit "time" for sphere to "grow along" (traverse) an edge
- want to simulate this process where sphere takes $w(u,v)$ "time" to traverse (u,v)
- slow simulation for integer weights:
  subdivide $\xrightarrow{w}$ ⇨ 

  w (max w)

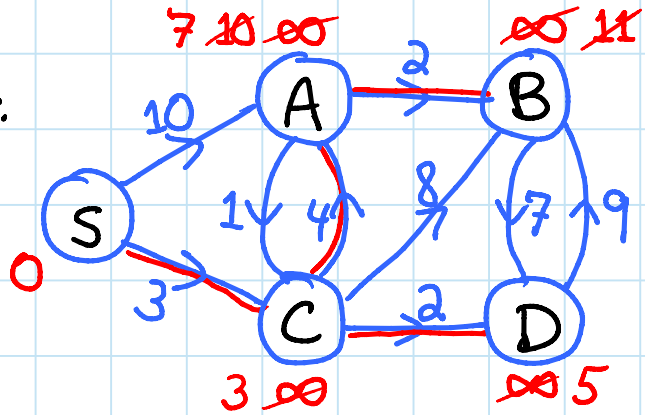
  $\Rightarrow O(V + E \cdot W)$ time
- fast simulation: discrete-event simulation using heaps to keep track of next event
  - event whenever sphere touches a vertex
  - just like F1 Kart Racing on PS2

Dijkstra's Algorithm: another relaxation algorithm

```
initialization(V)
Q = build heap on V using v.d as key
until Q empty:
    u = Q.extract-min()
    for v in Adj[u]:
        relax(u, v)  → may decrease v.d
        Q.decrease-key(v, v.d)
```

↳ each vertex v stores its location in Q so easy to find it & do decrease-key

Example:

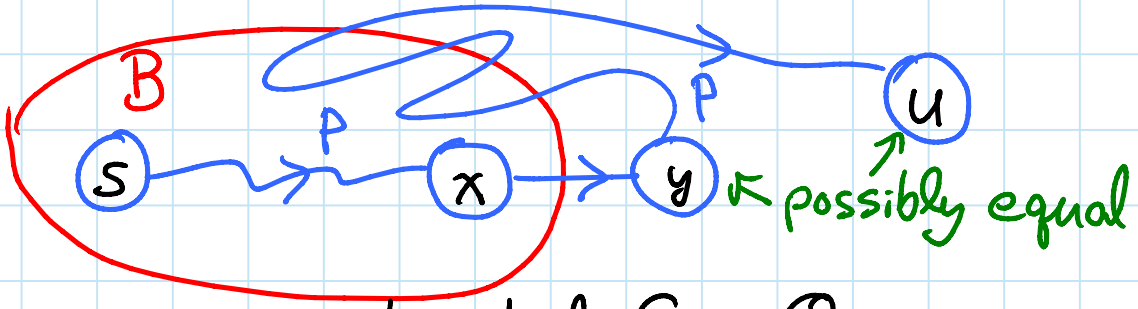


Q:	s	A	B	C	D
	0	∞	∞	∞	∞
		10	∞	3	∞
		7	11		5
		7	11		
			9		

Correctness Lemma: when u extracted from Q, $u.d = \delta(s,u)$

Proof: by induction on extraction order

- base case: $s.d = 0 = \delta(s,s)$ (no neg. weights)
- let $B = \{$vertices previously extracted from $Q\}$ (ball (inside of sphere))
  - $\Rightarrow$ by induction, $v.d = \delta(s,v)$ for all $v \in B$
  - ... when extracted, and still, because relaxation only decreases $v.d$ & $\geq \delta(s,v)$ by safety
- consider shortest path p from s to u
  - $\Rightarrow w(p) = \delta(s,u)$
- look at first edge (x,y) where p exits
  - exists because $s \in B$ & $u \notin B$



- when x was extracted from Q,
  $x.d = \delta(s,x)$ and we relaxed (x,y)
  - $\Rightarrow y.d \leq w($prefix of p from s to $y)$
    $= \delta(s,y)$ ← subpaths of shortest paths are shortest paths
    $\leq \delta(s,u)$ ← nonnegative edge weights
    $\leq u.d$ ← safety lemma
- but u was chosen to have min. d value
  - $\Rightarrow u.d = y.d = \delta(s,u) = \delta(s,y)$. □

Corollary: at end of Dijkstra's algorithm, $u.d = \delta(s,u)$ for all $u \in V$

- Correctness Lemma $\Rightarrow$ true at some time
- relaxation only decreases $u.d$
- Safety Lemma $\Rightarrow$ can't $\square$

Running time:

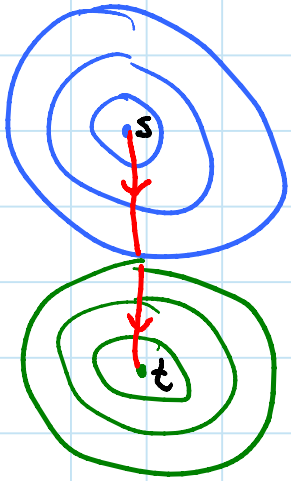
- $|V|$ extract-mins (never insert into Q!)
- $|E|$ decrease-keys (each edge relaxed exactly once)
- $|Q| \leq |V|$

data structure	extr.-min	decr. key	Dijkstra
binary heap [L4]	$O(\lg V)$	$O(\lg V)$	$O((V+E)\lg V)$
	great for $E = O(V)$ ←		$= O(E \lg V)$ if connected
unsorted array	$O(V)$	$O(1)$	$O(V^2+E) = O(V^2)$
			great for $E = \Theta(V^2)$ ↵
Fibonacci heaps [CLRS, ch. 19]	$O(\lg V)$	$O(1)$ amortized	$O(V \lg V + E)$
			↳ great always

Bidirectional search:

for computing just $s \to t$ shortest path

- interleave steps of
  - Dijkstra(G, s)
  - Dijkstra(reverse(G), t)
- stop when they meet



"Analysis": not worst case

- suppose $\delta(s,t) = d$
  & $f(r)$ = # vertices within distance r of s or t
  $\Rightarrow$ Dijkstra visits $f(d)$ vertices
  & bidirectional search visits $2f(d/2)$ vertices
- geometric data: $f(r) \approx r^c$ ($r=2$ or 3 for 2D/3D)
  $\Rightarrow 2f(d/2) = 2(d/2)^c = d/2^{c-1}$ (2× or 4×)
  $\Rightarrow$ constant-factor savings
- AI/robot planning, games/puzzles, etc.:
  $f(r) \approx c^r \Rightarrow 2f(d/2) = 2c^{d/2} = 2\sqrt{c^d}$!

A* search:

- choose heuristic $h: V \to \mathbb{R}$ "goodness of node"
- reweight: $w_h(u,v) = w(u,v) + h(u) - h(v)$

Claim: $\delta(u,v) = \delta(u,v) + h(u) - h(v)$

Proof: look at any $u \to v$ path $p: v_0 \to v_1 \to \cdots \to v_k$ ($v_0 = u$, $v_k = v$)

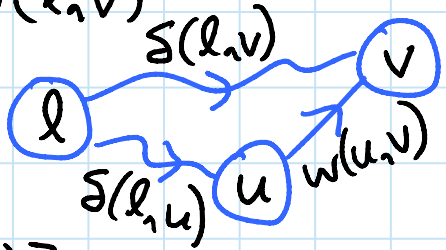
$$\begin{aligned} \Rightarrow w_h(p) &= [w(v_0,v_1) + h(v_0) - h(v_1)] \\ &+ [w(v_1,v_2) + h(v_1) - h(v_2)] \quad \text{telescoping} \\ &+ \cdots \\ &+ [w(v_{k-1},v_k) + h(v_{k-1}) - h(v_k)] \\ &= w(p) + h(v_0) - h(v_k) \\ &= w(p) + h(u) - h(v) \end{aligned}$$

$\Rightarrow$ same offset for all paths $u \to v$

$\Rightarrow$ shortest paths preserved $\square$

Landmarks: precompute SSSP from l & use $h(v) = \delta(l,v) - \delta(l,t)$

$$\begin{aligned} \Rightarrow w_h(u,v) &= w(u,v) + h(u) - h(v) \\ &= w(u,v) + \delta(l,u) - \delta(l,t) - [\delta(l,v) - \delta(l,t)] \\ &= w(u,v) + \delta(l,u) - \delta(l,v) \\ &\geq 0 \end{aligned}$$



$\Rightarrow$ can still use Dijkstra

- bias "toward" t
- $h(v) = \max_{l \in L} [\delta(l,v) - \delta(l,t)]$ also works

L: small set of landmarks