Shortest Paths: Intro

Homework preview
Weighted graphs
General Approach
Negative edges
Optimal substructure

Reading: Ch 24 (Intro)

MOTIVATION

Shortest way to drive from A to B
Google maps "get directions"

Formulation: Problem on a weighted graph $G(V,E)$
$w: E \to \mathbb{R}$

Two algorithms: Dijkstra $O(V \lg V + E)$
assumes non-negative edge weights

Bellman Ford $O(VE)$
general algorithm

Problem Set 5

- Use Dijkstra to find shortest path from CalTech to MIT
  - See "CalTech Cannon Hack" photos @ web.mit.edu
  - See Google maps from CalTech to MIT

- Model as a weighted graph $G(V, E)$, $w: E \to \mathbb{R}$

  $V =$ vertices (street intersections)

  $E =$ edges (streets, roads); directed edges (one way roads)

  $w(u,v) =$ weight of edge from u to v (distance, toll)

---

path $p = \langle v_0, v_1, \ldots v_k \rangle$

$(v_i, v_{i+1}) \in E$ for $0 \leq i < k$

$$w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$$

Notation: $v_0 \overset{p}{\rightsquigarrow} v_k$ means p is a path from v_0 to v_k

(v_0) is a path from v_0 to v_0 of weight 0

Define: Shortest path weight from u to v as

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if } \exists \text{ any such path} \\ \infty & \text{otherwise } (v \text{ unreachable from } u) \end{cases}$$

Single Source Shortest Paths

Given $G=(V,E)$, w and a source vertex s,
find $\delta(s,v)$ [and the best path] from
s to each $v \in V$

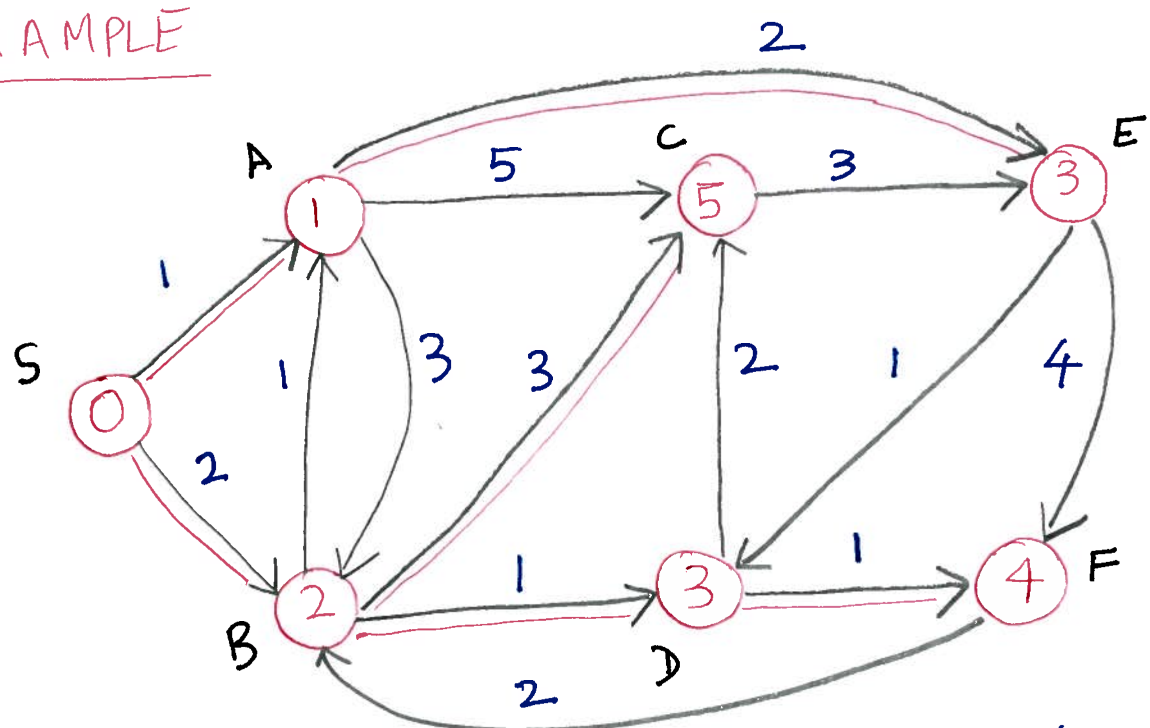
Data structures: $d[v]$ = value inside circle

$= \begin{cases} 0 & \text{if } v=s \\ \infty & \text{otherwise} \end{cases}$ ⟸ initially

$= \delta(s,v)$ ⟸ at end

$d[v] \geq \delta(s,v)$ at all times,
$d[v]$ decreases as we find better paths to v
$\pi[v]$ = predecessor on best path to v, $\pi[s]$ = NIL

EXAMPLE



— edges give predecessor π relationships.

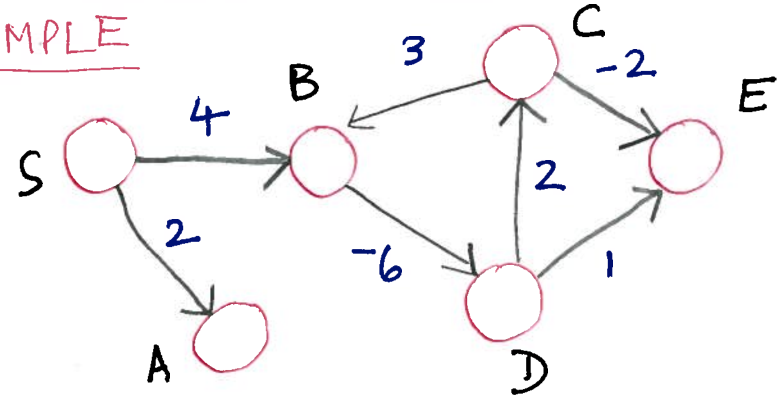
NEGATIVE-WEIGHT EDGES

- Natural in some applications (e.g. logarithms used for weights)
- Some algorithms disallow negative weight edges (e.g., Dijkstra)
- If you have negative weight edges, you might also have negative weight cycles

$\Downarrow$

may make certain shortest paths undefined!

EXAMPLE



$B \to D \to C$ has weight $-6 + 2 + 3 = -1 < 0$!

Shortest path $S \rightsquigarrow C$ (or B, D, E) is undefined
can go around $B \to D \to C$ as many times as you like

Shortest path $S \rightsquigarrow A$ is defined and has weight 2

If negative wt edges are present, s.p. algorithm should find neg wt cycles (e.g., Bellman Ford)

GENERAL STRUCTURE OF S.P. ALGS (no neg cycles)

Initialize: for $v \in V$: $d[v] \leftarrow \infty$

$\pi[v] \leftarrow \text{NIL}$

$d[s] \leftarrow 0$

Main: repeat:
  Select edge (u,v) [somehow]

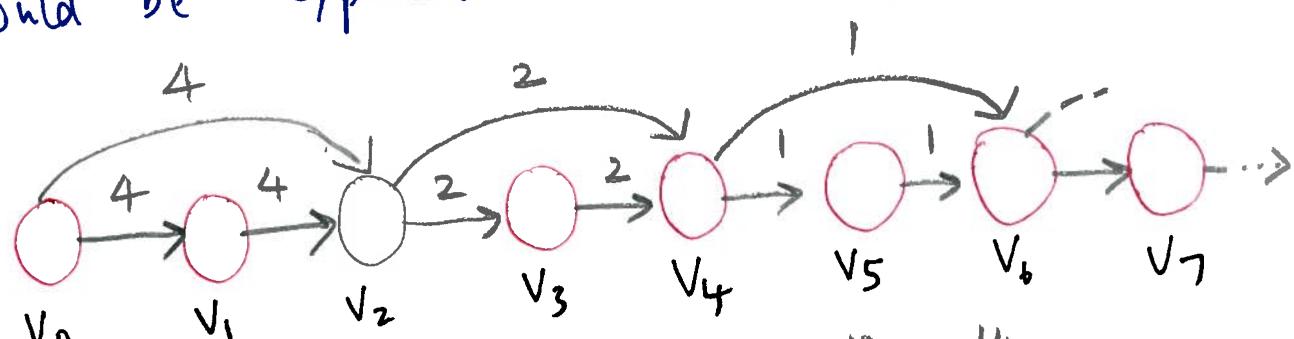
"Relax" edge (u,v):
  if $d[v] > d[u] + w(u,v)$:
    $d[v] \leftarrow d[u] + w(u,v)$
    $\pi[v] \leftarrow u$

until all edges have $d[v] \leqslant d[u] + w(u,v)$

COMPLEXITY

Termination? (needs to be shown even w/o negative cycles)

Could be exponential time with poor choice of edges.



4	8	10	12	13	14, 13
			10	11	12, 11
	4	6	8	9	10, 9
			6	7	8, 7

$T(0) = 0$

$T(n+2) = 3 + 2T(n)$ v_0, v_1; v_1, v_2; $v_2 .. v_n$; v_0, v_2; $v_2 .. v_n$

$T(n) = \theta(2^{n/2})$

OPTIMAL SUBSTRUCTURE

Theorem: Subpaths of shortest paths are shortest paths

Let $p = \langle v_0, v_1, \ldots v_k \rangle$ be a shortest path

Let $p_{ij} = \langle v_i, v_{i+1}, \ldots v_j \rangle$ $\quad 0 \le i \le j \le k$

Then p_{ij} is a shortest path

Proof:

$$p = \begin{tikzcd} v_0 \arrow[r, "p_{0j}"] & v_i \arrow[r, "p_{ij}"] \arrow[r, bend right, dashed, "p_{ij}'"'] & v_j \arrow[r, "p_{jk}"] & v_k \end{tikzcd}$$

If p_{ij}' is shorter than p_{ij}, cut out p_{ij} and replace with p_{ij}'; result is shorter than p. Contradiction. $\boxtimes$

TRIANGLE INEQUALITY

Theorem: For all $u, v, x \in X$, we have

$$\delta(u,v) \le \delta(u,x) + \delta(x,v)$$

Proof: