Lecture #7

CS 170

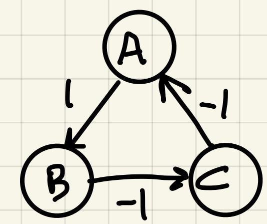
Spring 2021

Shortest Paths in Graphs

last time: 1) All edges have same weight $\Rightarrow$ BFS

today: 2) Edges can have different positive weights
$\Rightarrow$ Dijkstra's Algorithm

3) Edges can have negative weights
$\Rightarrow$ Bellman-Ford Algorithm



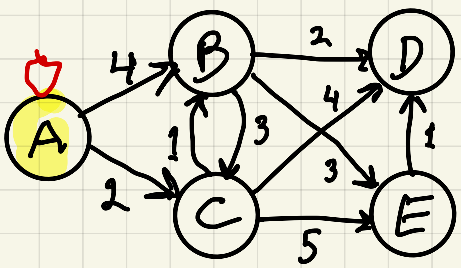
4) Detecting negative cycles

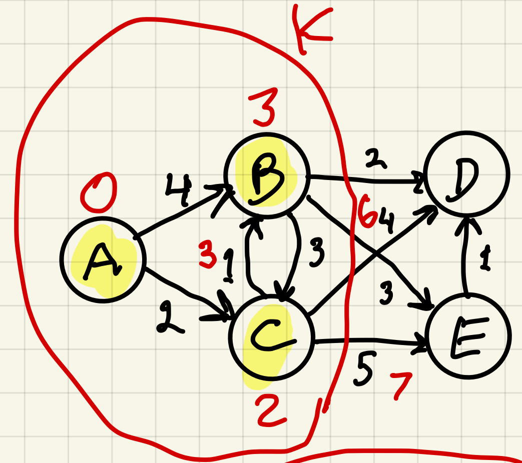
5) Shortest Paths in DAGs

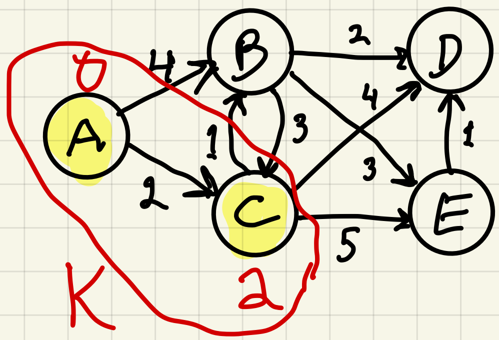
Notation: $G=(V,E)$, $\ell : E \to \mathbb{N}$ gives length of each edge

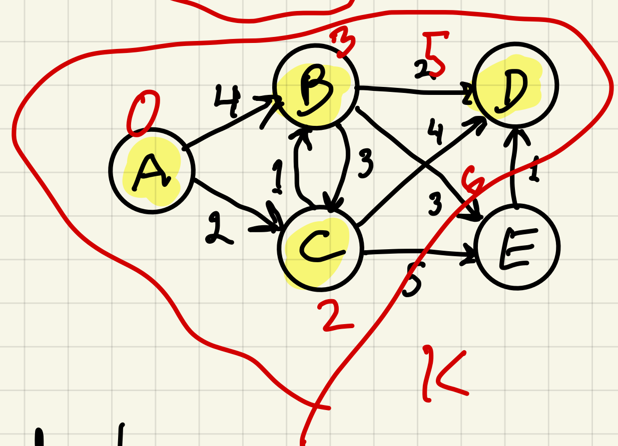
$d(s,v)$ = length of shortest path from s t

Example









Idea: at each step: update
K = vertices to which we know shortest path

There are no vertices outside K with shorter paths to them than those inside K

Dijkstra's Algorithm

Dijkstra(G, s) ... $G = (V, E)$

$dist[s] = 0$,

$\forall v \neq s$, $dist[v] = \infty$

$K = \emptyset$ ... vertices for which shortest paths known

while $K \neq V$

pick $u \in V / K$ with smallest $dist[u]$

$K = K \cup \{u\}$

for all $(u, v) \in E$

update(u,v) $\Big($ $dist[v] = \min(dist[v], dist[u] + \ell(u,v))$

Proof of Correctness for Dijkstra

Notation: $d(s,v)$ = length of a shortest path from s to v

Claim: At any time $\forall v \in K$, $dist[v] = d(s,v)$

Proof: Induction

Base Case: $K = \emptyset$ trivial

First Step: $K = \{s\}$ $\quad d(s,s) = 0 = dist[s]$

Induction Step: let v be vertex with smallest $dist[v]$: claim $dist[v] = d(s,v)$

Let $\underbrace{s \to \to \to \cdots a}_{\text{all in } K} \to \underset{\uparrow}{b} \to \to v \notin K$ be a shortest path

b first one not in K : Fact every prefix of a shortest path is a shortest path

① if $b = v \Rightarrow dist[v] = dist[b]$ since $b = v$

② if $b \neq v \Rightarrow$ $dist[b] < dist[v]$ contradicts alg choosing v

$\leq dist[a] + \ell(a,b)$ inner loop of alg

$= d(s,a) + \ell(a,b)$ since $a \in K$, by induction

$= d(s,b)$ since $s \rightsquigarrow b$ is a (prefix of) a shortest path

$= d(s,v)$

Dijkstra's Algorithm, Updated

Dijkstra(G, s) ... $G = (V, E)$

$dist[s] = 0$

$\forall v \neq s$, $dist[v] = \infty$

$\cancel{K = \emptyset}$ $U = V$ $(U = V / K)$ initialize PriorityQueue $Q \leftarrow V$, keys = dist

while $\cancel{K \neq V}$ $U \neq \emptyset$ Q not empty

pick $u \notin \cancel{V \setminus K}$ $u \in U$ with smallest $dist[u]$

$\cancel{K = K \cup \{u\}}$ remove u from U $u = \text{DeleteMin}(Q)$

for all $(u, v) \in E$

DecreaseKey$(Q, v, dist[v])$ $dist[v] = \min(dist[v], dist[u] + \ell(u, v))$

$\Rightarrow$ need data structure for picking smallest $dist(u)$, updating

Priority Queue: Binary Heap $\left\{ \begin{array}{l} \text{DeleteMin} \\ \text{DecreaseKey} \\ \text{Insert} \end{array} \right.$ $O(\log |V|)$

... Fibonacci Heap

Running time for Dijkstra

Count # operation

- Make Queue once: $|V|$ inserts, $\Rightarrow$ cost $= O(|V| \log |V|)$
  or $O(|V|)$
- DeleteMin: once per vertex: $|V|$
- DecreaseKey: once per edge: $|E|$

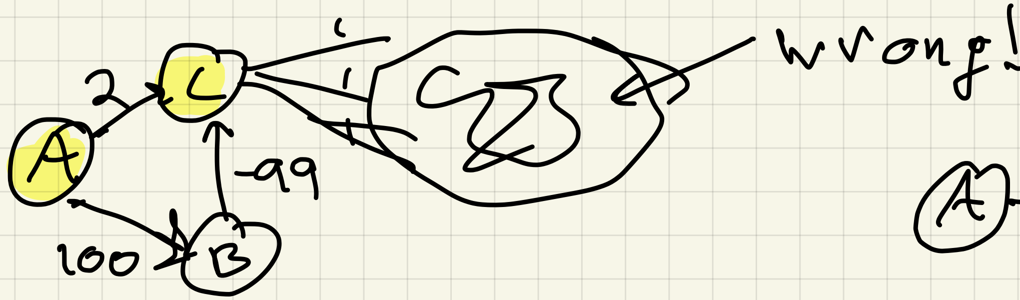
overall time: $O((|V|+|E|) \log |V|)$ using binary heap

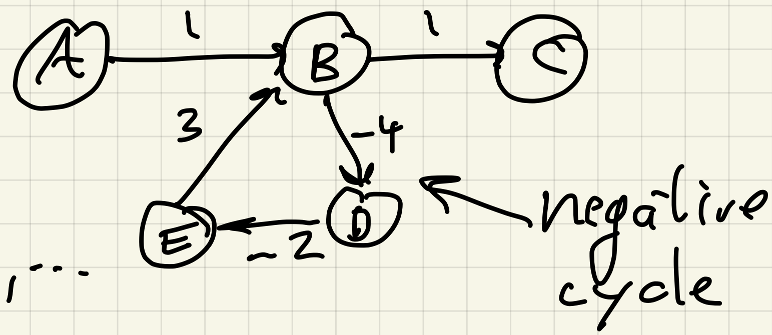
$O(|V| \log |V| + |E|)$ using Fibonacci

useful if $|E| \gg |V|$ "dense graph"

More complicated algs nearly $O(|V|+|E|)$

Shortest Paths with Positive or Negative Edge Lengths





dist(A,C) = 2, −1, −4, ...

if $(u,v) \in E$ update(u,v): $dist[v] = \min(dist[v], dist[u] + \ell(u,v))$

① update "safe" $dist[v] \geq d(s,v)$

$\Longrightarrow$ extra updates OK

② if shortest path from s to v is

$s \xrightarrow[1]{} \xrightarrow[2]{} \xrightarrow[3]{} u \xrightarrow[\cdots]{} v$ and $dist[u] = d(s,u)$

then after update $dist[v] = d(s,v)$

Bellman-Ford

Shortest path $s \to u_1 \to u_2 \to u_3 \to \cdots \to u_t \to v$ $\quad t \leq |V|-1$

$i=1$ update$(s, u_1) \Rightarrow dist[u_1] = d(s, u_1)$ → update

$i=2$ update $(u_1, u_2) \Rightarrow dist[u_2] = d(s, u_2)$ → update

$i=3$ update$(u_2, u_3) \Rightarrow dist[u_3] = d(s, u_3)$ → $\vdots$

$\vdots$

$i \leq |V|-1$ update$(u_t, v) \Rightarrow dist[v] = d(s, v)$ → update $\to dist[v] = d(s,v)$

Bellman-Ford

for $i = 1$ to $|V|-1$

forall $(u,v) \in E$, update(u,v)

no negative cycles $\Rightarrow$ all shortest path have $\leq |V|-1$ vertices

$\Rightarrow$ all updates appear in desired order $\Rightarrow$ all $dist[v] = d(s,v)$

Running time $= (|V|-1) \cdot |E| \cdot$ time(update) $= O(|V| \cdot |E|) = O(|V|^3)$

Shortest Paths in DAGs

DAG $\Rightarrow$ no cycles $\Rightarrow$ no negative cycles
$\Rightarrow$ pos + neg edge lengths OK

Idea: Topologically sort, starting with s

all edges go left to right (DFS)

$\Rightarrow$ all shortest paths look like
$(s, v_{i_1})(v_{i_1}, v_{i_2})(v_{i_2}, v_{i_3}) \cdots \quad i_1 < i_2 < i_3 \cdots$

$\Rightarrow$ updating all (v_i, v_k) in order of increasing i works

Cost = topological sort = DFS = $O(|E|)$
+ updating in order $\Rightarrow O(|V|+|E|)$

Detecting Negative Cycles

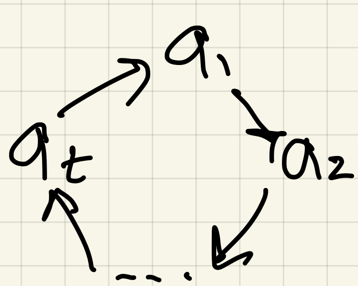
Bellman Ford assumed no neg cycles $\Rightarrow$
all shortest path have $\leq |V|-1$ vertices

Thm: No neg cycles $\Leftrightarrow$ Running Bellman Ford for one more iteration (update all edges once more) doesn't change any dist[v]

$\Rightarrow$ run Bellman $|V|$ times instead of $|V|-1$, signal neg cycle if any dist[v] changes

Proof: No neg cycles $\Rightarrow$ all shortest paths have $|V|-1$ vertices
$\Rightarrow$ updating once more "safe", nothing changes

Run BF, no dist changes



$\text{dist}[a_1] \leq \text{dist}[a_t] + \ell(a_t, a_1)$

$\text{dist}[a_2] \leq \text{dist}[a_1] + \ell(a, a_2)$

...

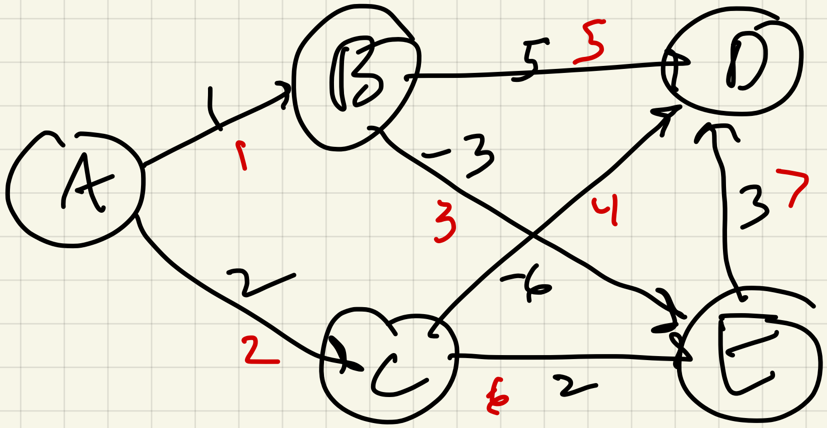
$\text{dist}[a_t] \leq \text{dist}[a_{t-1}] + \ell(a_{t-1}, a_t)$

otherwise dist[a] would chang.

$\sum_{j=1}^{t} \text{dist}[a_j] \leq \sum_{j=1}^{t} \text{dist}[a_j] + \sum$ all edges in cycle

no neg cycle! $\longrightarrow$ $0 \leq \sum$ all edges in cycle

lengths of



	i=0	i=1	i=2
A	0	0	0
B	∞	1	1
C	∞	2	2
D	∞	-4	-4
E	∞	-2	-2

victory!