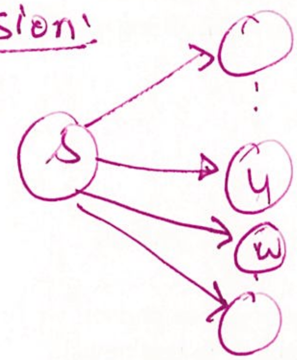
Nov 19

Attempt 4: $OPT(s, E')$ → cost of a shortest s-t path only using edges in E'.

$E' \subseteq E$

(•) **Recursion:**



If a shortest s-t path uses (s,u)

$$OPT(s,E') = C_{s,u} + OPT(u, E' \setminus \{(s,u)\})$$

In general:

$$OPT(s, E') = \min_{w:\ (s,w) \in E} \{ C_{s,w} + OPT(w, E' \setminus \{(s,w)\}) \}.$$

(•) **Ordering among subproblem:** Order according to size of $|E'|$ (order among s doesn't matter)

(•) # sub-problems $= n \cdot 2^m$ X not poly

Q: How many subsets of $\{1,..,m\}$ are there?

Ex: 2^m.

Using E' is overkill since we do NOT really need to keep track of which edges we have not used yet.

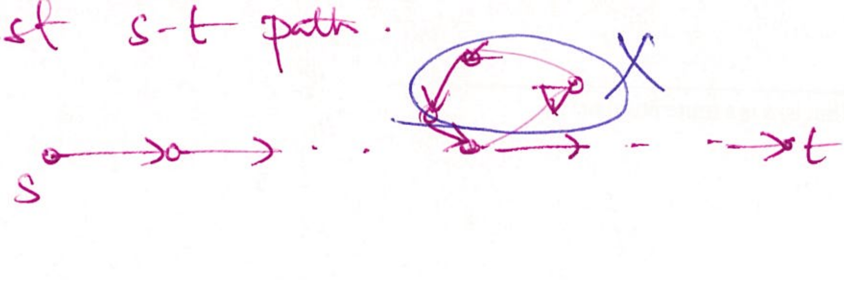
Want: keep track of how "close" we are to t.

Attempt 5: Bellman-Ford.

$OPT(s, i)$ = cost of a shortest s-t path using $\le i$ edges.

Prop: If G has no negative cycle $\Rightarrow \forall s, \exists$ a simple shortest s-t path.

Pf (idea): If not

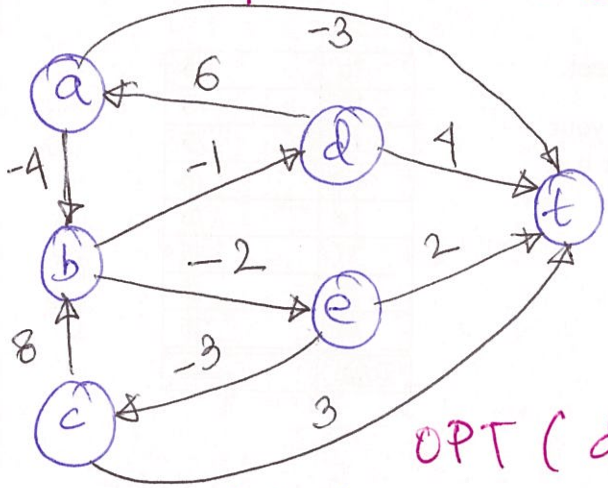


$OPT(s, i)$ $\quad$ $s \in V$ $\quad$ $0 \le i \le n-1$ $\Big\}$ # subproblems $= n \cdot n = n^2$ ✓

Q: What are final value(s) we want to compute?

A: $OPT(s, n-1)$ as by Prop every s has a shortest s-t path that is simple $\Rightarrow$ has $\le n-1$ edges

Goal: Compute $OPT(s, n-1)$ $\forall s \in V$



Focus on vertex d

$OPT(d, 0) = \infty$ $\quad$ (as $d \neq t$)

$OPT(d, 1) = 4$ $\quad$ [d, t]

$OPT(d, 2) = 6 - 3 = 3$ $\quad$ [d, a, t]

$OPT(d, 3) = 3$ $\quad$ [d, a, t]

$OPT(d, 4) = 6 - 4 - 2 + 2 = 2$ $\quad$ [d, a, b, e, t]

$OPT(d, 5) = 6 - 4 - 2 - 3 + 3 = 0$ $\quad$ [d, a, b, e, c, t]

$\Big\{ \begin{array}{l} OPT(d, 6) = 0 \\ OPT(d, 7) = OPT(d, 8) \ldots = 0 \end{array}$ $\quad$ $n - 1 = 5$

from Prop

Recall: $OPT(s, i)$ = cost of shortest s-t path using $\le i$ edges.

Recurrence: $OPT(t, 0) = 0$

$OPT(u, 0) = \infty$ $\quad \forall u \neq t$

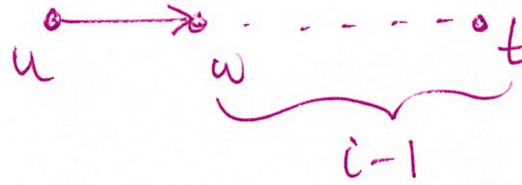
$OPT(u, i)$ for $i > 0$

Case 1: $\exists$ a shortest u-t path that uses $\le i-1$ edges.

$\Rightarrow OPT(u, i) = OPT(u, i-1)$

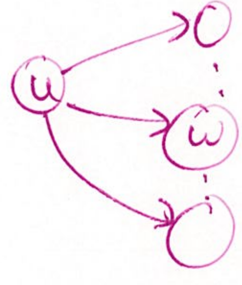
Case 2: All shortest u-t path with $\le i$ edges uses exactly i edges

Know: 1st edge path is (u,w)



$$OPT(u,i) = c_{u,w} + OPT(w, i-1)$$

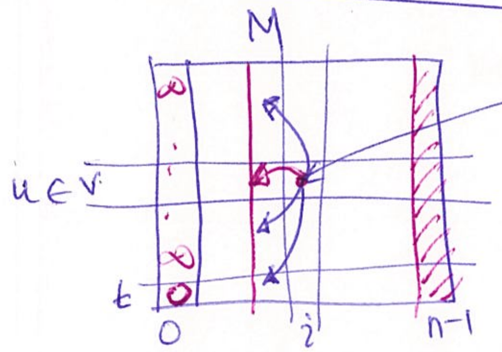
In general:



$$OPT(u,i) = \min_{\substack{w:\\ (u,w)\in E}} \{c_{u,w} + OPT(w,i-1)\}$$

OVERALL:

$$\boxed{OPT(u,i) = \min \left\{ OPT(u,i-1), \min_{\substack{w:\\ (u,w)\in E}} \{c_{u,w} + OPT(w,i-1)\}\right\}}$$



$M[u,i] = OPT(u,i)$

Output: $M[s,n-1]$ $\forall s \in V$

③ ordering among sub-problems

Column i only depends on column $i-1$

Ordering: go column by column L to R $0 \dots n-1$

Bellman-Ford algo

Can compute each column in $O(m)$ time.

0. Allocate an $n \times n$ matrix M $\leftarrow O(n^2)$
1. $M[t,0] \leftarrow 0$, $M[u,0] \leftarrow \infty$ $\leftarrow O(n)$ $\forall u \neq t$
2. for $i = 1 \dots n-1$
   for $u \in V$
   $O(n)$ $\{ M[u,i] \leftarrow \min \{ M[u,i], \min_{\substack{w:\\ (u,w)\in E}} \{c_{u,w} + M[w, i-1]\}\}$
3. Return $M[s,n-1]$ $\forall s \in V$ $\leftarrow O(n)$

$O(n^3)$ improve to $O(mn)$

for a given $u \in V$