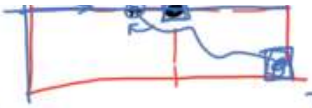
$(A[1\cdots i], B[1-j])$



$T(0,j) = j$
$T(i,0) = i$

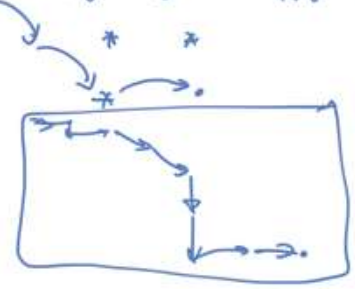
* $T(i,j) = \min\left(T(i,j-1)+1,\ T(i-1,j)+1,\ T(i-1,j-1) \right)$ if only if $A[i] = B[j]$.
  $i=0, j=0$

for $i = 1, \ldots, n$
  for $j = 1, \ldots, m$ ⊕

$O(nm)$ time.

$A[1..n]$ $A[1-i]$
$B[1..m]$ $B[1-j]$

i, j, k



Maximum Flows:

electrical current, water, traffic....

directed graph

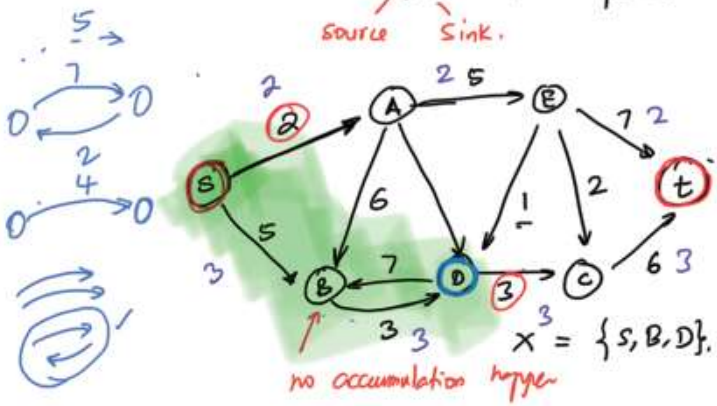
Problem : G, e has a capacity $u_e \geq 0$

s, t : two special vertices

source sink.

Greedy
Divide & Conquer
Dynamic Programming
Maximum Flows

Convention: No edge enters s,
Notation: No edge leaves t.



- Electric current :
  each edge is a wire
  u_e : max. current that can flow on e.

- Water pipes :
  each edge is a water pipe
  u_e : rate at which water can flow on this pipe.

- traffic : each each is a road.
  u_e : how much traffic flows on each edge (rate of traffic)
  eg, 10 trucks / hour.

What do we want ?

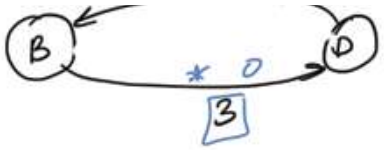
Defn: A flow f specifies a quantity f_e for each edge. :

(i) $0 \leq f_e \leq u_e$ $\forall$ edge e.

(ii) "flow conservation"
for every vertex v other than s or t,

$$\sum_{e:\ e \text{ comes into } v} f_e = \sum_{e:\ e \text{ goes out of } v} f_e$$

$\delta^-(v)$: edges entering v.
$\delta^+(v)$: edges leaving v.



??

Let f be flow.

flow going out of s := $\sum_{e \text{ leaving } s} f_e$

flow coming into t = $\sum_{e \text{ enter } t} f_e$

<u>Claim</u>: Flow coming into t = Flow going out of s.

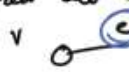
<u>Pf</u>: For every vertex $v \neq s,t$

$$\sum_{e \in \delta^-(v)} f_e = \sum_{e \in \delta^+(v)} f_e \quad \forall v$$

total flow entering v — total flow leaving v

value of the flow ≡ flow going out of s.

Add all of these equations:



e appears on the RHS for v and on the LHS for w.

$$\sum_{e' \in \delta^-(w)} f_{e'} = \sum_{e' \in \delta^+(w)} f_{e'} \quad : w$$

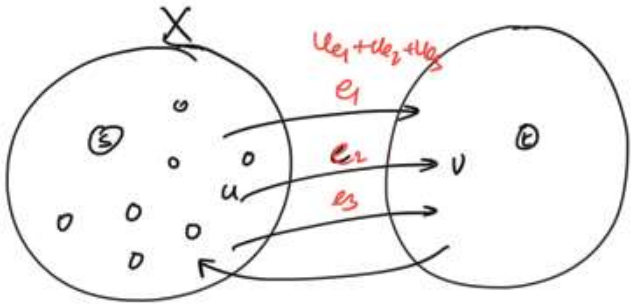
appears on the RHS for w.

when we add all of these equations,

any such edge appears on the LHS for v

$$\sum_{e \in \delta^+(s)} f_e = \sum_{e \in \delta^-(t)} f_e$$

<u>Goal</u>: To find a flow of maximum value



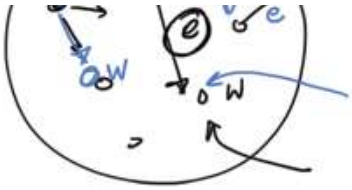
<u>Defn</u>: A s-t <u>cut</u> is a set of vertices X such that X contains s and does not contain t.

We say that $e = (u,v)$ leaves X if $u \in X$, $v \notin X$.

$e = (u,v)$ enters X if $u \notin X$, $v \in X$.

Capacity $(X) = \sum_{e \text{ leaving } X} u_e$

Claim: Max. flow = capacity(X). ||
from s to t



Pf: For every vertex $v \in X$, $v \neq s$

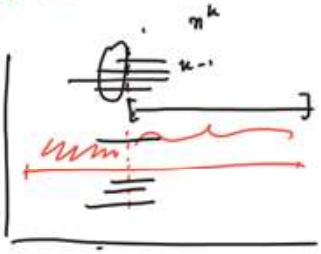
$$\sum_{e \in \delta^-(v)} f_e = \sum_{e \in \delta^+(v)} f_e.$$

Add all of the equations

$$\sum_{e \in \delta^+(s)} f_e = \underbrace{\sum_{\substack{e \text{ going out} \\ \text{of } X}} f_e - \sum_{\substack{e \text{ coming into} \\ X}} f_e}_{\geq 0}$$

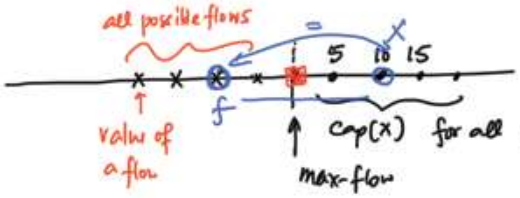
$$\leq \sum_{e \text{ going out of } X} u_e = \text{capacity}(X).$$

$\forall X$: $\min_X$ capacity(X)



28/10/21 | 75% post mine 1.

Every cut X gives us an upper bound on the max flow from s-t.



Suppose we find a flow f and an s-t cut X
value of f = cap(X).
⇒ f is max-flow & X is min cut.

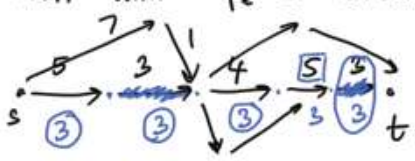
max flow $\leq$ min cap(X). : is there equality too?
= X: X is an s-t cut

Thm: max flow from s-t = min capacity of a cut. ||. ✓

Alg: start by building flow incrementally, and stop when its value = capacity of a cut. ||.

(i) Start with $f_e = 0$ on all edges. v

(ii)

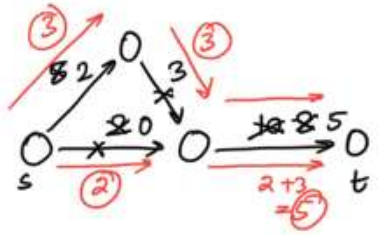


continue - Find a path from s to t.

P: Let c be the min. capacity of an

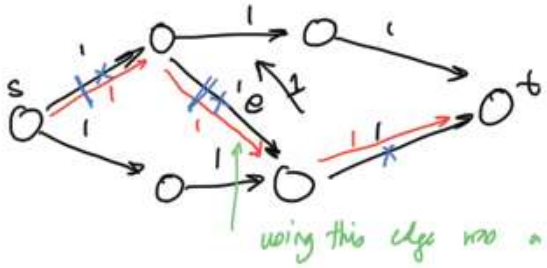
- edge in P. ~~Send~~ Increase c amount of flow on all edges in P.
- update the capacity of every edge to $u_e - f_e$: remove edges of capacity 0.

residual capacity.



Is this going to find the maximum flow? NO

(11)



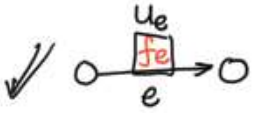
max flow = 2.

using this edge was a "mistake"

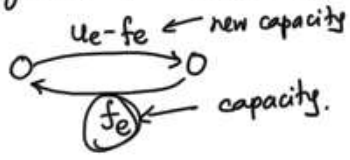
Modify the algorithm:

New Idea:

Allow the algorithm to "undo" a mistake.



≡



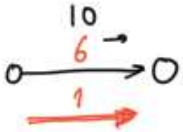
G_f

Residual graph.

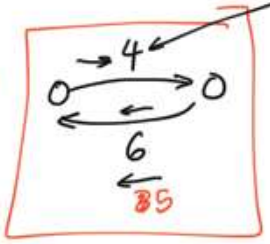
G, flow f

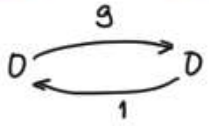
residual cap.

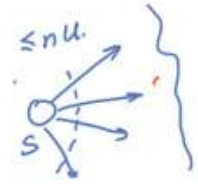
Ex:



G.







G, f_e

Initialize $f=0$, $G_f = G$

repeat:

$O(m+n)$ time

find a path P from s to t in G_f.

Let c be the smallest residual capacity of an edge in P.

Send c amount of flow along P.

update f in G, update G_f.

(i) Construct G_f $O(m+n)$

(ii) $s-t$ in G_f $O(m+n)$

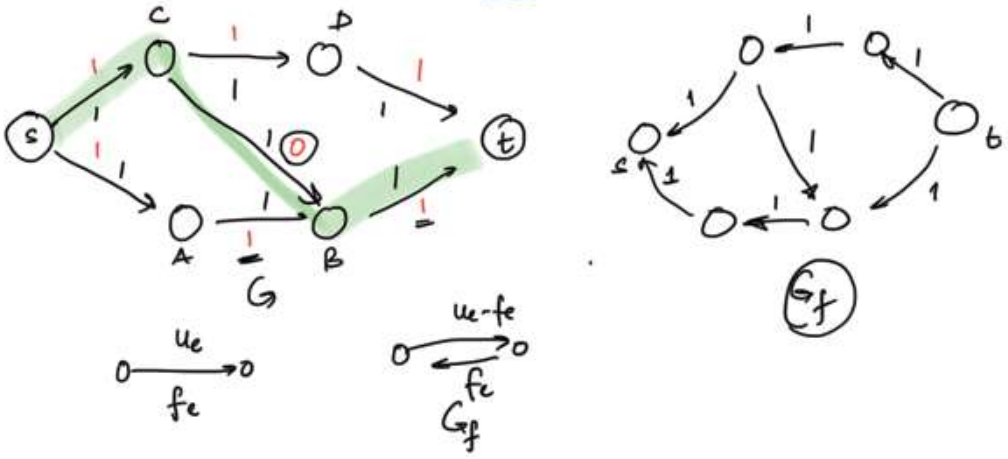
(iii) $O(n)$

flow inc. $0 \to nU$

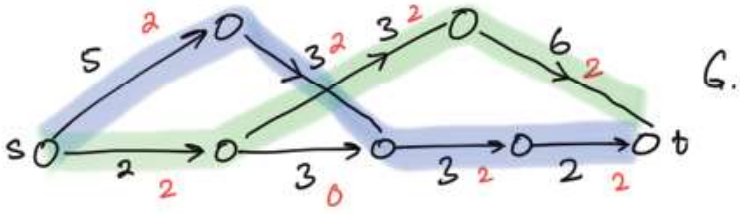
≥ 1 unit

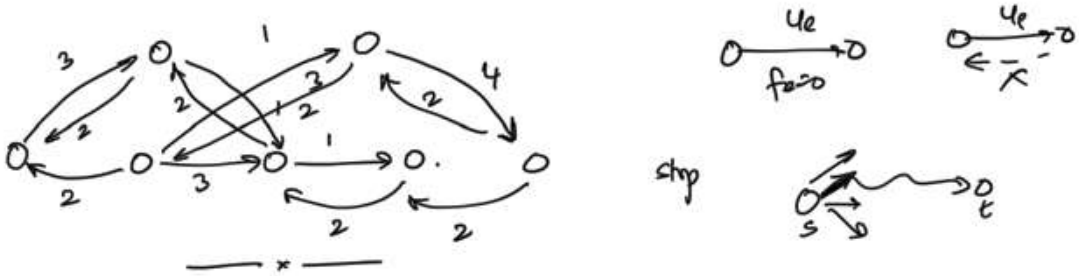
$\Rightarrow \leq nU$ iterations. ✓

$O(mC)$.



Another Example:





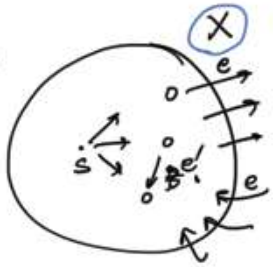
Whenever we find a path from s to t in G_f, we increase the flow from s to t

the process stops when there is no path from s to t in G_f.

Claim: If there is no path from s to t in G_f, then f is a max. flow.

Pf: (Last time we had shown)

$\forall$ f is any flow, $\forall$ X, f $\quad$ value(f) = cap(X)

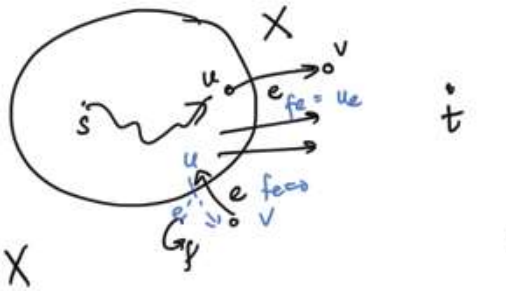


flow out of X = flow going out of X − flow coming into X

$$\text{value}(f) = \sum_{e \text{ leaving } X} f_e - \sum_{e \text{ entering } X} f_e = \sum_{e \text{ leaving } X} u_e = \text{cap}(X).$$

Suppose there is no path from s to t in G_f.

Let $X = \{v : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$.



$s \in X,\ t \notin X$

(i) Let e be an edge leaving X : why isn't v also in X?
e cannot be present in G_f. $f_e = u_e$

$f_e = u_e$
o ⟶ o

(ii) e enters X : e' be the reversal of e. e' is not present in G_f (else $v \in X$)

$f_e = 0$

f is a max flow & X is a min-cut max flow $\le nU$.

Ford-Fulkerson Alg: Assume all capacities are integers, U: max. cap. of an edge.

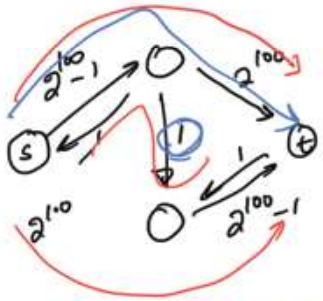
In each iteration: - form G_f $O(m)$ time.
- increase the flow by ≥ 1 unit

$\le nU$ iterations, $O(mnU)$ time.

$O(mnU)$ time. 2^{100}

Comment: all cap. are integ. $1, U$.

(i) if U is large, then alg. can take $\Omega(n)$ time.



- Shortest path in G_f (use BFS) : $O(mn^2)$ time.
- find a path where the min. residual cap. is as large as

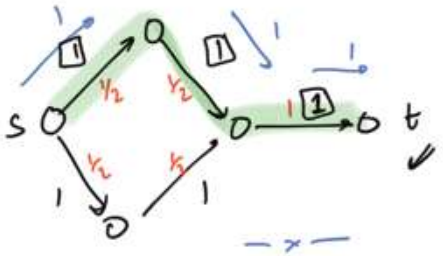
11 possible. $O(mn^2 \log U)$ time.

(ii) If all capacities are integer, then the alg. always deals with integers.
⇒ the max flow found by the alg. sends integral amount of flow on each edge.

↓

there is a max flow which only sends integer flows.

Integrality of Max-Flow.

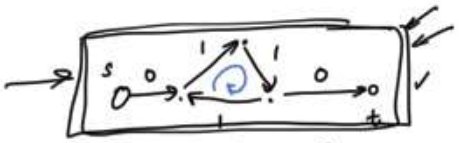


— x —

1/11/21. (i) Max flow = Min. cut, alg. for finding these.

(ii) Integrality of max-flow. : there is a max-flow which is integral.

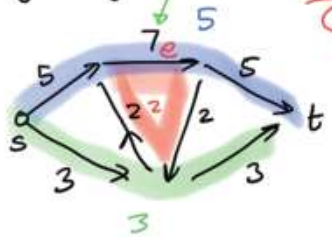
(iii) How do we think of a flow?? Can we think of a flow as consisting of flows along paths?

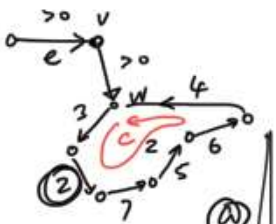


Thm [Path decomposition of flow] Let f be any flow in G. Then we can find s-t paths $P_1, \dots, P_\ell$ and values $v_1, \dots, v_\ell$ and cycles $C_1, \dots, C_k$ and values $w_1, \dots, w_k$ such that for every edge e,

$$f_e = \sum_{P_i \text{ contains } e} v_i + \sum_{C_j \text{ contains } e} w_j \qquad : k+\ell \le m.$$

Pf: H: subgraph of G consisting of edges with $f_e > 0$.



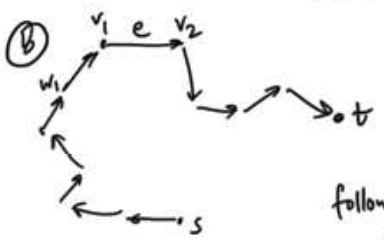


e∈H

(A) ✓ (i) Repeat a vertex : we have a cycle C with $f_e > 0$ on all edges.

(ii) Min. f_e on an edge in C = Δ and subtract Δ from each edge in C.

(B) ✓ ≥1 edge drops out of H



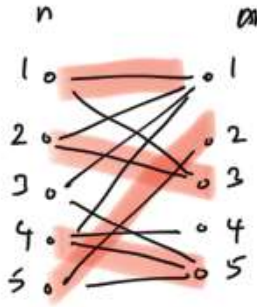
follow backwards incoming edges from v: either find a cycle or a walk from s-t

Applications of Max flow:

$U = 1$ $O(mn)$.

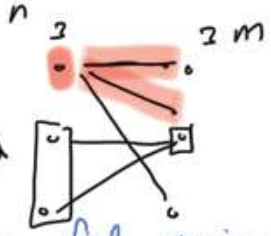
(1) Bipartite Matching:

Matching: subset of edges which do not share a common vertex.



e.g. V_1: set of student
V_2: set of hotel rooms.

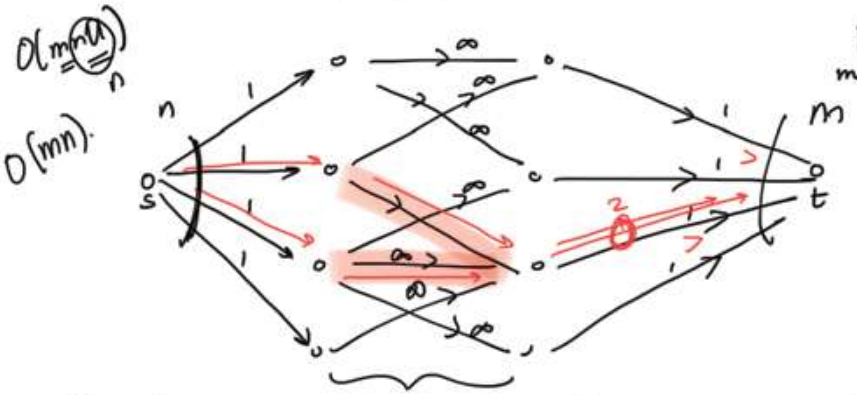
- Can we have a matching of size n ??
- What is the max. size of a matching?



2.

s, t ve directed

How do we use max flow to find maximum matching?



P → P'
matching max-flow.

$O(m\sqrt{n})$

G'

Thm: there is a flow of value k from s to t in G' if and only if there is a matching of size k in G.

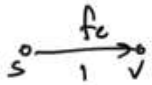
If: (⇐) Suppose there a matching of size k in G. : for every edge $e = (u,v)$ in the matching, send 1 unit of flow along

P_e : s, u, v, t

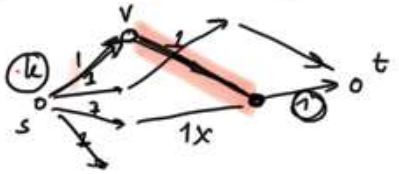
All of them paths are disjoint.

(⇒) Suppose there a flow f of value k from s to t.

we can assume that f is integral



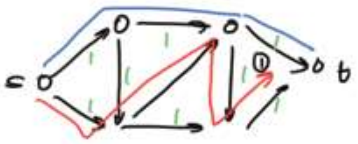
$f_e = 0$ or 1.



Look at the edges between V_1 and V_2 which are carrying 1 unit of flow. There will be k such edges and they will form a matching.

(2) Disjoint paths:

given a directed graph G. We say that two paths



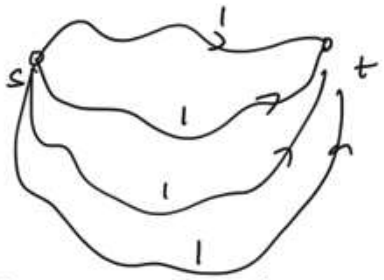
from s to t are "edge" disjoint if they do not share a common edge.

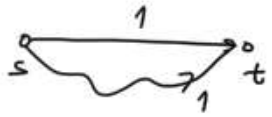
- we want to find the maximum no. of edge disjoint paths from s to b.

Place a capacity of 1 on every edge find max-flow from s to b.

Claim: there is a flow of value k from s to t if and only if there are k edge disjoint paths from s to b.

Pf: ($\Leftarrow$) Suppose are k- edge- disjoint paths from s to t.





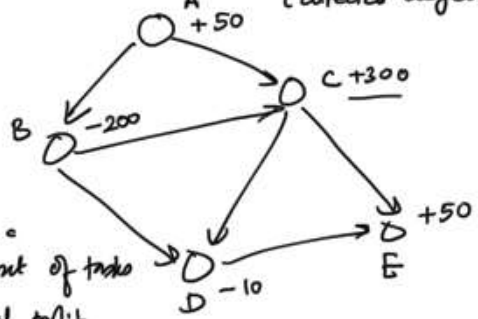
($\Rightarrow$) Suppose there is a flow of value k from s to b. (follows from path decomposition).

(3) Application of min-cut :

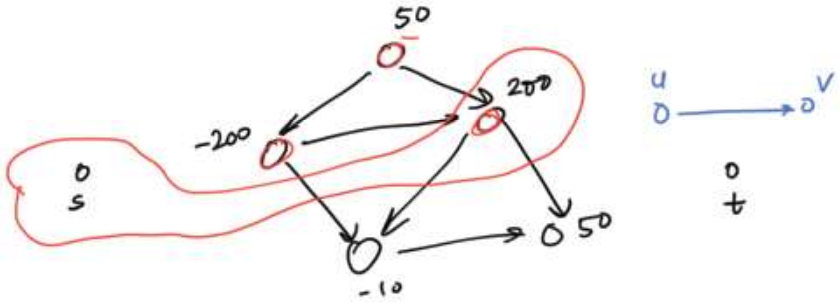
Project Selection Problem: DAG. precedence constrain tasks (directed acyclic graph).

Every task either generates or costs money.

v : p_v



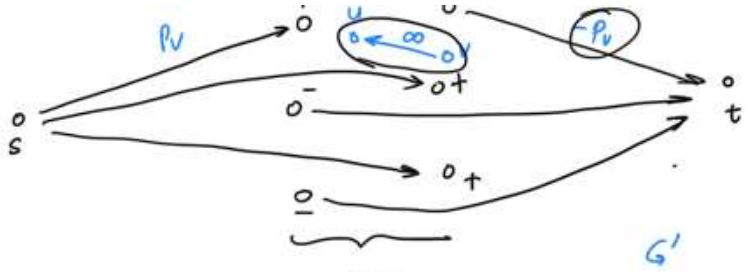
Q: we want to perform a subset of tasks such that the total profit is maximized.



Idea: Add a vertex s, t .... min-cut :

How do we ensure that the min-cut is valid ??

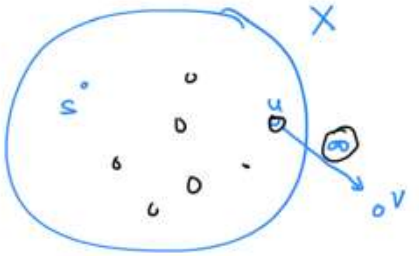
If $(u,v) \in G$



then in G', we add (v,u) with ∞ capacity.

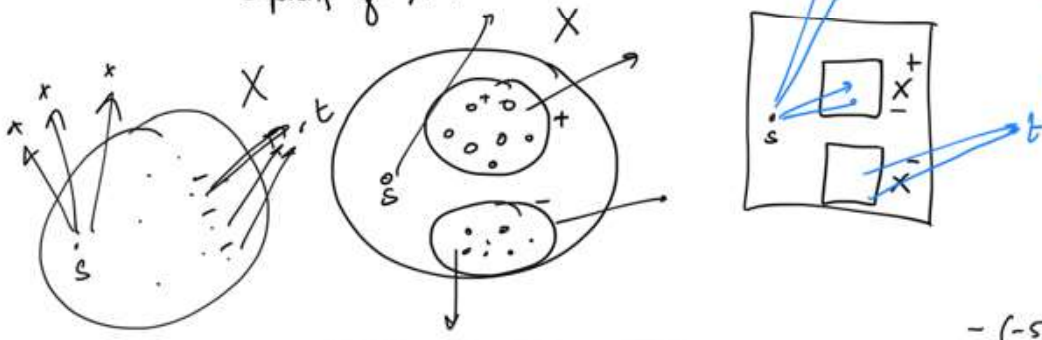
DAG

Claim: find a min-cut of separating s and t.



Let X be the min-cut.
$u \in X$. $v \to u$ in G.
if $v \notin X$, then (u,v) has cap. ∞. The cut X has ∞ capacity.

Capacity of X ?



$$\infty \quad \underbrace{cap(X)}_{a} = \underbrace{\sum_{v \in V^+ - X^+} p_v}_{} + \underbrace{\sum_{v \in X^-} (-p_v)}_{} \qquad -(-5\infty) = 5\infty$$

$$= \sum_{v \in V^+} p_v - \sum_{v \in X^+} p_v - \sum_{v \in X^-} p_v$$

$$= \underbrace{\sum_{v \in V^+} p_v}_{\text{fixed number.}} - \underbrace{\left[\sum_{v \in X} p_v\right]}_{\text{net profit of } X.}$$

Min of cap(X) = max profit of X.

$a + b = C$.

$$= \underbrace{\sum_{v \in V^+} p_v}_{=C} - \underbrace{\sum_{v \in X} p_v}_{\text{net profit of } X.}$$