Applications of cuts and flows.

Many applications

- some algorithmic
- some combinatorial
- very important black box
  to know about as an algorithm
  designer and a graph theorist

This lecture:

Three applications.

① An algorithm for maximum matching in bipartite graphs.

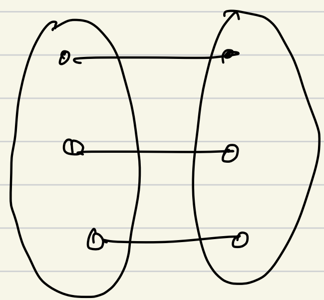
② An algorithm for vertex cover on bipartite graphs

③ A proof of Hall's theorem

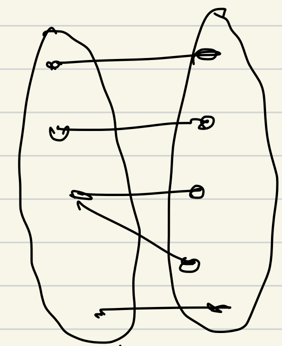
Notation: All graphs in this lecture will be undirected, all flow networks are directed.

Max Matching on Bipartite Graphs

Matching : - subset of edges that do not share a common vertex

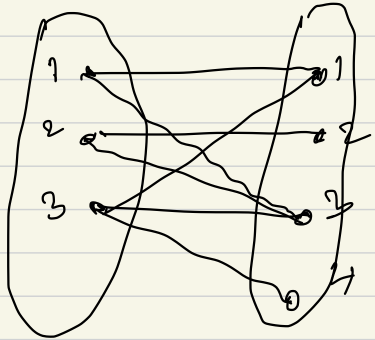


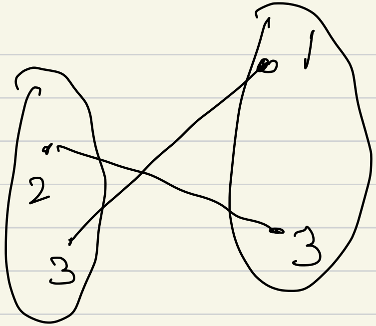
Matching

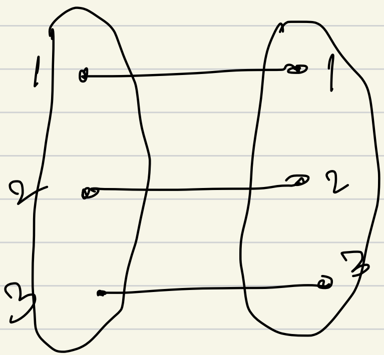


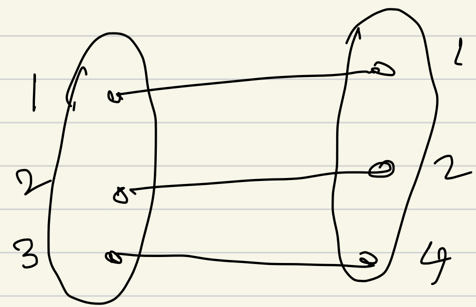
Not a matching

Maximum matching : matching M of largest # edges







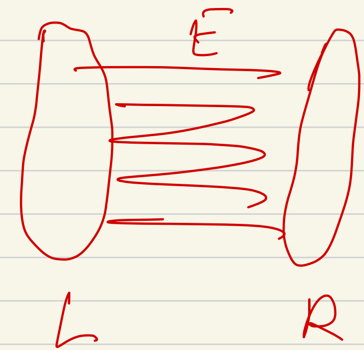


Max matching may not be unique.

Computing the max bipartite matching.

Input: A bipartite graph $G = (L, R, E)$

Output: the max matching M in G.



- Want to design an efficient algorithm for this.

- Question makes sense over non bipartite graphs as well, and very efficient (but not very easy) algorithm is known over these as well.

- Edmond's Algorithm due to Jack Edmonds
  - check out his picture :)

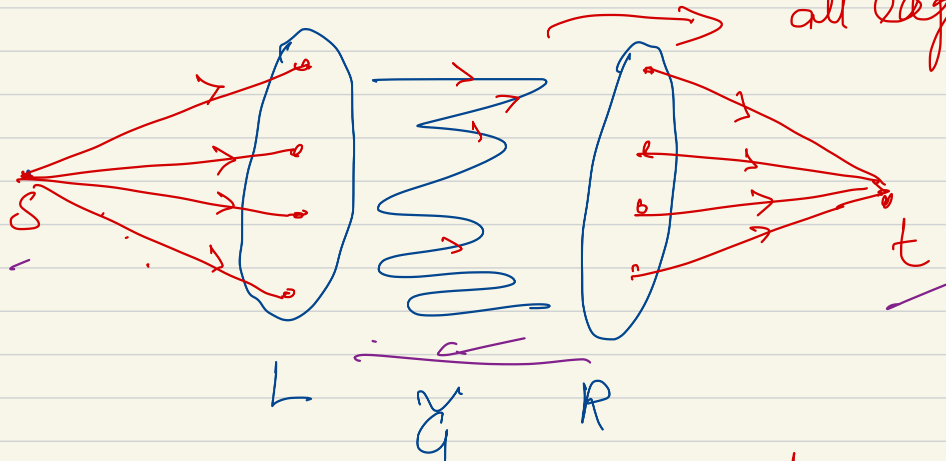
- We will not see this in this class though.

Max Matching in bipartite ghs

via max cut

① From a bipartite graph to a flow network

all edges left to right



$u(e) = 1$ for all edges (old as well as new)

② Bipartite matching in G $\Leftrightarrow$ max flow in $\tilde{G}$

Lemma 1

Max value of $s \to t$ flow in $\tilde{G}$

= size of the max matching in G.

Pf:

① Matching to flow.

M = max matching

$|M| = k$.

Is there a flow of cardinality $\geq k$?

$\sim$ Matching = $\{(u_i, v_i) : i \in \{1, 2, \ldots, k\}\}$

$$\left\{\begin{array}{l} f(u_i, v_i) = 1 \\ f(s, u_i) = 1 \\ f(v_i, t) = 1 \end{array}\right\}$$

$u \in L$
$v \in R$

(2) f - max flow. $val(f) = k$.

want to show that there is a matching of size $\geq k$.

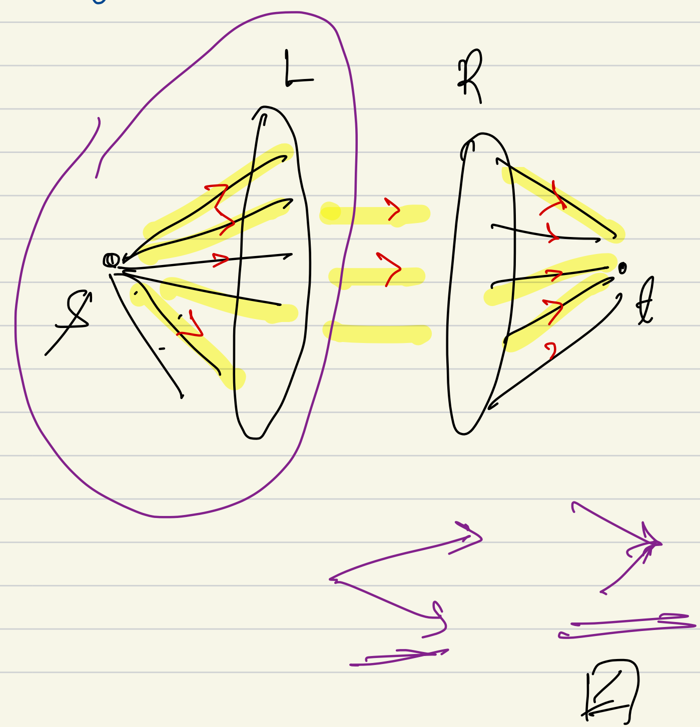
Recall: if the edge capacities are integral, then max flow can be assumed to be integral, without loss of generality.

Here: edge capacity 0/1.

So, an integral flow assigns 0/1 to each edge

So, what does an integral flow on $\mathcal{G}$ look like?

Do you see a matching of cardinality k in $\mathcal{G}$ from here?



From the Ford-Fulkerson algorithm for max flow, we get:

Theorem: Max matching in a bipartite graph can be found in time $O(nm)$.

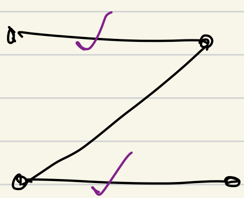
n = # vertices

m = # edges.

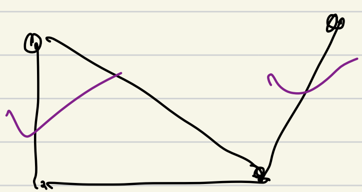
Appl. 2 : Perfect Matching in bipartite graphs.

Perfect Matching

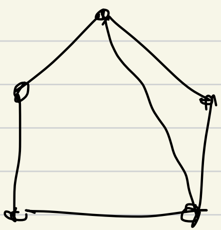
A graph G is said to have a perfect matching if there is a matching M in G such that all the vertices of G are matched in M.



1



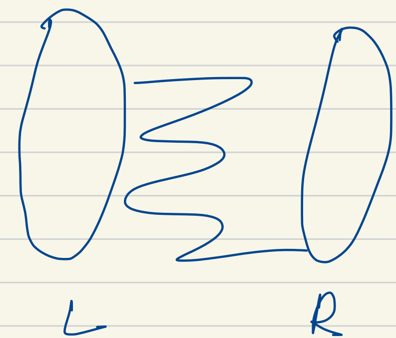
2



3

Perfect matchings?

Q: When does a bipartite graph have a perfect matching?



Necessary condition:

$$|L| = |R|$$

clearly not sufficient . . .

- Run the max matching algorithm that we saw.
- if $|M| = |L| = |R|$, then there is a perfect matching.

Is there a clean combinatorial condition on G to have a perfect matching?

Theorem [Hall's theorem]

A bipartite graph $G = (L, R, E)$ with $|L| = |R|$
has a perfect matching
if and only if

$$\forall\, S \subseteq L,\ |N(S)| \geq |S|,$$

$$N(S) = \{ w \in R \text{ s.t } \exists\, v \in S \text{ with } (v, w) \in E \}$$

↳ neighborhood of S.

- A very useful, very important theorem.
- profound and surprising consequences and applications.
- Many proofs known
- Here: a proof via flows and cuts.

Proof:

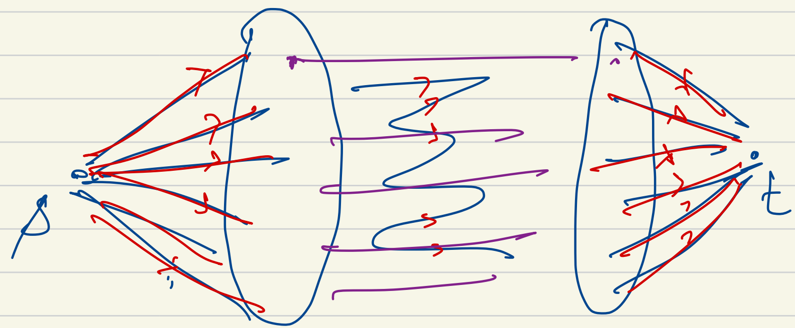
Easy direction:

Perfect Matching $\Rightarrow \forall S \subseteq L, |N(S)| \geq |S|$

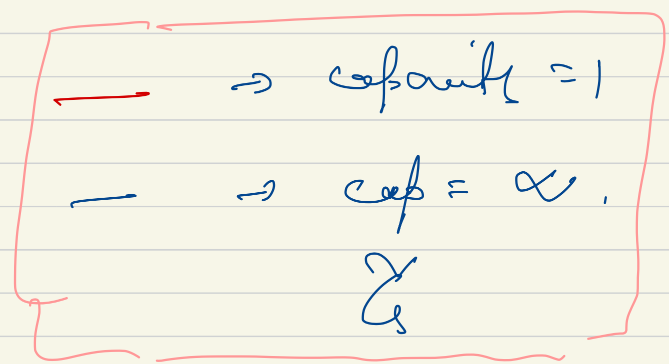
why?

Not so easy direction:

A flow network.



What can we say about the max flow in G?



(a) Is the max flow value even finite?

(b) What is the max flow value? How large can it get?

Plan:

Claim: min cut value in $\tilde{G}$ is at least $|L|$.

Using the claim to get a perfect matching.

from (b) max flow $\leq |L| = |R|$

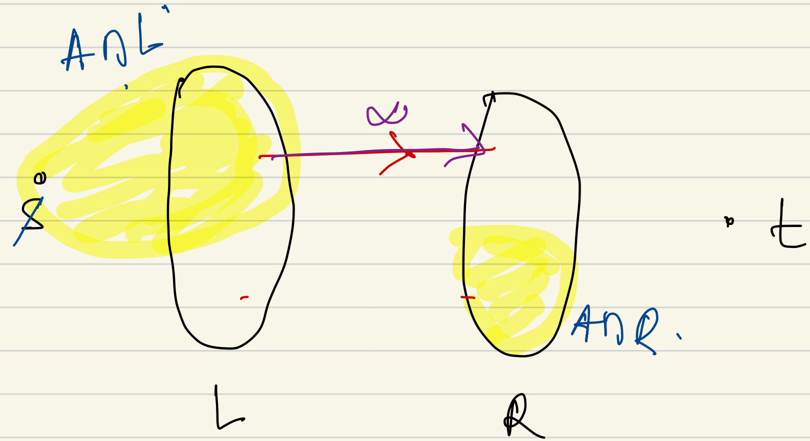
from Thm max flow $=$ min cut $\leq |L|$

From claim: $|L| \le$ max flow $=$ min cut $\le |L|$

$\Rightarrow$ max flow $=$ min cut $= |L|$.

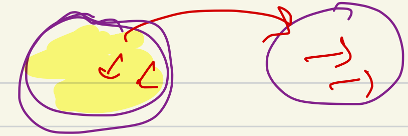
How does this give a perfect matching in G. ?

Proof of Claim: Min-cut $\geq |L|$.

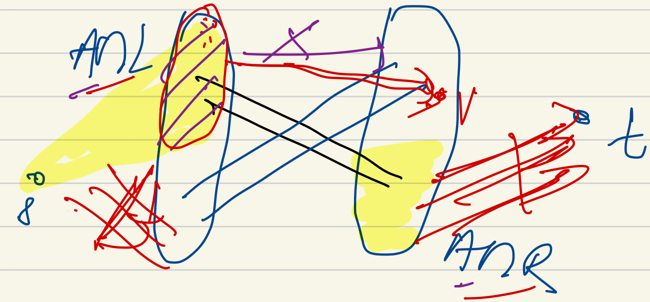


$A \rightarrow$ arbitrary s-t cut

Obs 1: If there is a Red edge, then cap(A) $= \infty$.
and done.



Else: No Red edges.



$\Rightarrow N(A \cap L) \subseteq A \cap R.$

Capacity of this cut:

$$|L| - |A \cap L| + |A \cap R|$$

Where are the edges going across the cut? from A to $V \setminus A$

But, $A \cap R \supseteq N(A \cap L)$

$\Rightarrow |A \cap R| \geq N(A \cap L) \geq |A \cap L|$

$$\Rightarrow \quad |L| + \left(|A \cap R| - |A \cap L|\right)$$

$$\Rightarrow \quad |L| + \left(|N(A \cap L)| - |A \cap L|\right)$$

$$\Rightarrow \quad |L|$$

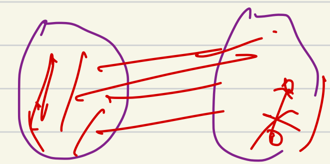
Vertex cover in bipartite graphs.

Vertex cover: $G = (V, E)$ — a graph, undirected

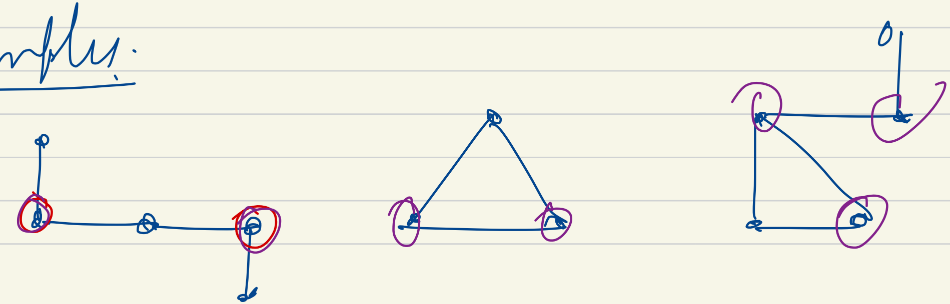
$A \subseteq V$ is said to be a vertex cover of G

if

$\forall\, (x,y) \in E$, at least one of x, y is in A.



Examples:



- V is always a vertex cover.

Minimum Vertex cover:

Vertex cover of smallest cardinality.

Want: an efficient algo for min vertex cover

— Known to be NP hard on general graphs.

Here: an efficient algorithm for vertex cover on bipartite graphs.

— via max cut – min flow machinery.

Min VC on bipartite graphs

Input: Bip graph $G = (L, R, E)$

Output: Min cardinality vertex cover of G.

In fact: will solve a slightly stronger problem.

Min weight vertex cover on bipartite graphs.

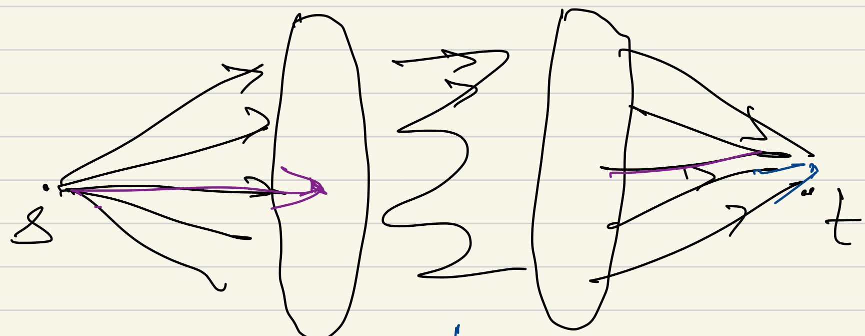
Input: Bip graph $G = (L, R, E)$, weight fn $w : L \cup R \to \mathbb{Z}_{\geq 0}$

Output: Vertex cover A of G s.t

$$w(A) = \sum_{v \in A} w(v)$$ is minimized.

Again:

- will construct a flow network $\tilde{G}$ from G
- relate min VC in G to min cut on $\tilde{G}$



$u(s,v) = w(v)$

$u(e) = \infty$.

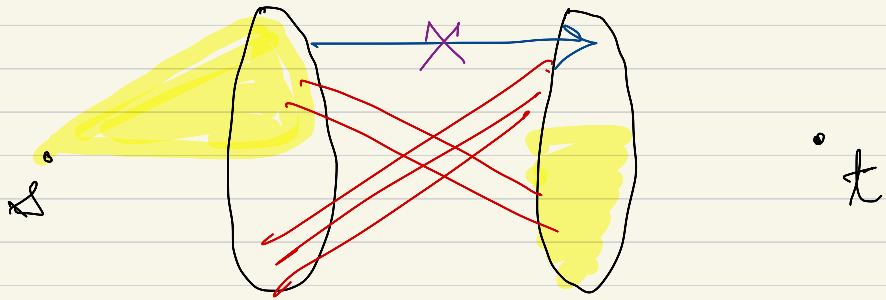
$u(v,t) = w(v)$

From the discussion on Hall's theorem:

- Max flow / min cut values are finite inspite of edge capacities of some edges being infinite.

What do min s-t cuts with finite capacity look like?

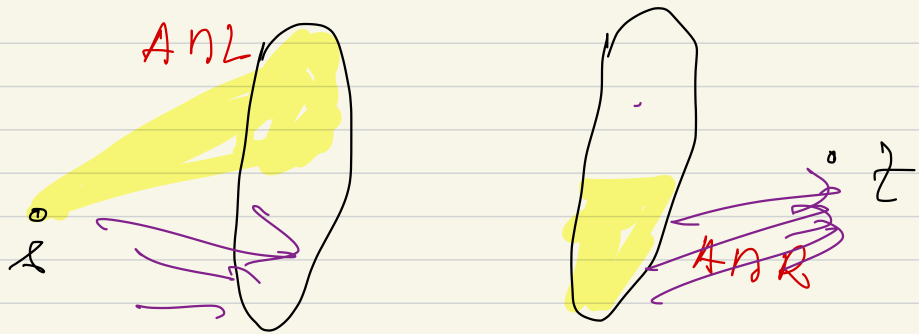
Saw this in Hall's theorem.



No blue edge : - -

and this is Hall's theorem

Can you spot a vertex cover of G from the above picture?



- $A \cap L \cup R \setminus (A \cap R)$
- $L \setminus (A \cap L) \cup A \cap R$ ✓
- L
- R

vertex covers.

wt of $L \setminus (A \cap L) \cup A \cap R$

$$= \sum_{v \in L \setminus (A \cap L)} w(v) + \sum_{v \in A \cap R} w(v)$$

$$= \underbrace{\sum_{v \in L \setminus (A \cap L)} u(s, v)} + \underbrace{\sum_{v \in A \cap R} u(v, t)}$$

= capacity of the cut A.

Lemma 1:

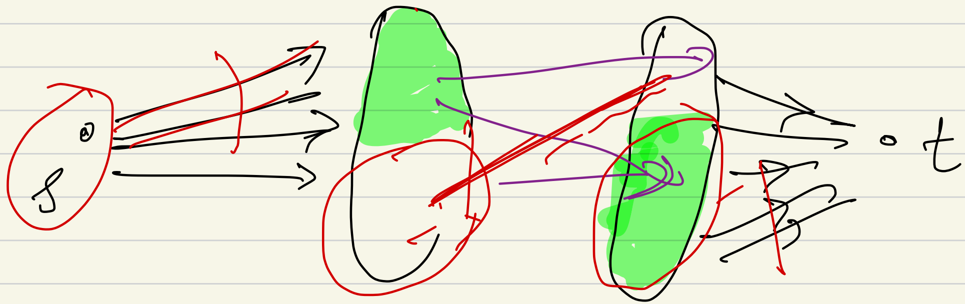
A — any s-t cut in $\tilde{G}$ with finite capacity.

Then, $L \setminus (A \cap L) \cup A \cap R$ is a vertex cover of G with weight of VC = capacity of A.

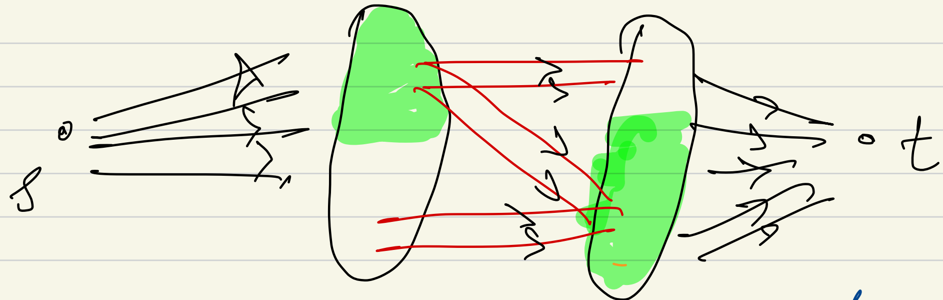
Pf:

$\square$

From vertex covers to s-t cuts.

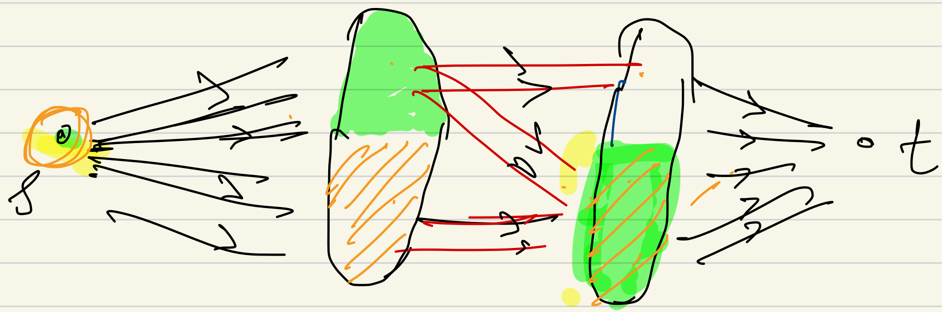


Where are the edges?



Do we see an s-t cut with finite capacity here?

No white-white edges.



capacity of shaded cut?

Lemma 2 (Converse)

Given a vertex cover B of G, there is s-t cut in $\tilde{G}$

$L \setminus B \cup R \cap B \cup \{s\}$

s.t weight of B = capacity of the cut.

From Lemma ① and Lem ② $\Rightarrow$ can use the FF algo on $\check{G}$ to find the min vertex cover in G.

$\blacksquare$