CS 314 - Network Flow Applications

Note Title 4/19/2010

Announcements

- HW due Wednesday in class
- Next HW up - due next Wednesday (in class & written)
- Final - May 10, 12-2pm

Network Flow

$C = \sum_{e \text{ out of } s} c_e$ or $C = \sum_{e \text{ into } t} c_e$

- Have $O(mC)$ algorithm to find maximum flow in a network.

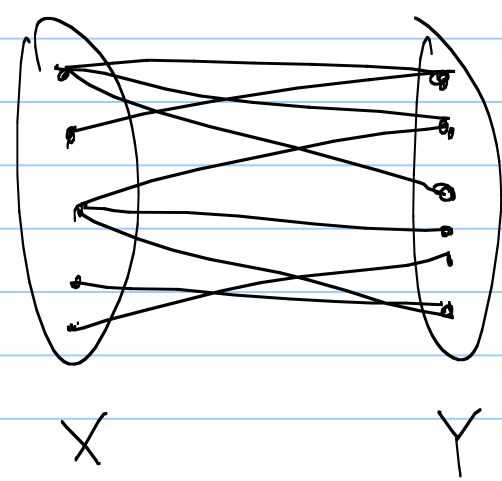
- Sections 7.3 + 7.4 discuss alternate implementations that run in:
  - $O(m^2 \log_2 C)$
  - $O(mn)$
  - $O(n^3)$

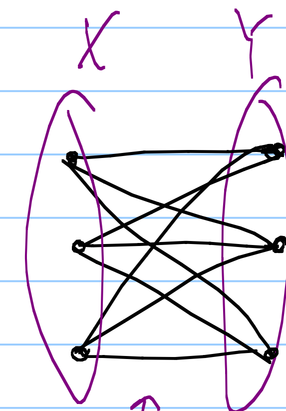
(We may come back to some of these later on...)

7.5: Bipartite Matching

Dfn: A *bipartite graph* is an undirected graph $G=(V,E)$ whose vertex set can be partitioned into $X \cup Y = V$, where every edge in E has one endpoint in X and the other in Y.

Ex:

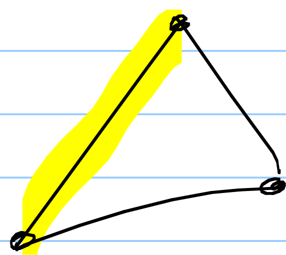


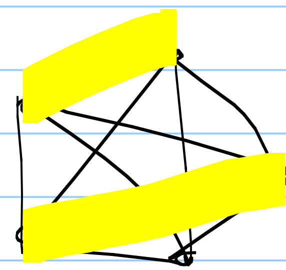


Complete bipartite graph $K_{3,3}$

Dfn: A *matching* in a graph G is a subset $M \subseteq E$ of edges such that each vertex of G appears in *at most* one edge of M.

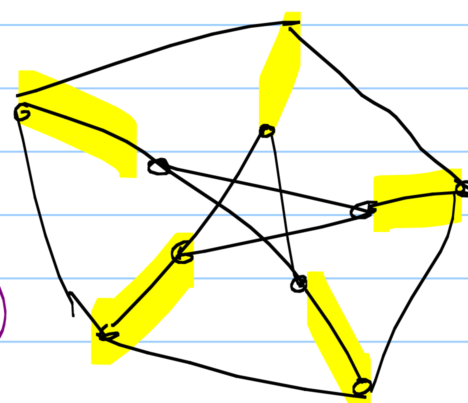
Ex:

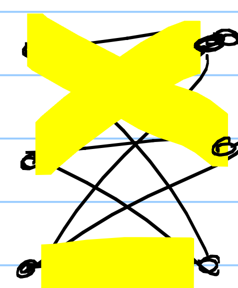


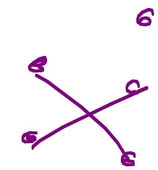


largest possible matching has $\frac{n}{2}$ edges

perfect matching





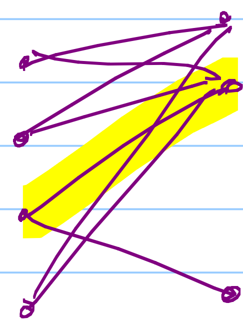


Bipartite Matching Problem

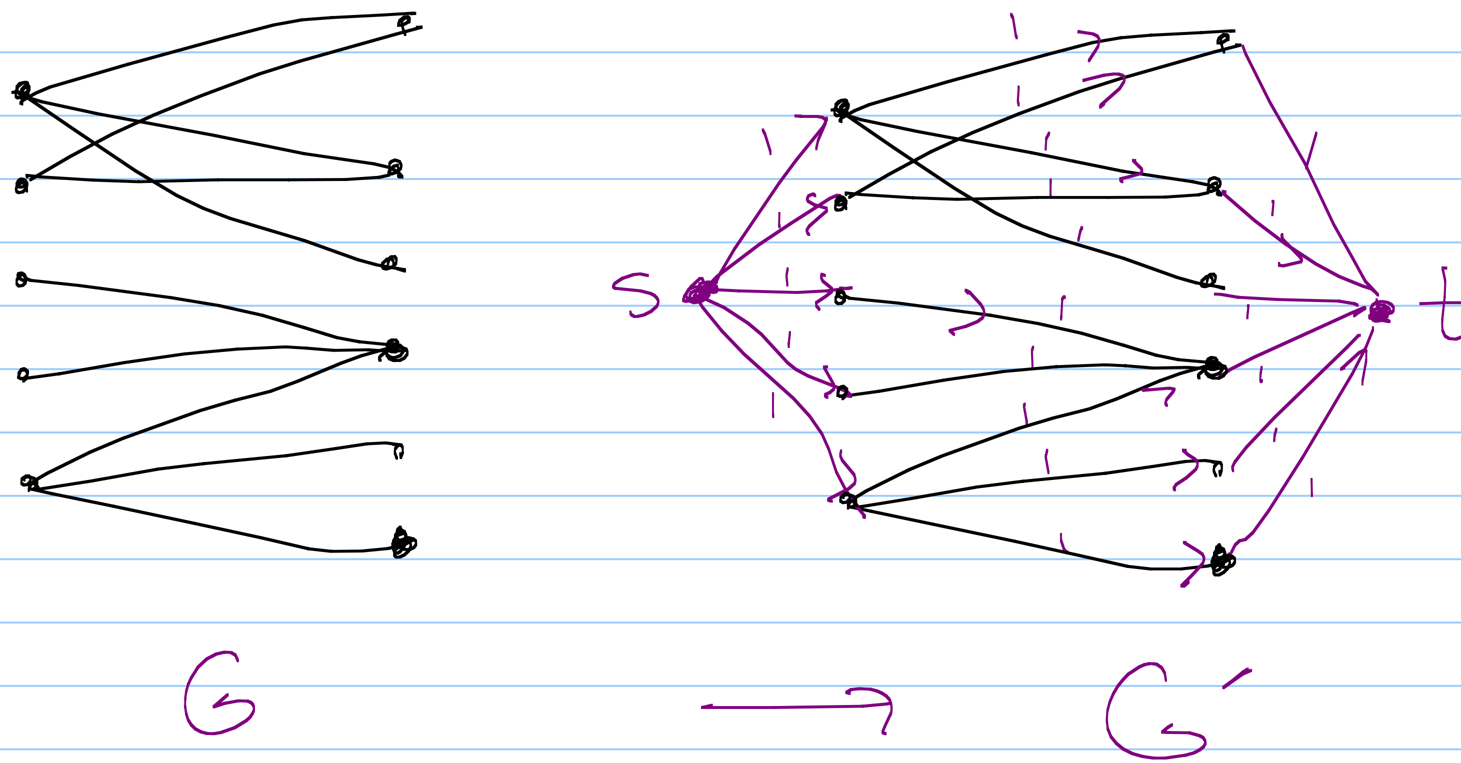
Given a bipartite graph G, find a matching with the largest possible size.

Ideas? Start with nodes that have low degree.

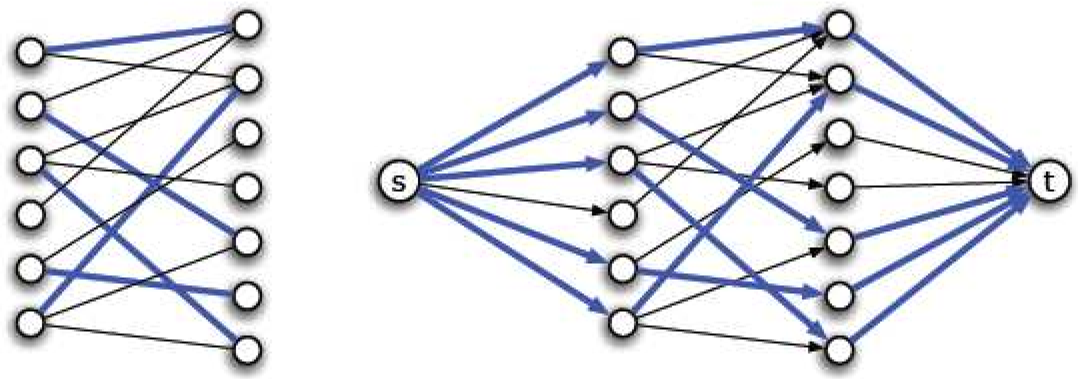
greedy



We'll reduce this to a flow problem.



Claim: G has a matching of size k $\iff$ G' has a flow of value k.

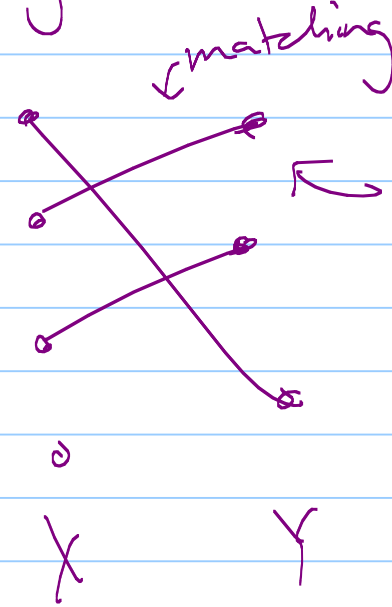


A maximum matching in a bipartite graph G, and the corresponding maximum flow in G'.

proof :

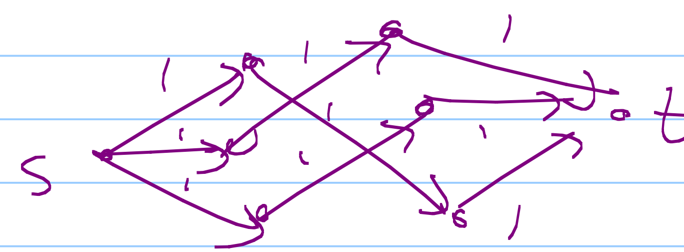
matching (in G) $\Rightarrow$ flow (in G') :

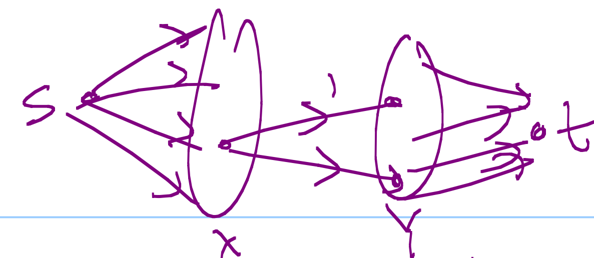
Spps we have a matching of K edges in G.



← these edges in G' get a flow of value 1

add a flow of value 1 going into every vertex of X, + out of every vertex of Y





<u>proof cont:</u>

Have a flow of value k in G!

<u>flow ⇒ matching</u>: Consider the edges from X to Y used in flow.

Form M by taking the X to Y edge in G.

① M contains k edges

$\{s\} \cup X$ + $\{t\} \cup Y$ is a cut

flow across it has value k.

So # edges going from X to Y must be k.

② Each node in X is the tail of at most one edge in M.

conservation constraint:

Spps $v \in X$ appears twice in M.

Then 2 units of flow leave v.

Impossible, since only 1 edge goes into v.

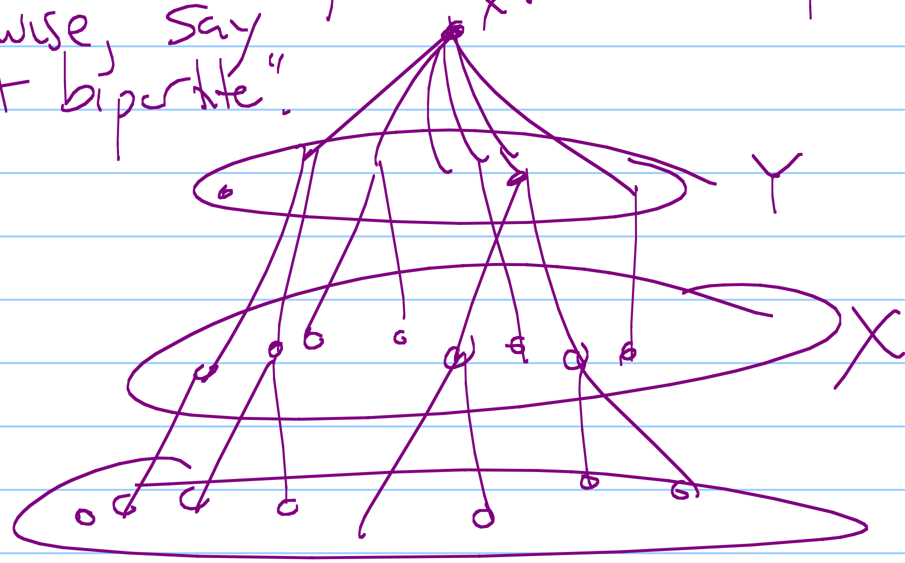
Runtime: Input graph G had n vertices
& m edges.

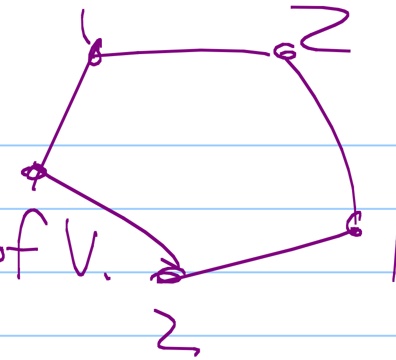
Added 2 vertices & n edges to get G'
$C \le n$

$$\Rightarrow O(mC) = O(\underbrace{(m+n)}_{\text{added } n \text{ edges to get } G'} n) = O(mn)$$

Bipartite testing

If G is bipartite, find partition of V.
Otherwise, say "not bipartite".

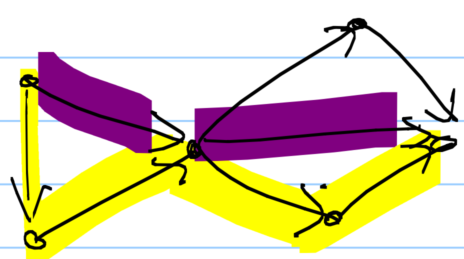




O(m)

7.6: Disjoint Paths in Directed and Undirected graphs

Dfn: Two paths are edge-disjoint if they do not have any edges in common.



Goal: Find the maximum number of edge disjoint paths between 2 vertices s & t.

Given a directed graph G, & s & t how can we reduce this to a flow problem?

- Give every edge capacity 1.

  Find max flow in G!

  Return that value.

<u>Claim:</u> G has k edge-disjoint s-t paths
$\iff$ G' has a flow of value k.

<u>pf:</u> <u>$\Rightarrow$:</u> Given k s to t paths
send 1 unit of flow
along each.

<u>$\Leftarrow$:</u> Induction on # edges in the flow.