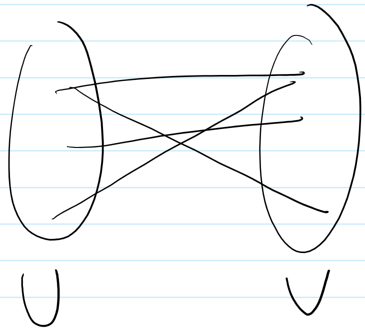
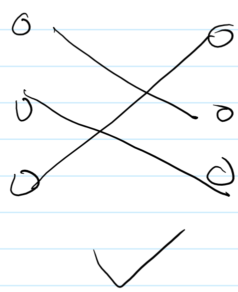
COMPSCI 330 Lecture 12 Bipartite Matching

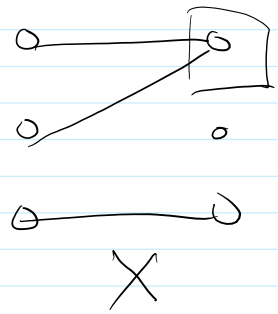
Wednesday, October 5, 2016 2:52 PM

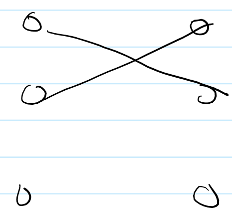
- Recall: Bipartite Graph: (U, V, E), every edge $e = (u,v)$, $u \in U$, $v \in V$



- matching: a subset of edges that do not share vertices.

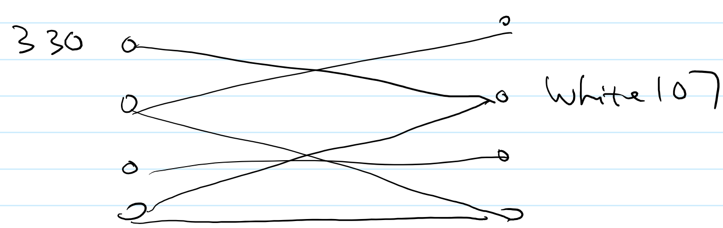






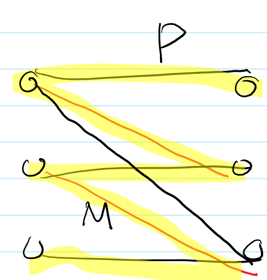
matching, not perfect

- perfect matching: if all vertices are connected to 1 edge in matching.
- example Courses classrooms

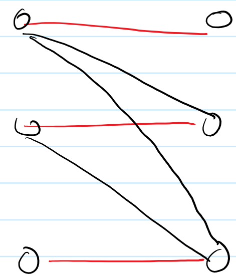


- problem: Given bipartite graph G, find the matching with maximum number of edges.

- augmenting path
  - red: matched edge
  - cannot add edge to matching
  - but maximum matching is larger
  - Q: how to improve this matching?
  - augmenting path: path connecting an unmatched vertex in U



- augmenting path: path connecting an unmatched vertex in U and an unmatched vertex in V, and alternates between unmatched and matched edges.



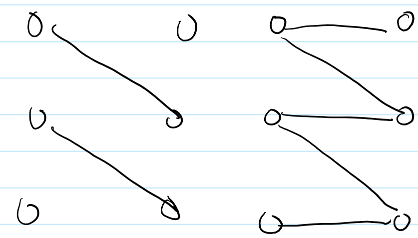
$M \oplus P$

- augment

"xor" operation: S_1, S_2 are two sets of edges

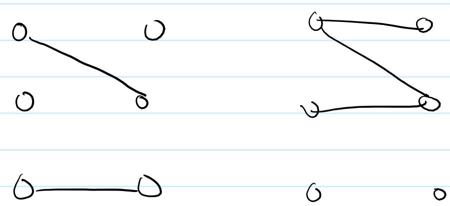
$e \in S_1 \oplus S_2$ if $e \in S_1, e \notin S_2$ or $e \in S_2, e \notin S_1$

Claim: if M is a matching, P is an augmenting path (with respect to M) $M \oplus P$ is a matching with one more edge than M.



M P

Proof: by picture.



- Augmenting path algorithm

$M = \phi$

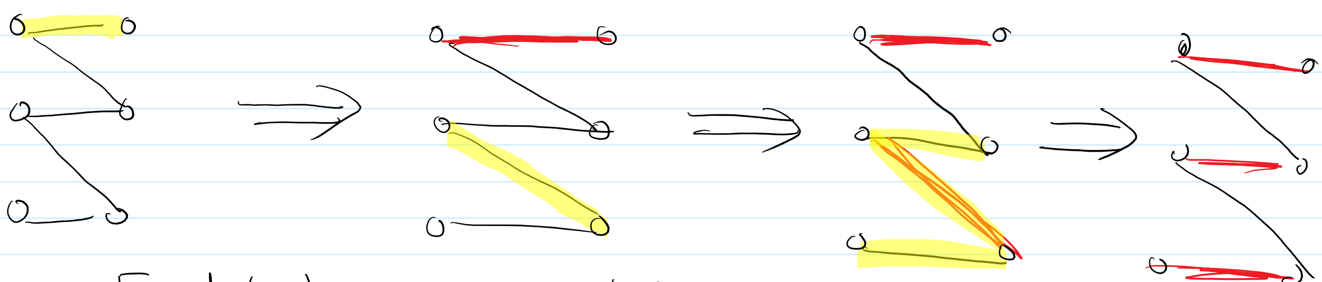
repeat

try to find augmenting path P (DFS/BFS) M

if found, $M = M \oplus P$

until there is no augmenting path

P



Find (u) ← u is on Left side of graph, u unmatched

mark u as visited

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

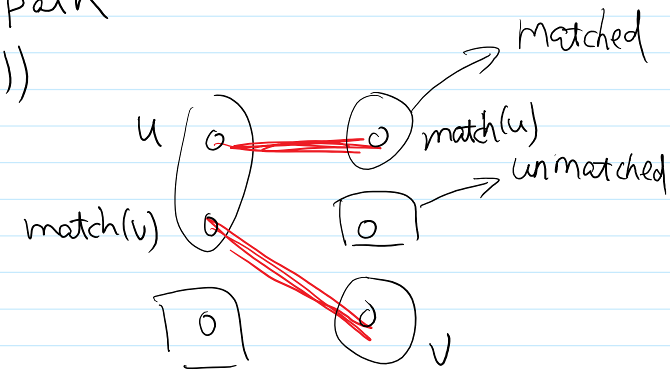
if v is not matched

found augmenting path

If v is not matched

found augmenting path

else Find(match(v))

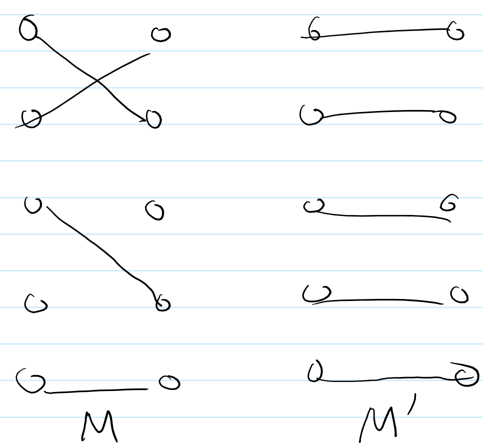


— Correctness: Theorem: if M is not a maximum matching, then there exists an augmenting path.

Proof: Assume M' is a maximum matching

$$|M'| > |M|$$

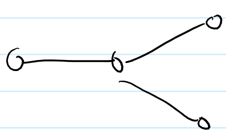
consider $M \oplus M'$



"degree": # edges adjacent to a vertex

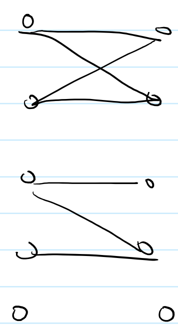
degree of any vertex in $M \oplus M' \leq 2$.

$\Longrightarrow$ no branching

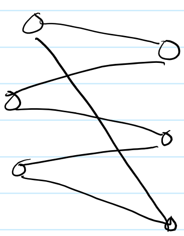


branching

connected components are of the following cases

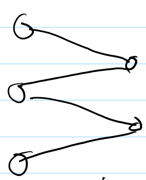


cycle



same # edges in M, M'

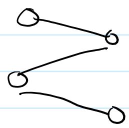
path



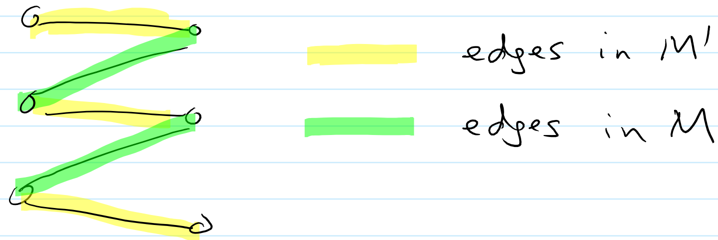
even length

same # edges in M, M'

path



recall $|M'| > |M| \Rightarrow$ there must be a component in $M \oplus M'$ that has more edges in M'



This is an augmenting path! $\square$

- running time: $\leq n$ iterations

  each iteration runs DFS $O(n+m)$

  $O(n^2+nm)$ $\quad (O(nm))$