27 Aug 2021 CS 6820
Analysis of Algorithms

https://www.cs.cornell.edu/courses/cs6820/2021fa

Recommended: D. Kozen, "Design & Analysis of Algs"

Matchings in Bipartite Graphs

A graph $G=(V,E)$ is *bipartite* if its vertices can be partitioned into two sets L, R such that every edge has one endpoint in L and the other in R.

Exercise. Show that there is a linear time algorithm, given the adj list of G, that either:

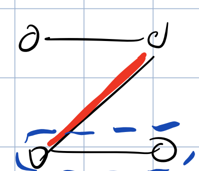
(a) outputs a partition $V = L \cup R$ as above, or

(b) finds a cycle of odd length.

Def. A matching in a graph G is a set of edges that have no endpoints in common.

BIPARTITE MAXIMUM MATCHING PROBLEM: Given bipartite G, find a matching with max # of edges.

BIPARTITE PERFECT MATCHING: Given bipartite G, decide whether $\exists$ a matching such that every vertex belongs to an edge. If so, find one such matching.



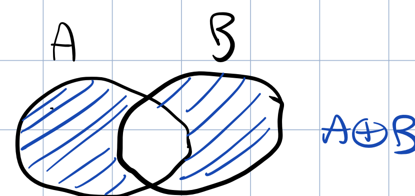
Def. If G is a graph, M is a matching, P is a path:

- P is a M-alternating if its edges alternate between belonging, not belonging to M.
- P is M-augmenting if it is

M-alternating and its first and last edges don't belong to M, and the endpoints of P don't belong to any edges of M.

Lemma. If M is a matching and P is an M-augmenting path then $M \oplus P$ (the symmetric difference of M and P) is a matching with one more edge than M.

$M \leftarrow M \oplus P$



Lemma 2. If M_1, M_2 are two matchings in G and $|M_2| > |M_1|$ then $M_1 \oplus M_2$ contains an M_1-augmenting path.

In particular if M_1 is not a maximum matching then G must contain an M_1-augmenting path.

Proof of Lemma 2. In the edge set $M_1 \oplus M_2$ every vertex belongs to ≤ 2 edges and every connected component of $M_1 \oplus M_2$ is an isolated vertex or a path or cycle that is alternating w.r.t. both M_1 & M_2.

type	(# M_2 edges) − (# M_1 edges)
cycle	$\emptyset$
even lgth paths	$\emptyset$
odd lgth paths starting @ M_2 ending @ M_2	1
odd lgth paths start @ M_1 end @ M_1	−1
isolated vertex	$\emptyset$

Because $|M_2| > |M_1|$ there is at least one component of this type

These are M_1-augmenting paths.

To compute a max matching in G:

1. initialize $M = \emptyset$
2. while $\exists$ an M-augmenting path P

   $M \leftarrow M \oplus P$

   endwhile
3. output M

Lemma 1 $\Longrightarrow$ while loop iterates $\leq \frac{1}{2}|V|$ times

Lemma 2 $\Longrightarrow$ termination condition for while loop guarantees we output a maximum matching.

Coming Monday: an efficient algorithm to find M-augmenting path when G is bipartite.