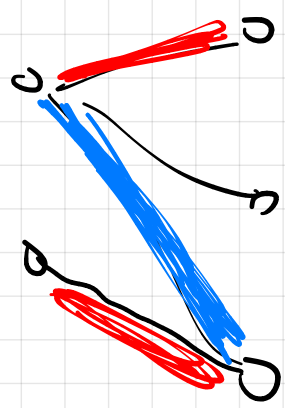
21 Aug 2023 CS 6820: Analysis of Algorithms

A matching in a graph is an edge set s.t. each vertex belongs to at most one edge.

("exactly one" .... perfect matching)

maximum matching problem: given undirected $G=(V,E)$ find a matching of max cardinality.

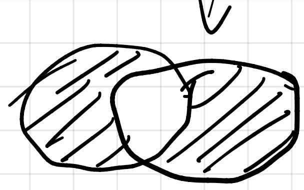
G:



If G is a graph and M is a matching:

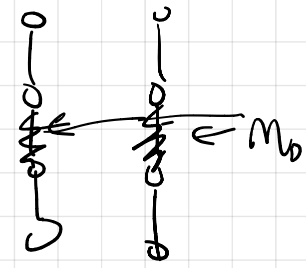
- an M-alternating path is a path in G whose edges alternate between belonging to M and not belonging.
- a free vertex (w.r.t. M) is a vertex of G that doesn't belong to any edge of M.
- an M-augmenting path is an M-alternating path between two free vertices.

Observation. If M is a matching and P is an M-augmenting path, the symmetric difference $M \oplus P$ is a matching with one more edge than M.



Lemma. If G is a graph, M_0 and M_1 are matchings, and $|M_1| > |M_0|$ then $M_0 \oplus M_1$ contains an M_0-augmenting path.

Corollary. If M_0 is not a max matching then there exists an M_0-augmenting path.



Proof of lemma.

The graph with edge set $M_0 \oplus M_1$ has max degree 2.

So its connected components are isolated vertices, paths, and cycles.

Furthermore the paths and cycles are M_0-alternating. ($\therefore$ cycles have even length)

At least one connected component of $M_0 \oplus M_1$ has more M_1 edges than M_0 edges.

It must be an M_0-augmenting path.

Generic max matching algorithm.

how to implement?

Initialize $M = \emptyset$

while G contains an M-augmenting path, P:

replace M with $M \oplus P$.

endwhile

output M.

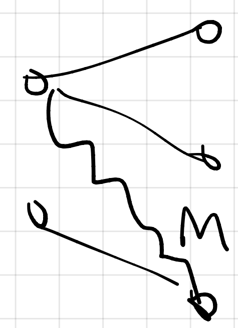
A procedure to find M-augmenting paths when G is bipartite....

say $V(G) = L \cup R$ and every edge has one endpoint in L and the other in R.

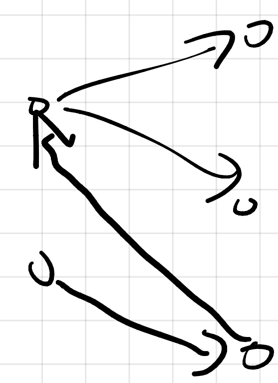
Def. The <u>residual graph</u> G_M is a directed graph with vertex set $V(G)$ and edge set

$$E(G_M) = \begin{cases} (u,v) & \text{if } u \in L,\ v \in R,\ \{u,v\} \notin M \\ (v,u) & \text{if } u \in L,\ v \in R,\ \{u,v\} \in M. \end{cases}$$

<u>Example.</u>



G_M



$\{$M-augmenting paths in $G\} \longleftrightarrow \{$directed paths in G_M from free vtx in L to free vtx in $R\}$

$\Uparrow$

BFS finds in $O(m+n)$ time

$m =$ # edges

$n =$ # vertices

The whole max-matching algo takes $O(mn + n^2)$.

(At most $\frac{n}{2}$ while loop iterations because a matching has $\leq \frac{n}{2}$ edges. Each loop iteration takes $O(m+n)$ time to construct G_M and run breadth-first search on it.)