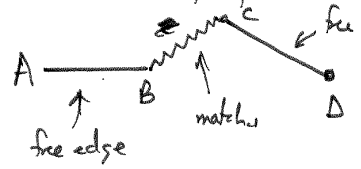
Matching Input: Undirected graph $G=(V,E)$, $w: E \to \mathbb{Z}^+$
G may not be connected.

A subset $M \subseteq E$ is called a matching if no two edges of M share a common vertex. A node is called free if there is no edge of M incident to it, otherwise it is a matched node.

Problem: (a) Find a matching M of G with maximum cardinality. — Maximum matching problem.
(b) Find a matching M of G that maximizes $\sum_{e \in M} w(e)$ — Max weighted matching problem.

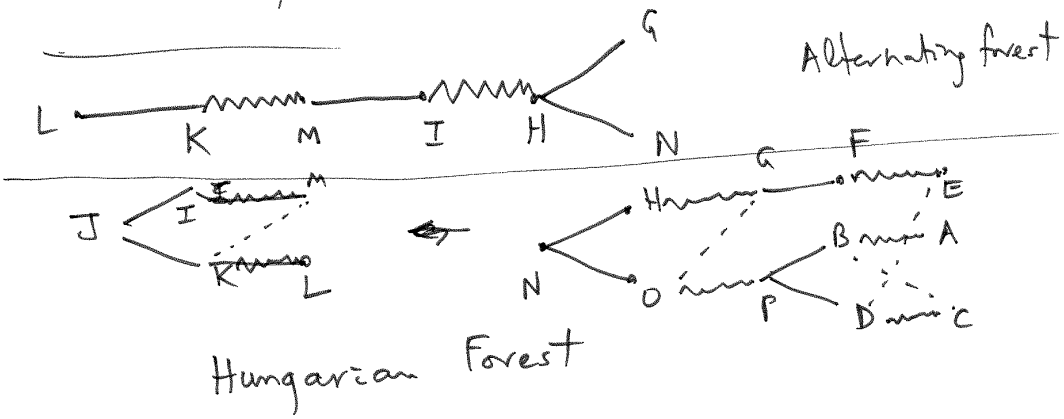
Algorithms for matching:
(i) For bipartite graphs — max matching algorithm. — easy.
(ii) For general graphs — " — requires contraction of cycles.
(iii) For bipartite graphs — weighted matching algorithm — based on Primal-Dual method of linear programming
(iv) For general graphs — weighted matching algorithm — primal-dual — much more difficult

Idea: Alternating paths: start from a free node.



starts and ends with free nodes
Swap matched & free edges on this path.

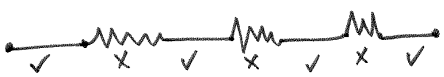
Size of matching increases by 1.



Hungarian Forest

Theory: - A path is alternating if it is comprised of free and matched edges in alternating order.

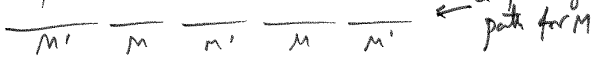
- A path is an augmenting path (i.e., if can be used to augment the cardinality of the matching) if both ends of path ~~to~~ have ~~free edges and end at~~ free nodes.



Theorem:
Let M and M' be two matching in $G=(V,E)$ and let $|M'|>|M|$
Then there exists an augmenting path with respect to M.

Proof: Consider the graph $S = M \oplus M' = \{e \mid e \in M \text{ or } e \in M' \text{ but } e \notin M \cap M'\}$

Every connected component of S is either a path or a cycle.
Every cycle has even length — equal number of edges from M and M'.
Since $|M'|>|M|$, there is a path in S with more edges of M' than edges of M.



Definitions: A matching M is called a maximal matching if there is no matching M' that has $M \subsetneq M'$.
A matching is perfect if all nodes are matched in it.

Matching in bipartite graphs

Run DFS/BFS — classify nodes as Outer & Inner. (two sides of the bipartition)
Use a queue to hold outer nodes to be processed (outer nodes in alternating forest)

Start from a free outer node.

Process Q:
Next node is u
Search over free edges from u (u,f_1) (u,f_2)
From each of $f_1 \dots f_k$ - go through a matched edge — mates of $f_1 \dots f_k$ are added to Q. $u(f_k)$
$f_1 \dots f_k$ — any free node — augmenting path found.