Lecture 13: The Bipartite Maximum Matching problem.
Date: March 4, 2026 **Scribe:** Noura Allugmani

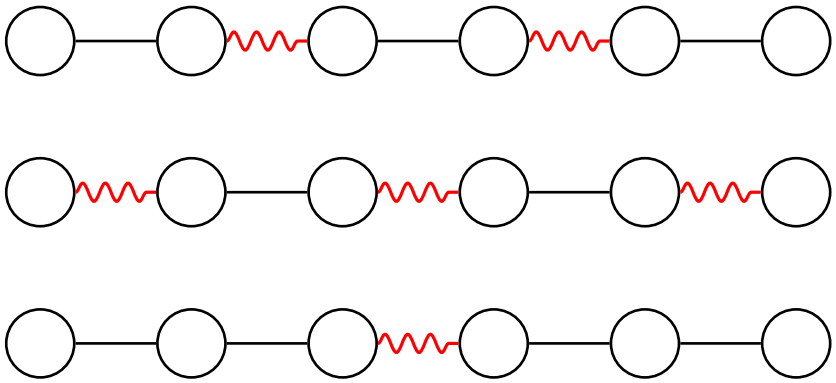
Given: a bipartite graph $G = (L, R, E)$.
Find: a matching $M \subseteq E$ of maximum size.

1 Some Terminology

Given a matching M and some $u \in V \Rightarrow u$ is "matched" if it belongs to some $e \in M$. Otherwise, u is "free". An augmenting path (with respect to M) is a path P such that:

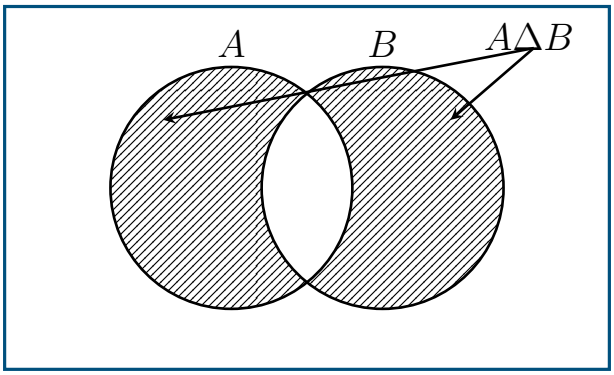
1. the endpoints of P are free w.r.t M
2. each other $u \in P$ is matched w.r.t. M.

Example 1.1.



Given tow sets A, B, let their symmetric difference be

$$A \Delta B = (A \mid B) \cup (B \backslash A).$$

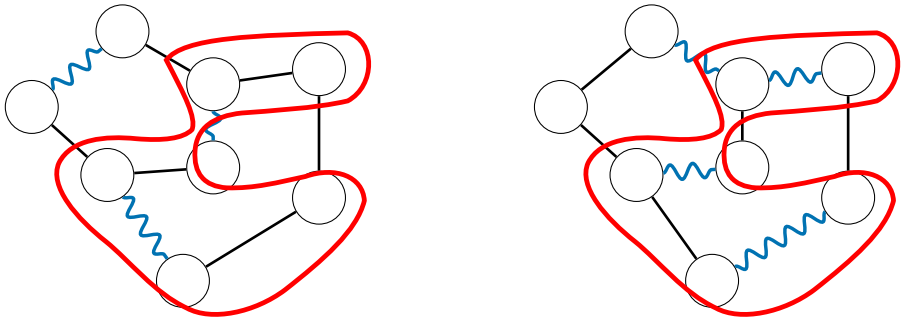


Lemma 1.2. *If M is a matchiny and P is an augmenting path w.r.t. M, the $M' = P \Delta M$ is a matching with $|M'| = |M| + 1$.*

Proof. Note that P has an odd # of edges, and its edges alternate between e s.t. $e \notin M$ and e s.t. $e \in M$. Its first and last edges, e_0, e_k are s.t. $e_0, e_k \notin M$.
$\Rightarrow M' = M \Delta P$ is s.t. $|M'| = |M| + 1$.
$\Rightarrow M'$ is a matching since:

- any $a \in V \setminus P$ has the same match in M as in M'.
- each $u \in P$ has exactly one neighbor in M'.

□



M M'

Lemma 1.3. *A matching M is a maximum matching if and only if it contains no augmenting path.*

Proof. $\Leftarrow$ we proved the contrapositive. If M has an augmenting path, then by Lemma1.2 M can not be a maximum matching.
$\Rightarrow$ Suppose M^* is a maximum matching but M is not, $\Rightarrow \quad |M| < |M^*|$. WTS that M has an augmenting path (we will use M and $M^*$to build an augments path w.r.t. M). Consider $M \Delta M^*$, and the induced subgraph $G' = (V, M \Delta M^*)$.
Claim: Each $u \in V$ had $\deg_{G'}(u) \leqslant 2$.

- if u is free in both M and M^*, deg $(u) = 0$.
- if u is matched in M but free in M^*, $\deg(u) = 1$.
  - $M \Delta M' = (M \setminus M') \cup (M^* \setminus M)$
- if u is free in M but matched in M^*, similarly $\deg(u) = 1$.
- if u is matched in both M and M^*.
  - u is matched to the same node in both M and $M^* \Rightarrow \deg(u) = 0$
- u is matched to different nodes in M and $M^* \Rightarrow \deg(u) = 2$.

Given the claim we just showed, G' can be decomposed into isolated nodes, cycles, and paths (similar to a proof we have seen a few lectures ago). Since $|M| < |M^*|$, there must exist at least on component of G' with more edges from M^* than from M.

- Can not be the isolated nodes.
- Can not be the cycles (these have edges that alternative M and M^*)

$\Rightarrow$ the component must be a path.

- Can not have an even # edges (because same counting argument as for cycle).
- must have an odd # of edges.
  - can not start / end with edges from M (opposite of what we are looking for, these have more edges from M than from M^*).
  - path must start with edges from $M^* \Rightarrow P$ must be an augmenting path w.r.t. M.

□

2 Algorithm

1. $M \leftarrow \varnothing$
2. while M has an augmenting path do

   2a Find an augmenting path P w.r.t. M

   2b $M \leftarrow M \Delta P$

return M

3 Feasibility

By lemma1.2, ALG maintains the invariant that M is a matching.

4 Termination

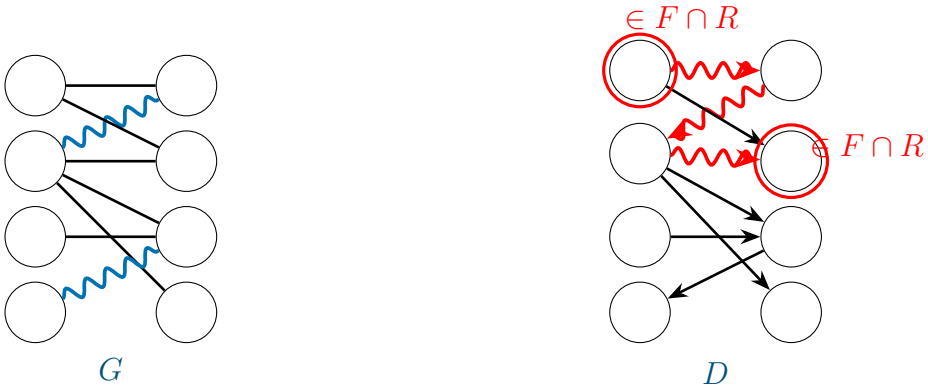
On each it iteration, $|M|$ increase by 1 (by lemma1.2). The size of a matching can not exceed $n/2$ where $n = |V|$, so ALG must terminate after $\leq \frac{n}{2}$ iterations.

5 Optimality

At some point, it exists the while loop $\Rightarrow$ find M has augmenting path. By lemma1.3, M must be a maximum matching.

Note: ALG is under specified since we have not described how to do step $(*)$ 2a.

If G is bipartite, $(*)$ is fairly easy. Given $G = (L, R, E)$ and an incumbent matching M, turn G into a directed graph D by orienting each edge $e = \{l, r\} \in E$ as:
$\rightarrow (l, r)$ if $\{l, r\} \notin M$.
$\rightarrow (r, l)$ if $\{l, r\} \in M$.



If F is the set of free nodes w.r.t. M, follow any directed path starting in $F \cap L$ and ending in $F \cap R$.
In step $(*)$, if G is bipartite, one can use standard graph traversal algorithm in D to find augmenting paths.

6 Running time (w.r.t. size of G, i.e. $n = |V|$ and $m = |E|$)

Given $f, g : \mathbb{N} \longrightarrow \mathbb{N}$, we say $f = O(g)$ if $\exists\ k \in \mathbb{N}$ and $C \in \mathbb{R}_{>0}$ s.t. $f(n) \leq Cg(n), \quad \forall n \geqslant k$.

Building D takes $O(n+m)$. Traversing D to find augmenting path takes $O(n+m)$. We may assume $m \geq \frac{n}{2}$. Otherwise $\exists$ isolated nodes that we can remove $\Rightarrow O(n+m) = O(m)$.

We saw ALG terminates after $\leq \frac{n}{2} = O(n)$ iterations. Each iteration takes $O(m)$ time. Overall, ALG terminates in $O(mn)$ time.

Can ALG be generalized to run faster or work in non-bipartite graphs?

- If you augment various augmenting paths at the same time, there is an algorithm that runs in time $O(m\sqrt{n})$ (Hopcroft–Karp algorithm).
- For non-bipartite graphs, $(*)$ is still possible but much more subtle. Edmonds' (1965) " Paths, Trees, and Flowers" that works in $O(nm^2)$ time overall. (Final project idea.)