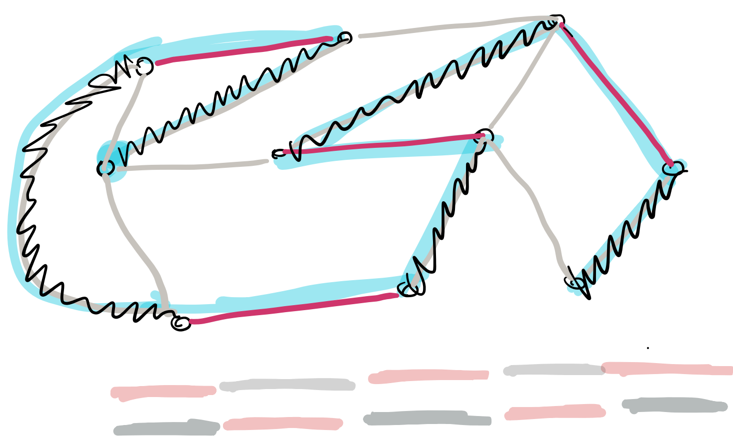
# Network Flow Applications and variants (Matchings) 11.01

Def$^n$: A matching in a graph is a collection of vertex-disjoint edges



← size of the matching is # edges ie 4

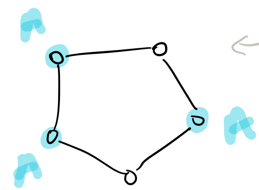
← not bipartite

A matching is maximal if there is no edge that can be added to the matching that is vertex disjoint with edges in the matching.

A matching is maximum if $\nexists$ a matching with more edges

Def$^{\underline{n}}$ A undirected bipartite graph $G=(V,E)$ is a graph whose edge set can be partitioned into $A, B$ $[\, A \cup B = V,\ \ A \cap B = \emptyset \,]$ such that

$E \subseteq A \times B$ ie all edges are between A and B.



none of these (odd cycle) in a bipartite graph

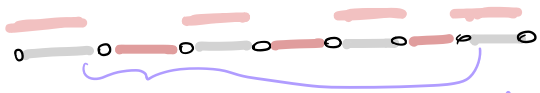
Matchings have applications in

- Flow networks
- Scheduling and planning
- Modeling bonds in Chemistry
- Graph colouring
- Stable Marriage problem
- Neural Nets, AI
- 2D, 3D image analysis + processing
- Document processing

We saw that the Augmenting Paths Alg can be used to solve Max Bipartite Matching, but when we use that method on bipartite graphs with unit-weight edges, we can speed it up to $O(\sqrt{n}\, m)$. It is called the Hopcroft-Karp-Karzanov algorithm. (1973)

**Def$^n$** A vertex that is not matched in a matching is free.
(unsaturated)

**Def$^n$** M-alternating path in a graph G with a matching M is a path



that alternates between edges $\in M$ and edges $\notin M$

An M-augmenting path is an M-alternating path that starts and ens in free vertices

**Lemma**

If G has a matching M and $\exists$ an M-alternating path P that starts and ends in free vertices, then

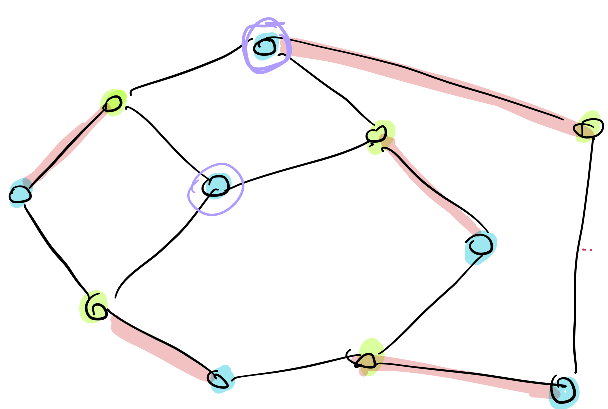
$M \oplus P$ is a matching with one more matched edge than M.

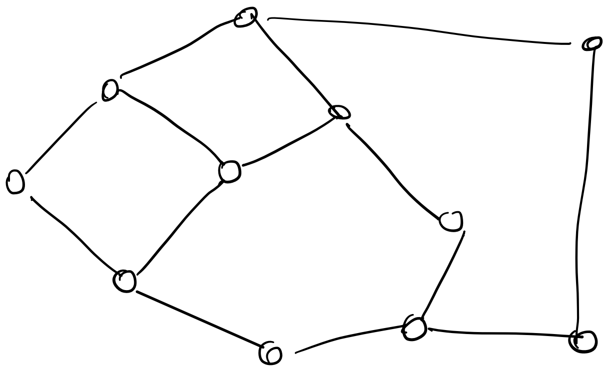
"Symmetric difference" of M and P

$$= M - \text{edges in } \underbrace{M \cap P}_{\text{in both}} + \text{edges in } P \setminus M$$

ie in P but not in M.

Eg:





## Hopcroft Karp Karzanov Alg (Maximum Matching)

Input: bipartite graph $G$

Output: maximum matching $M \subseteq E$

$M = \emptyset$

do

$\mathcal{P} = \{ P_1, P_2, .. P_k \}$ a maximal set of vertex-disjoint M-augmenting paths

$M = M \oplus (P_1 \cup P_2 \cup .. \cup P_k)$

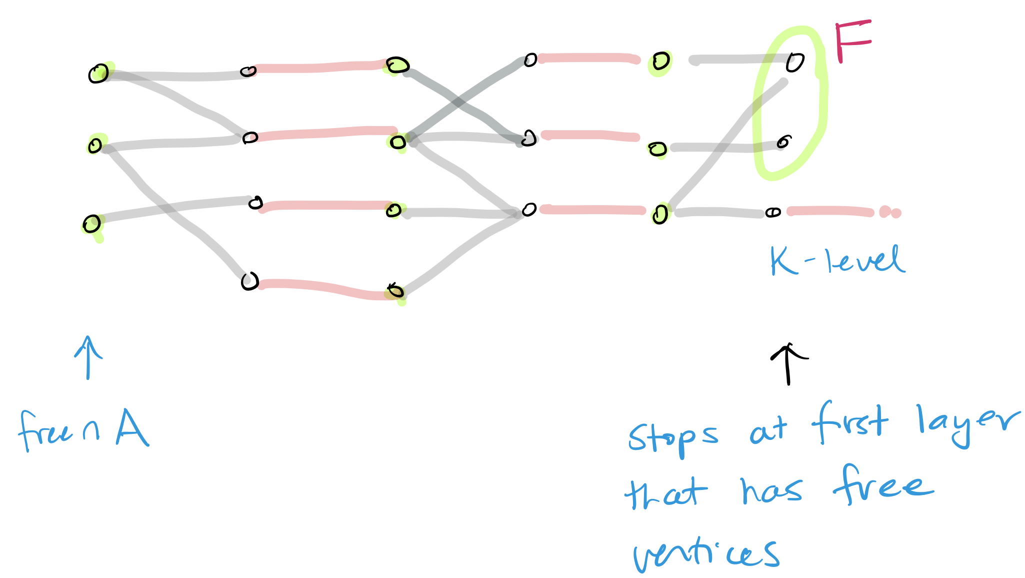
until $\mathcal{P} = \emptyset$

How do we do this efficiently and achieve a $O(\sqrt{n}\, m)$ running time? (Recall Network Flow alg was $O(n^2 m)$)

1. Find the bipartitions A and B – how fast can you do this?

2. Run a "special BFS" where the start is all free vertices in A.

"special BFS" = use non-M edges from A to B
use M-edges from B to A.




Let F be the set of free verts at level K, first level at which free verts are found.

3. Do "special DFS" from F to free ∩ A
↖ alternating
tracing a path back from verts in F to free ∩ A one at a time.

- The DFS should go level to level

- if it gets to free ∩ A, remove the path from the levels so subsequent paths don't reuse those verts.

Repeat from F until ∄ any F-to free ∩ A paths.

At this point, we may want to claim ∄ any M-augmenting paths of length K left in G

4. Return to $\boxed{2}$ to find a set of M-augmenting paths of length $K' > K$. If no such level $K'$ exists that has free verts, halt.

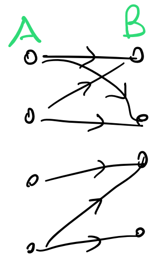
This Alg is clearly "correct" (finds maximum matching) if we can show 2 things:

1. A matching M is maximum $\iff$ ∄ a M-augmenting path in G.

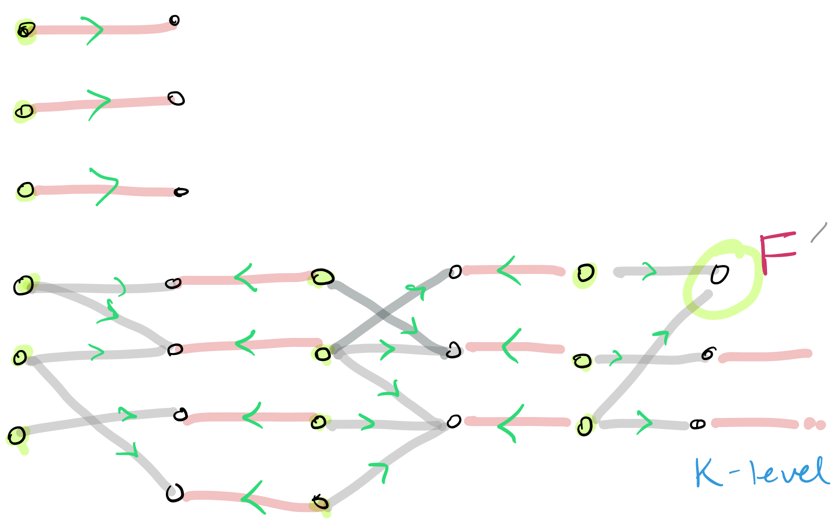
2. Claim: After Step 3 for K level.

$\nexists$ M-augmenting paths of length K

Proof: uses the bipartite nature of the graph.




Direct all edges from A to B.

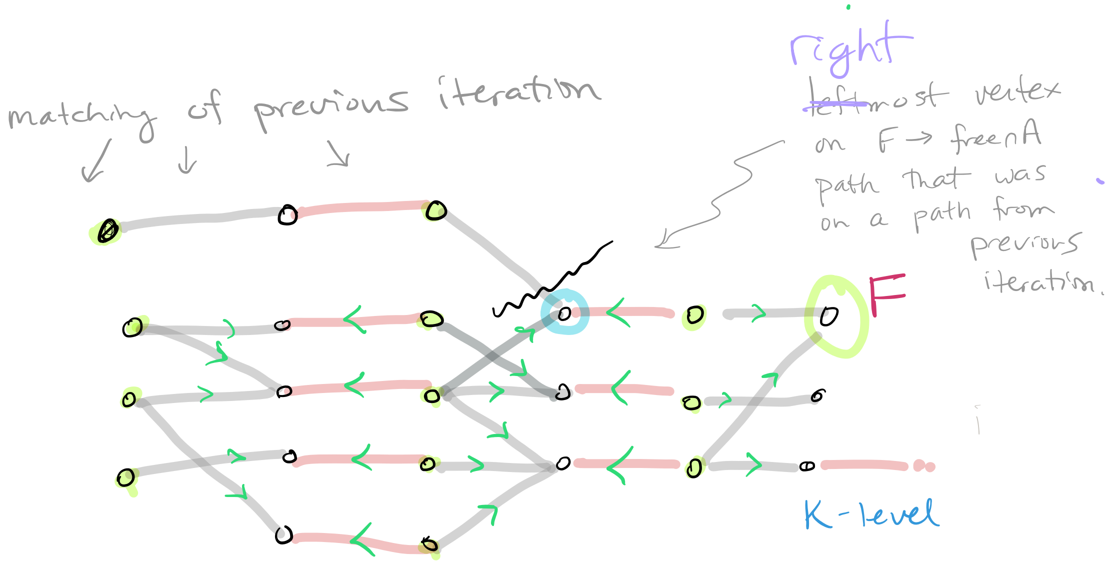



If in the next iteration, you find a free vertex in level K – AGAIN! – Then $\exists$ at least one M-aug path of length K – again!

BUT – if it was There in previous iteration, your DFS from F should have found it.

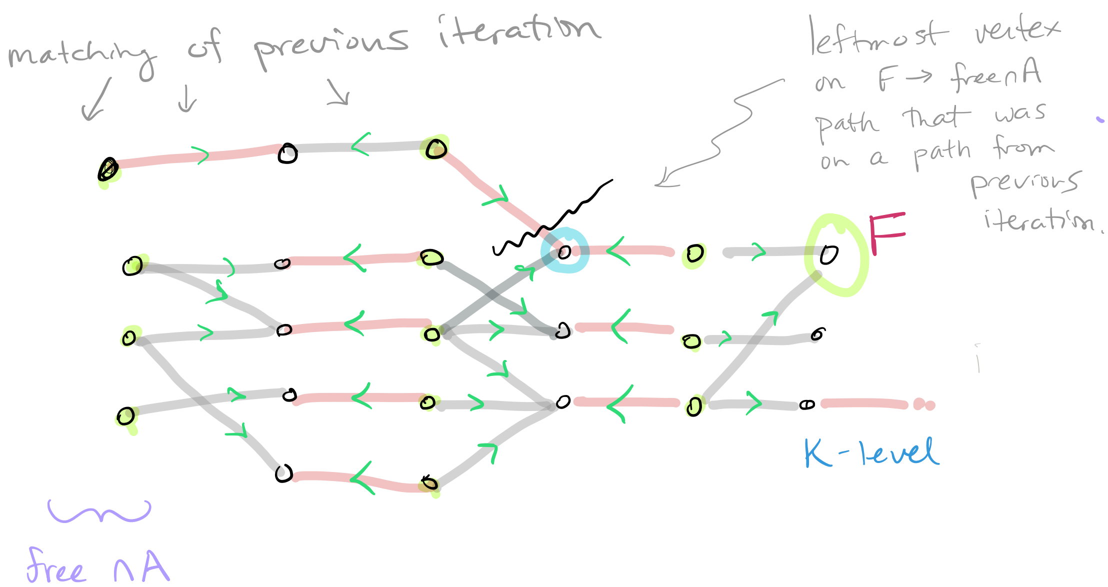
SO – it wasn't There in previous it because the matching edges changed in last iteration.

BUT – that operation "flipped" The direction of the matching edges.




But... the last iteration "flipped" the colour of those edges of the previous iteration

Hence, before the flip, they looked like this:




But then the path from last iteration does not start with a free vertex $\Rightarrow\Leftarrow$.

**Lemma:** Hopcroft-Karp-Kazarov uses $\leq 2\sqrt{n}$ iterations of the main loop to find a maximum matching.

**Proof:** After $\sqrt{n}$ iterations, the M-augmenting paths must be of length $\geq \sqrt{n}$.

How many such vertex-disjoint paths (of length $\geq \sqrt{n}$) can exist in $G$?

$\leq \sqrt{n}$ because: disjoint; and

$\sqrt{n} \cdot \sqrt{n} = n$ $\boxtimes$

Theorem: If $M$ is a matching that is not maximum, then $M$ has a $M$-augmenting path.

Proof: Let $M^*$ be a larger matching.

1. if $M^*$ has any edge $e$ that is not incident (touching) any edges of $M$, then $e$ is an $M$-augmenting path.

2. Otherwise iterate the following step:

Iterate: find a vertex that is $M^*$-saturated but not $M$-saturated.

Grow the $M^*$-$M$-alternating path from that vertex until it can grow no more. If it ends in an $M$-unsaturated vertex — that is an $M$-alternating path. If not find another $M^*$-saturated, $M$-unsaturated vertex and continue.

The process must continue until it finds an $M$-alternating path, since the only way to end early is to run out of $M$-unsaturated $M^*$-saturated vertices. But in that case, the number of $M$-edges $=$ # $M^*$ edges, contradicting that $M^*$ is a bigger matching. $\blacksquare$