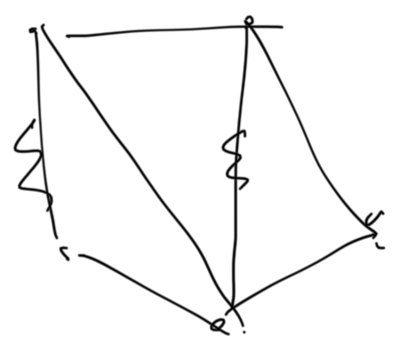
Lecture 6 (Graph theory)

[illegible]

Matching in a graph.

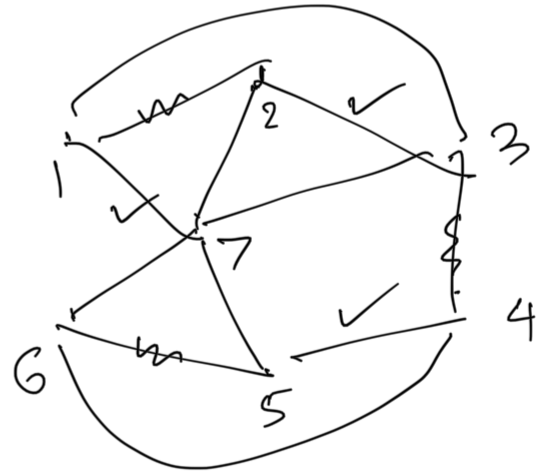
Defn: Given a graph G, a matching $M \subseteq G$ is a subgraph whose every vertex has degree one (in M). [No two edges in M are adjacent]

G:



$\wr$: Matching (maximum)

A matching M is called maximum if its cardinality is maximum among all matchings in G.



$\S$: Maximum matching

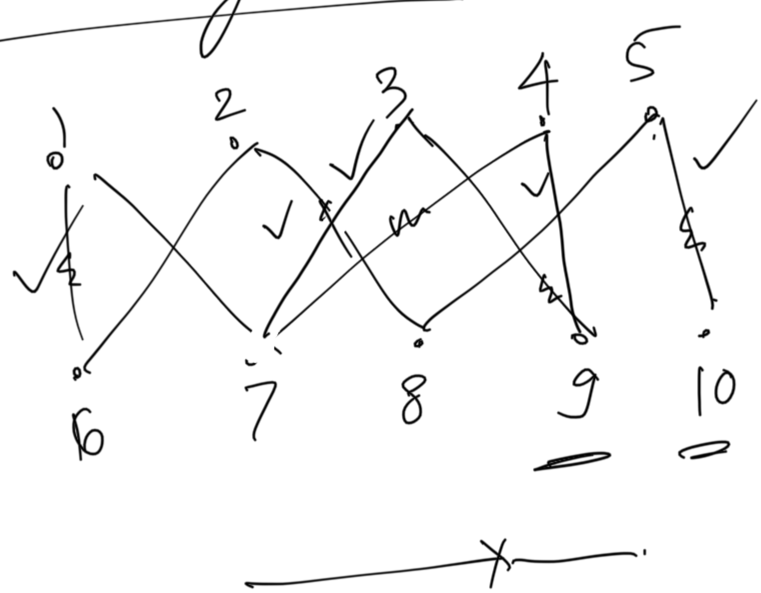
$\checkmark$: Maximum matching

A maximum matching M is called **perfect** if $|G|$ is even

& $|M| = \frac{|G|}{2}$. [maximum possible value]

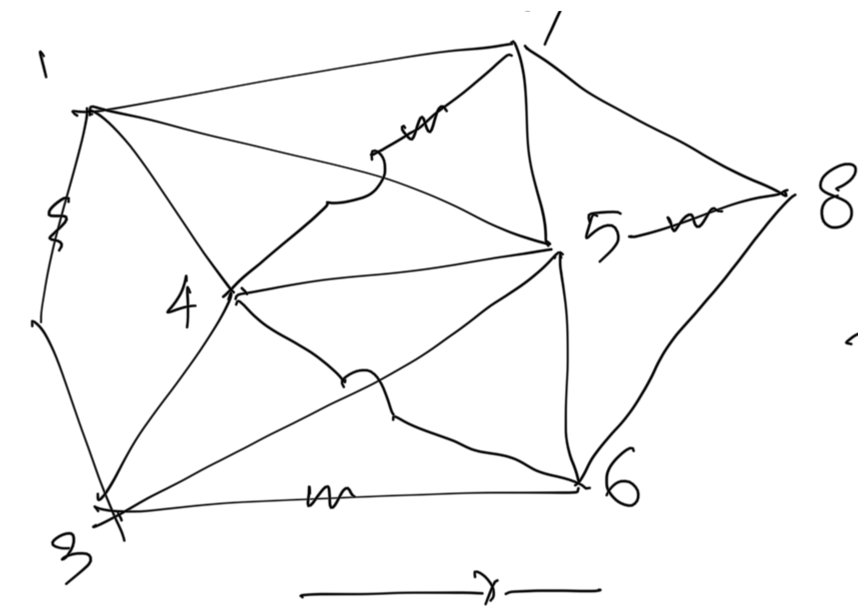
(All vertices of G are matched in M.)

Bipartite graph:



perfect matching?

G:



$\sim$: A perfect matching.

Question: Given a graph G, with $|G|$ even, how fast can we decide if G has a perfect matching? [Perfect matching problem].

Naive algorithm: $G = (V, E)$, $|G|$ even.

Enumerate all possible subsets of E of cardinality $|G|/2$. $\#: \binom{E}{|G|/2}$ exponential in $n = |G|/2$.

For each enumerated subset $M \subseteq E$, decide if M is a matching.

if # edges is m & # vertices is n, then the # possible subsets of size $n/2$ $\binom{m}{n/2} \simeq \left(\frac{m}{n/2}\right)^{n/2}$ exponential in n.

pairs of E of $\binom{n}{n/2}$

Naive algorithm takes exponential (in n) time.

Can we do better?
Can this problem be solved in poly(n, m) time?
$m \leq n^2 \Rightarrow$ poly(n).

<u>Yes</u>: [Beginning of complexity theory].

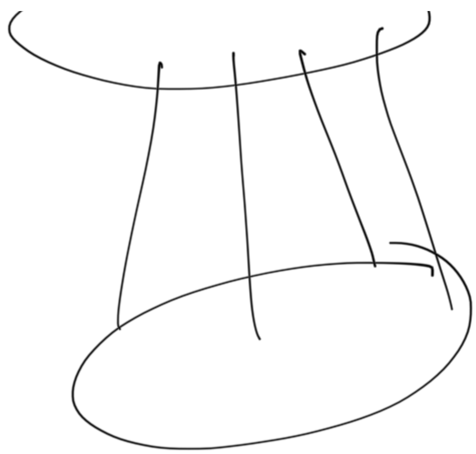
<u>Bipartite case</u> [Easier]:

Thm [Hungarian method] [König, et al.].

Given a bipartite graph $G = (V, E)$, $V = A \cup B$, with $|G|$ even & $|A| = |B|$,

<u>In the course</u>.

whether G has a



B

$n = |V|$
$= |G|$

perfect matching can be decided in polynomial time,

($O(n^4)$) — König et al.

$\downarrow$

$O(n^3)$ — Edmonds — Karp

$\downarrow$

edges

$O(m n^{1/2})$ — Micali, Vazirani.

vertices.

?

Non-bipartite case

Thm [Edmonds] — A fundamental result.

Given a non-bipartite graph G, one can decide whether G has a

with $\underset{\overset{\shortparallel}{n}}{|G|}$ even, whether G …
perfect matching can be decided in polynomial $\left(\mathrm{poly}(n)\right)$ time.

$$\mathrm{poly}(n) = O(mn^2) \text{ time — Edmonds}$$

$$O(mn^{1/2}) \text{ — Micali \& Vazirani.}$$

This result led Edmonds to introduce the complexity class P:

P: The complexity class of problems that can be solved in polynomial $\left(\mathrm{poly}(N)\right)$ time.

N: total bitlength of the input.

Thm [Edmonds]: [Rephrased]

Not in this course. The perfect matching problem for general graphs is in P.

(This result was the beginning of the theory of P [Complexity theory]

NP

$P \neq NP$ Conjecture.

Basic theory of Matching in (bipartite) graphs.

$G = (V, E)$: a graph.

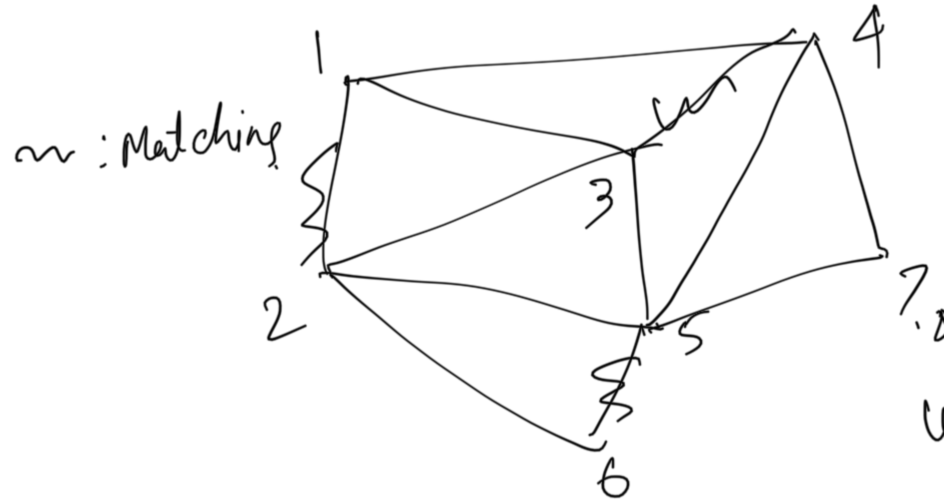
M: Matching (an independent set of edges in E)

independent = non-adjacent.

∴ vertex in M has indegree 1.

U = The set of endpoints of the edges in M.

We say that the vertices in U are <u>matched</u>.



$U = \{1, 2, 3, 4, 5, 6\}$ matched.

7 : unmatched.

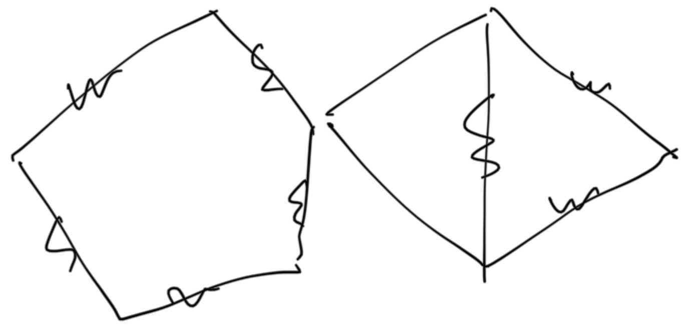
K-factor [a generalization of a perfect matching].

$K \geq 1$: integer.

A <u>K-factor</u> in G is a K-regular spanning subgraph of $G = (V, E)$.

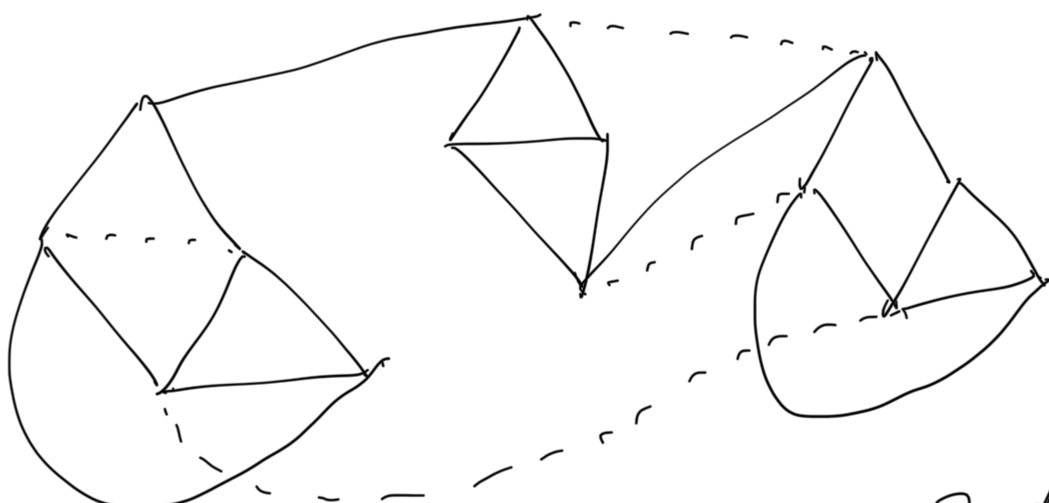
<u>1-factor</u> := perfect matching.

2-factor := union of disjoint cycle which spans G.



~ : 2-factor.

3-factor: spanning cubic graph.



— : 3-factor.

Given a graph G, how fast can one decide if G has a 3-factor?

polynomial?

(inherently) exponential? ← P ≠ NP

NP-Complete

theory of ...

algorithm

Matching in Bipartite graphs

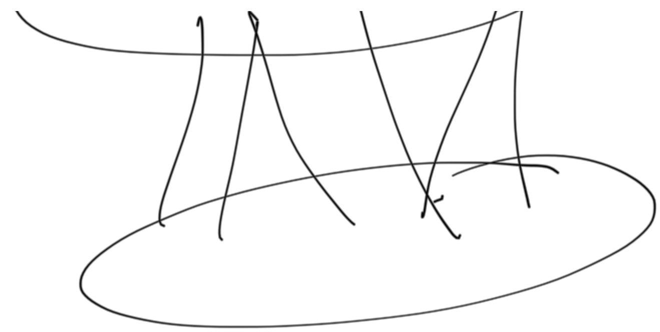
Goal: Develop the theory, matchings for ~~in~~ bipartite graphs culminating in a polynomial time algorithm for finding a perfect matching in a bipartite graph (if one exists).

Begin --

Let $G = (V, E)$ be a bipartite graph.

$$V = A \cup B$$

disjoint classes.

$G:$  B

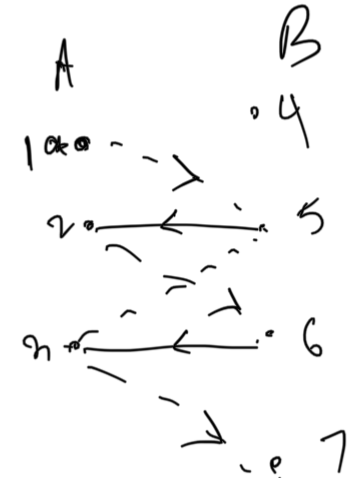
Let $M \subseteq G$ be a matching (possibly partial)

(not perfect.)

<u>Defn</u>: An <u>alternating path</u> P in G w.r.t. M is a path in G which starts at an <u>unmatched</u> vertex in A & contains alternating edges in $E \setminus M$ & M.

[Can be trivial] (length $= 0$).

<u>Example</u>:



— : Matching M

⟶ : one alternating path in G w.r.t. M

$P: (1, 5, 2, 6, 3, 7)$

Augmenting?

Yes.

Defn: An <u>augmenting path</u> $P \subseteq G$

w.r.t. M is an alternating path which ends at an unmatched vertex in B.

Importance of augmenting path:

<u>Prop:</u> Let $M \subseteq G$ (bipartite) be a matching

$\overset{\shortparallel}{(V,E)}$

$\overset{\shortparallel}{A \dot{\cup} B}$

& $\underset{\substack{\subseteq G \\ \subseteq E}}{P}$ an augmenting path in G w.r.t. M.

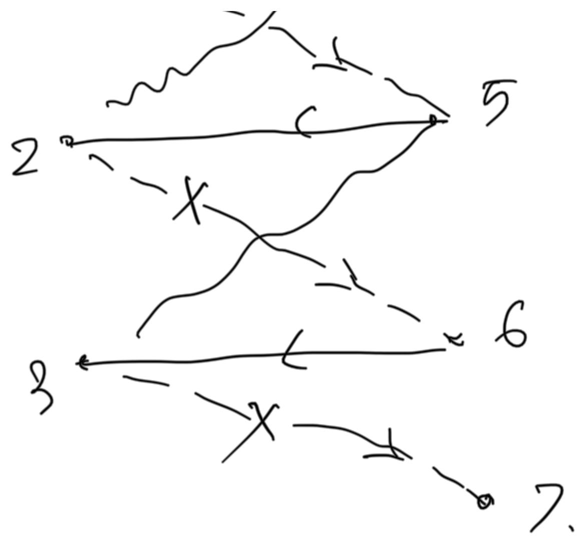
Then $M' := \underbrace{M \oplus E(P)}_{\text{Contains all edges in } M \text{ not in } P \text{ \& all edges in } P \text{ not in } M.} = M \setminus P \cup P \setminus M.$

is a matching

with $|M'| = |M| + 1$

<u>Size of the matchings increases.</u>

$1 \cdots 4$ $\quad$ —: M



$\longrightarrow$: P

$\sim$: remaining edges.

M' : X

$M = \{(2,5), (3,6)\}$

$P = \{(1,5), (2,5), (2,6), (3,6), (3,7)\}$

$M' := M \oplus P = \{(1,5), (2,6), (3,7)\}$

An idea for constructing a maximum matching M in a bipartite graph G in poly time.

Start with $M = \{\}$

main problem Do this in polynomial time.

At every time find an augmenting path P in G wrt M.

w.r.t. ...

Body: Replace M by $\underbrace{M \oplus E(P)}_{\text{layer}}$.

\# iterations = \# edges in a maximum matching $\leq \frac{|G|}{2}$.

Keep doing this, until there is no augmenting path in G w.r.t. M.

One can show that this algorithm eventually ends with a maximum matching M in G.