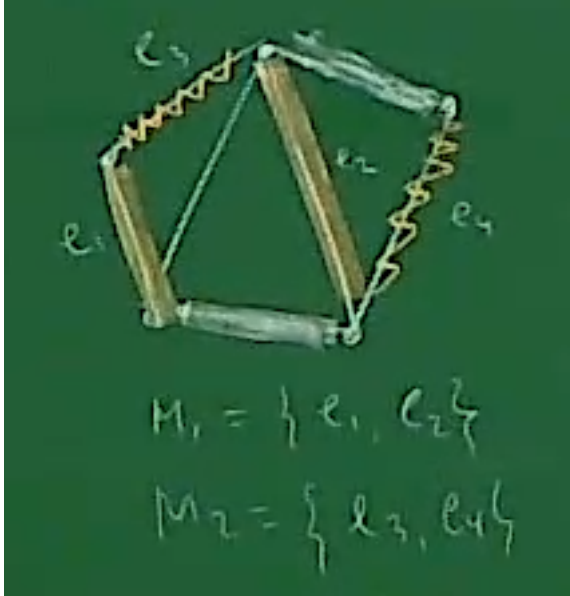


Matching

- Matching in G ist Menge $M \subseteq E$, sodass jeder Knoten höchstens eine Kante von M berührt

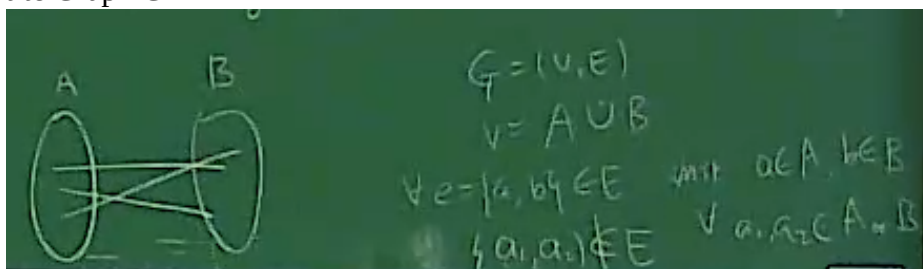


- falls M ein Matching in G ist & $\{x, y\} = [x, y] \in M$, dann
 x ist mit y gematcht (umgekehrt auch)

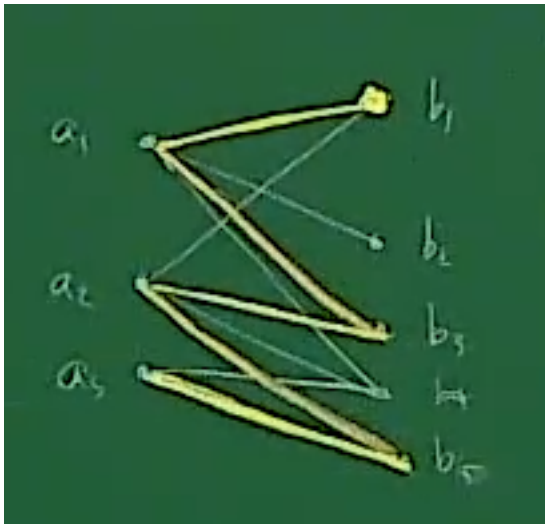
- Matching ist perfekt, wenn jeder Knoten gematcht ist
 - $|M| = \frac{|V|}{2}$
 - größtmögliches Matching

Satz von Hall

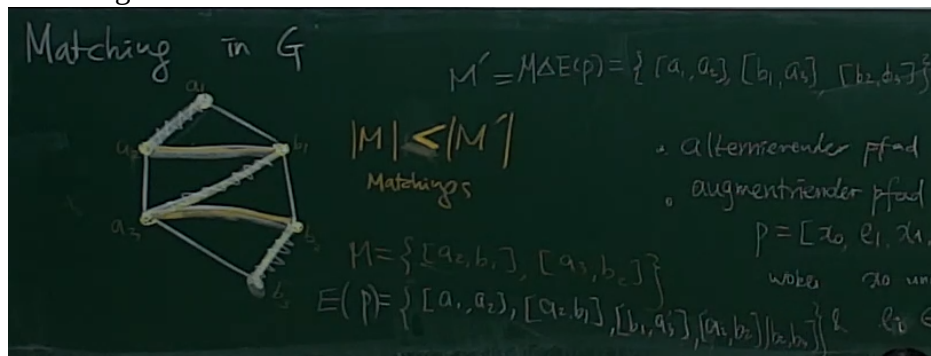
- bipartite Graph G



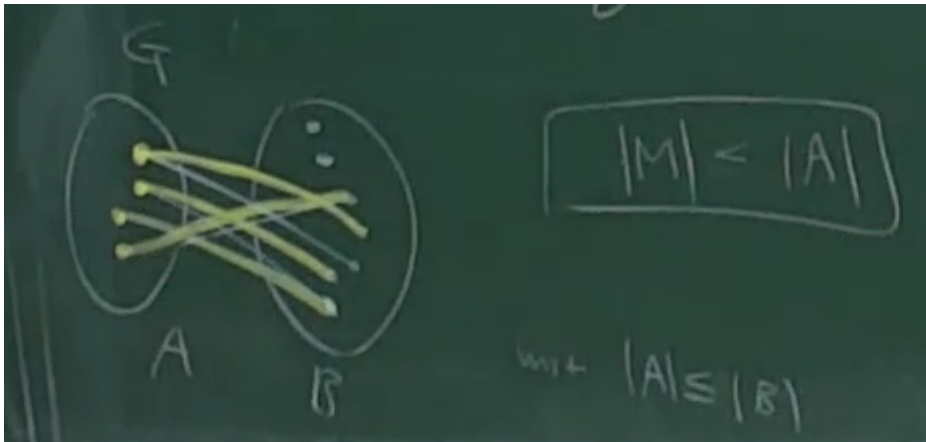
- G hat Matching, wenn $N(S) \geq |S|$ für $\forall S \subseteq A$



- Pfad $P = (x_0, e_1, x_1, e_2, x_2, \dots, e_n, x_n)$ alternierend bzgl. M , wenn
 - x_0 ungematcht:
 - * $e_i \in M$ für gerade i
 - * $e_i \notin M$ für ungerade i
- P ist augmentierend bzgl. M , wenn
 - alternierend
 - letzte Knoten x_n ungematcht
 - $M' = M \triangle E(P) = (M \setminus E(P)) \cup (E(P) \setminus M)$ symmetrische Differenz
 - * Matching mit einer Kante mehr als M



- falls M nicht größtmöglich \implies augmentierender Pfad existiert
 - nicht größtmöglich solange $|M| < |A|$ mit $|A| \leq |B|$



Bipartite Matching

- Input: $G = (A \cup B, E)$ bipartite Graph mit $|A| \leq |B|$
- Output: größtmögliche Matching in G
- erstelle M (aus augmentierter Pfad P) mit Kante mehr bis größtmöglich
- Verfahren:
 - sei $M = \emptyset$
 - wiederhole bis $|M| = |A|$
 - * $P = \text{Augmenting-Path}(G, M)$
 - * $M = M \triangle E(P)$
 - return M

Augmenting-Path

- Input: $G = (A \cup B, E)$ bipartite Graph, Matching M
- Output: augmentierender Pfad P bzgl. M
- Verfahren:

– Skriptum

Vorgehensweise: Wir beginnen mit $M = \emptyset$ und vergrößern M rekursiv durch verbessernde Pfade, bis M größtmöglich ist.

Hierfür verwenden wir eine Variante von Algorithmus 3.5 (BFS) mit den folgenden zwei Unterschieden:

- (1) Die gereihte Liste der abzuarbeitenden Knoten besteht am Anfang nicht nur aus einem Knoten, sondern aus allen ungematchten Knoten in A (in beliebiger Reihenfolge).

*

(2) Liegt der aktuell erste Knoten x der Liste in B , dann

- beenden wir den Algorithmus, falls x ungematcht ist;
- ansonsten fügen wir den Knoten y , mit dem x gematcht ist, an das Ende der Liste hinzu, setzen $v(y) = x$ und entfernen x aus der Liste.

*

Solange M noch nicht größtmöglich ist, findet dieser Algorithmus einen ungematchten Knoten $x \in B$. Zusammen mit allen seinen Vorgängern (und den Kanten zwischen ihnen) bildet x einen verbessernden Pfad P .

Wir ersetzen M durch $M \Delta E(P)$ und wiederholen die obige BFS-Variante.

Der Algorithmus endet, sobald die BFS-Variante keinen ungematchten Knoten in B findet. Dann ist M größtmöglich.

*

– Tafel:

Verfahren: (i) füge die Quelle s und die Senke t hinzu
 (ii) füge gerichtete Kante
 – von s nach ungematchten Knoten $a \in A$
 – von ungematchten Knoten $b \in B$ nach t

*

(iii) Orientiere
 – jede ungematchte Kante von A nach B
 – jede gematchte Kante von B nach A
 $H =$ Der resultierende gerichtete Graph
 (iv) Verwende $BFS(H)$ mit s als Wurzel
 Sei $T = BFS(H)$
 (v) Verwende $Path\text{-}finder(T, t)$ um einen kürzesten Pfad von s nach t zu finden
 Sei $P = Path\text{-}finder(T)$
 (vi) Return: $P \leftarrow P \cup s, t$

*

[[Graphentheorie]]