Algorithmen und Wahrscheinlichkeit

Woche 3

Ilya Maier

Bemerkungen

- Wenn man Latex ausprobieren möchte:
  - overleaf.com
  - Texifier
  - es gibt ein Template und eine Beispiel Datei auf meiner Webseite

- Pigeonhole principle, bipartiter Graph, k-regulärer Graph

Announcements

- nächste ÜS online -> Link auf meiner Webseite

CPC Meetups Kickoff

19:00 11.03.2024 CAB H56

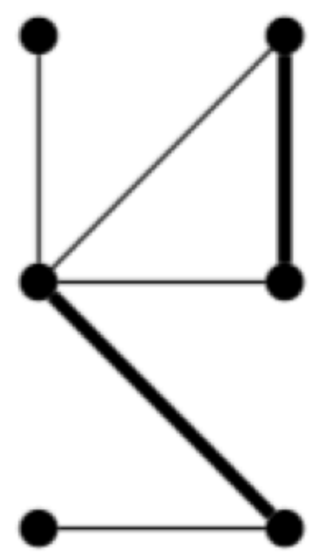
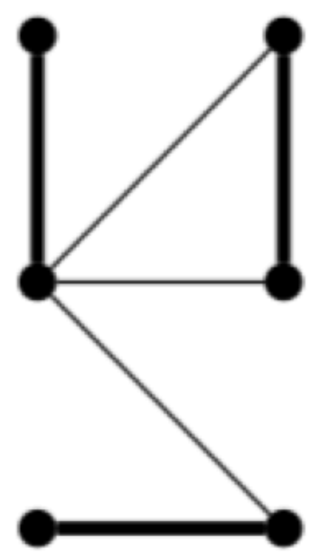
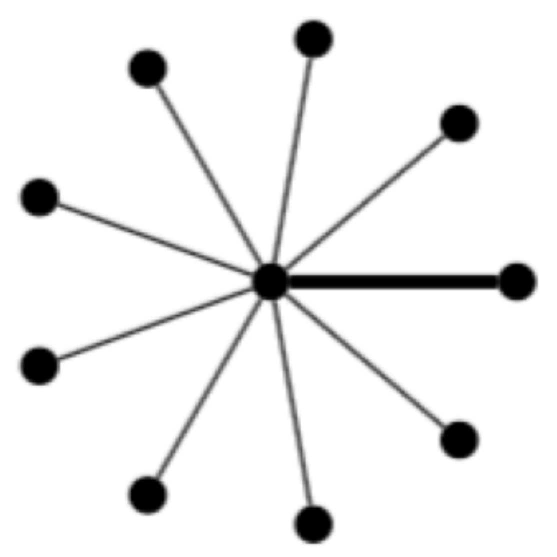
Open to all skill levels!

vseth vis

Matchings - Definitionen

Matching: Eine Kantenmenge $M \subseteq E$ in einem Graphen $G = (V, E)$,
wobei jeder Knote in V zu höchstens einer Kante in M inzident ist (inzident zu 0 oder 1 Kante)

Überdeckt: Ein Knote v ist überdeckt von Matching M, falls v zu einer Kante in M inzident ist

Perfektes Matching: Ein Matching M, sodass alle v überdeckt sind

Inklusionsmaximales Matching*(maximal)***:** Ein Matching M, sodass $\forall e \in E : M \cup e$ ist kein Matching

Kardinalitätsmaximales Matching*(maximum)***:** Ein Matching M, sodass $\neg \exists M' \subseteq E : |M'| > |M|$

$\Longrightarrow$ Ein KM ist ein IM

Matchings - Sätze

Satz: Für ein IM M_{ink} und ein KM M_{kar} gilt: $|M_{ink}| \geq \frac{|M_{kar}|}{2}$

Beweis: Für jede Kante e in M_{kar} muss M_{ink} mindestens ein Endpunkt von e bedecken, sonst $M_{ink} \cup e$ ist ein Matching!! (Widerspruch)

$\Longrightarrow |M_{kar}| \leq$ #Endpunkte in $M_{ink} = 2|M_{ink}|$

Satz von Hall (Heiratssatz)

Ein <u>bipartiter</u> Graph $G = (A \uplus B, E)$ hat ein Matching M der Kardinalität $|M| = |A|$ **gdw** $\forall X \subseteq A : |X| \leq |\mathcal{N}(X)|$

Korollar (Frobenius)

$\forall k$: jeder k**-reguläre bipartite Graph** hat ein **perfektes Matching**

<u>Beweis</u>: $\forall X \subseteq A : k|X| =$ #Kanten von X nach $\mathcal{N}(X) \leq k|\mathcal{N}(X)|$
$\Longrightarrow$: $\forall X \subseteq A : |X| \leq |\mathcal{N}(X)|$

Matchings - Matching Algorithmen

Greedy
Wähle zufällig eine Kante und lösche sie und die inzidenten Kanten bis $|E| = \varnothing$
$\Longrightarrow$ Findet **ein inklusionsmaximales Matching** in $O(|E|)$

Gabows Algorithmus

In 2^k**-regulären bipartiten Graphen** kann man in Zeit $O(|E|)$ ein **perfektes Matching** bestimmen

→ 1. Finde eine Eulertour   2. Entferne jede zweite Kante →2^{k-1}-regulärer bipartiter Graph   3. Iteriere

Cole, Ost, Schirras Algorithmus

In k**-regulären bipartiten Graphen** kann man in Zeit $O(|E|)$ ein **perfektes Matching** bestimmen

Hopcroft-Karp

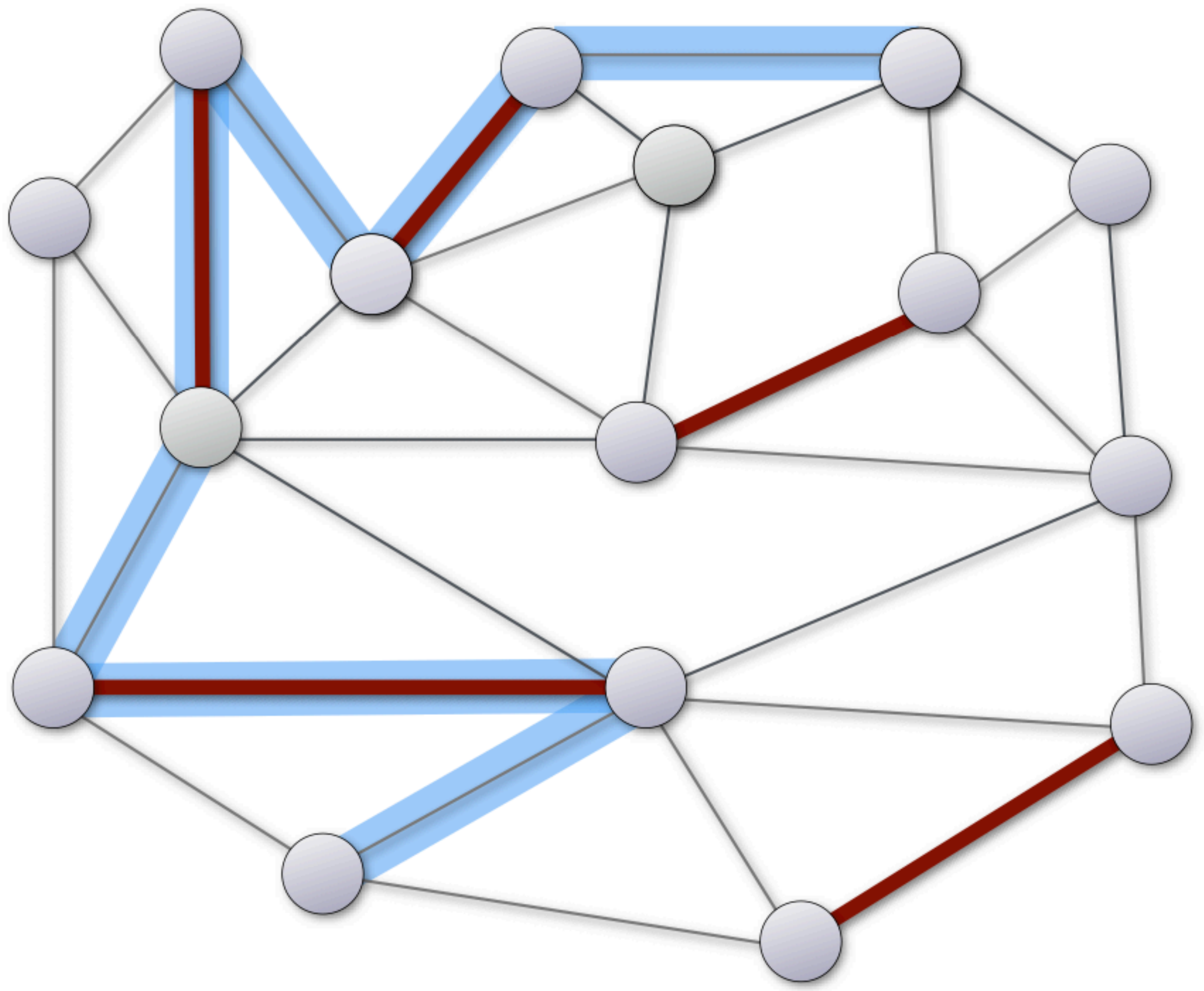
In **bipartiten Graphen** kann man in Zeit $O(\sqrt{|V|} \cdot |E|)$ ein **maximales Matching** bestimmen

Matchings - Augmentierende Pfade

M**-augmentierender Pfad**:

1) abwechselnd Kanten aus nicht M und M
2) beginnt und endet in einem von M unüberdeckten Knoten

Das Vergrößern von M mit einem M-augmentierenden Pfad P: $M' = M \oplus P$



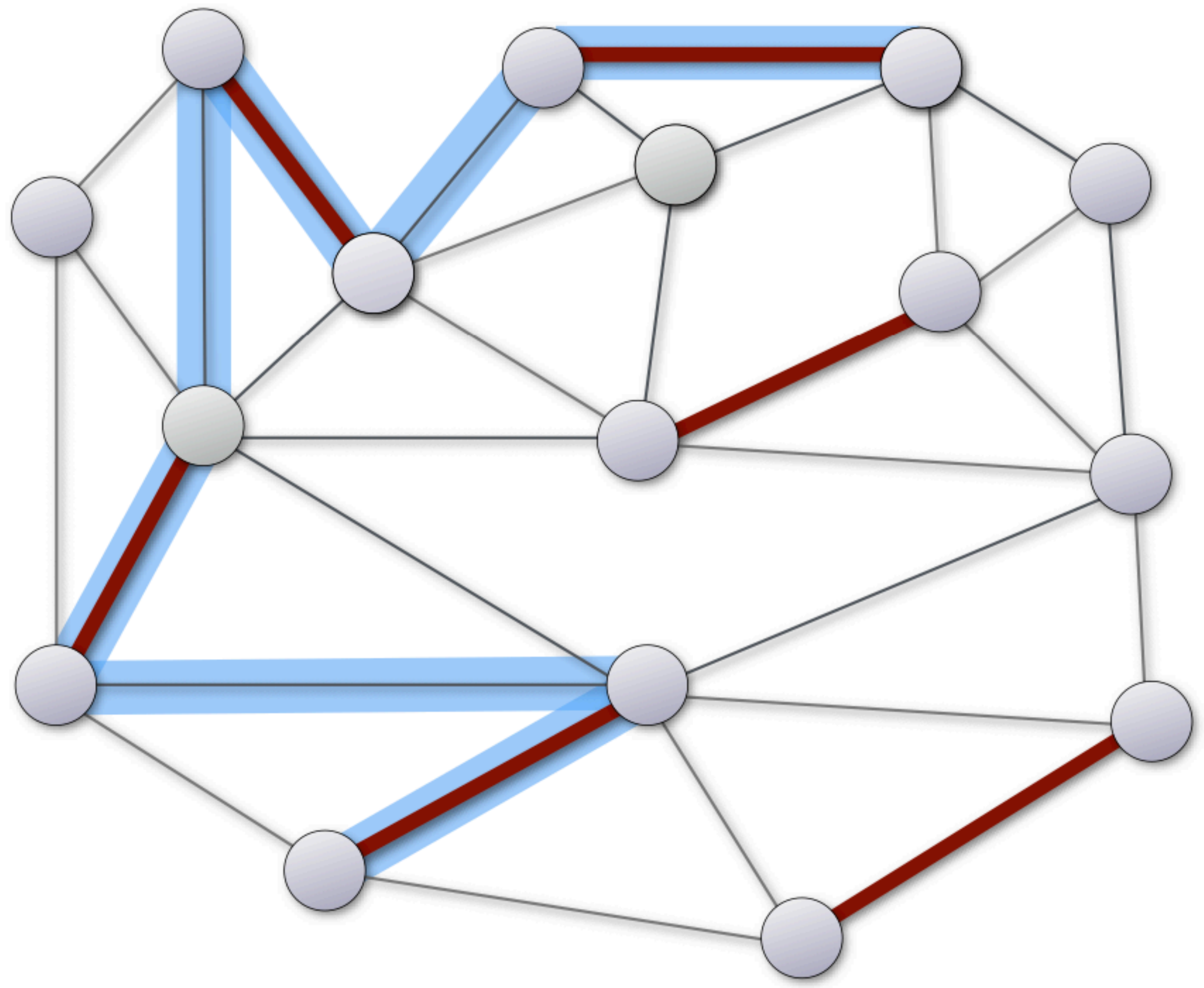
Matchings - Augmentierende Pfade

M**-augmentierender Pfad**:

1) abwechselnd Kanten aus nicht M und M
2) beginnt und endet in einem von M unüberdeckten Knoten

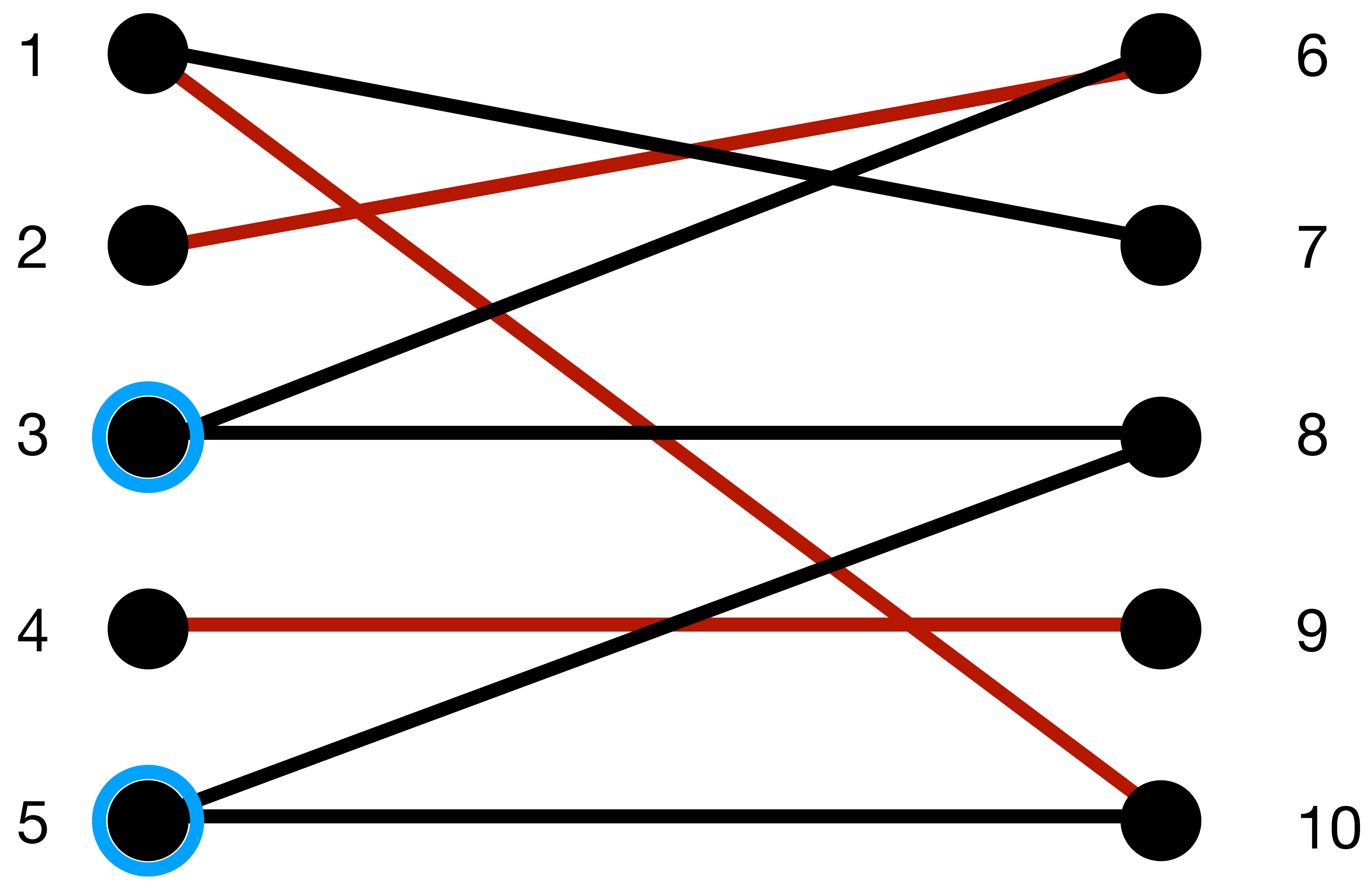
Das Vergrößern von M mit einem M-augmentierenden Pfad P: $M' = M \oplus P$

Satz von Berge:
Jedes Matching, das nicht kardinalitätsmaximal ist,
besitzt einen augmentierenden Pfad

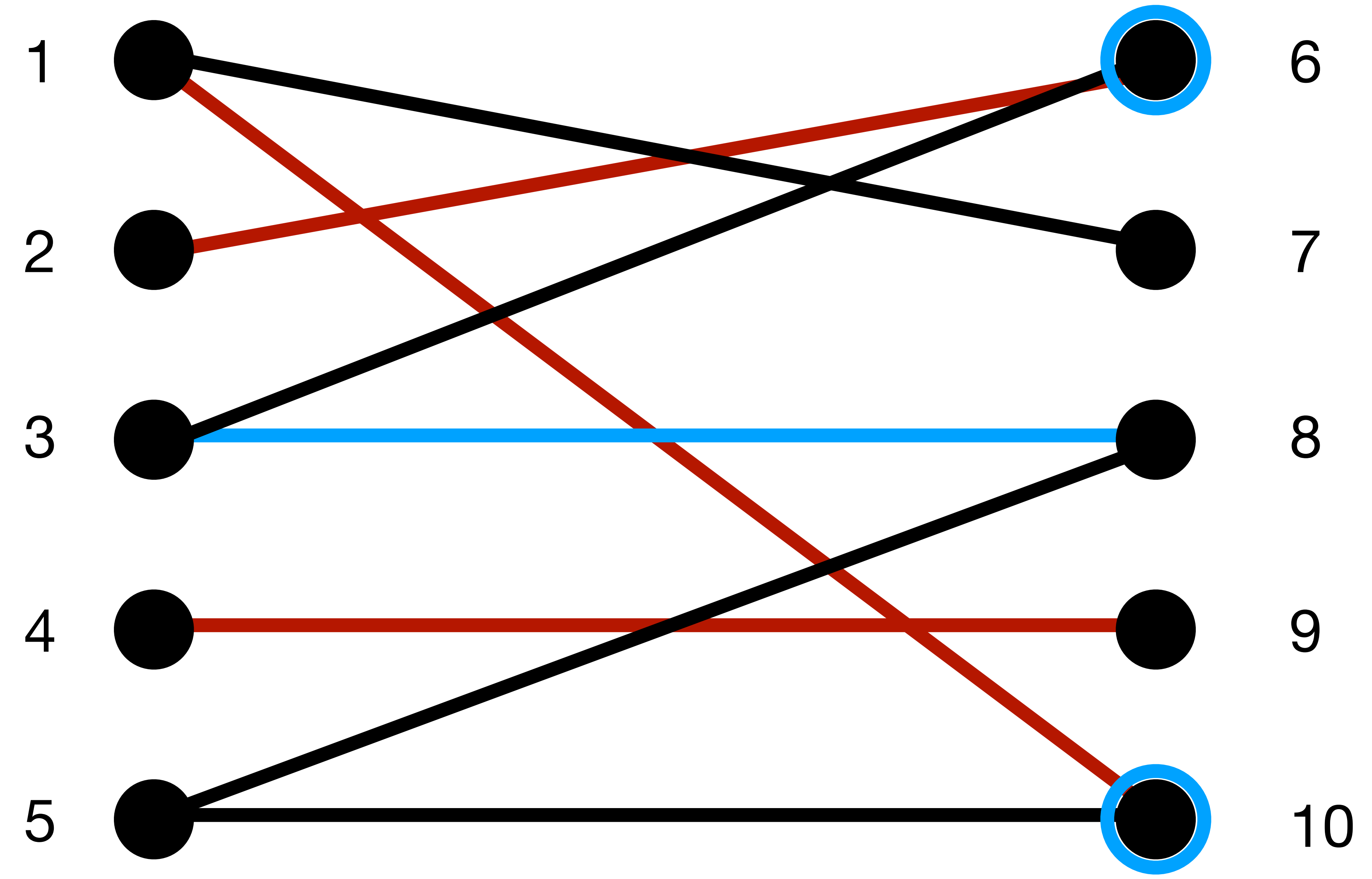


Für zwei beliebige Matchings M und M' wobei $|M| < |M'|$:
$M \oplus M'$ hat **mindestens $|M'| - |M|$ knoten-disjunkt M-augmentierende Pfade**

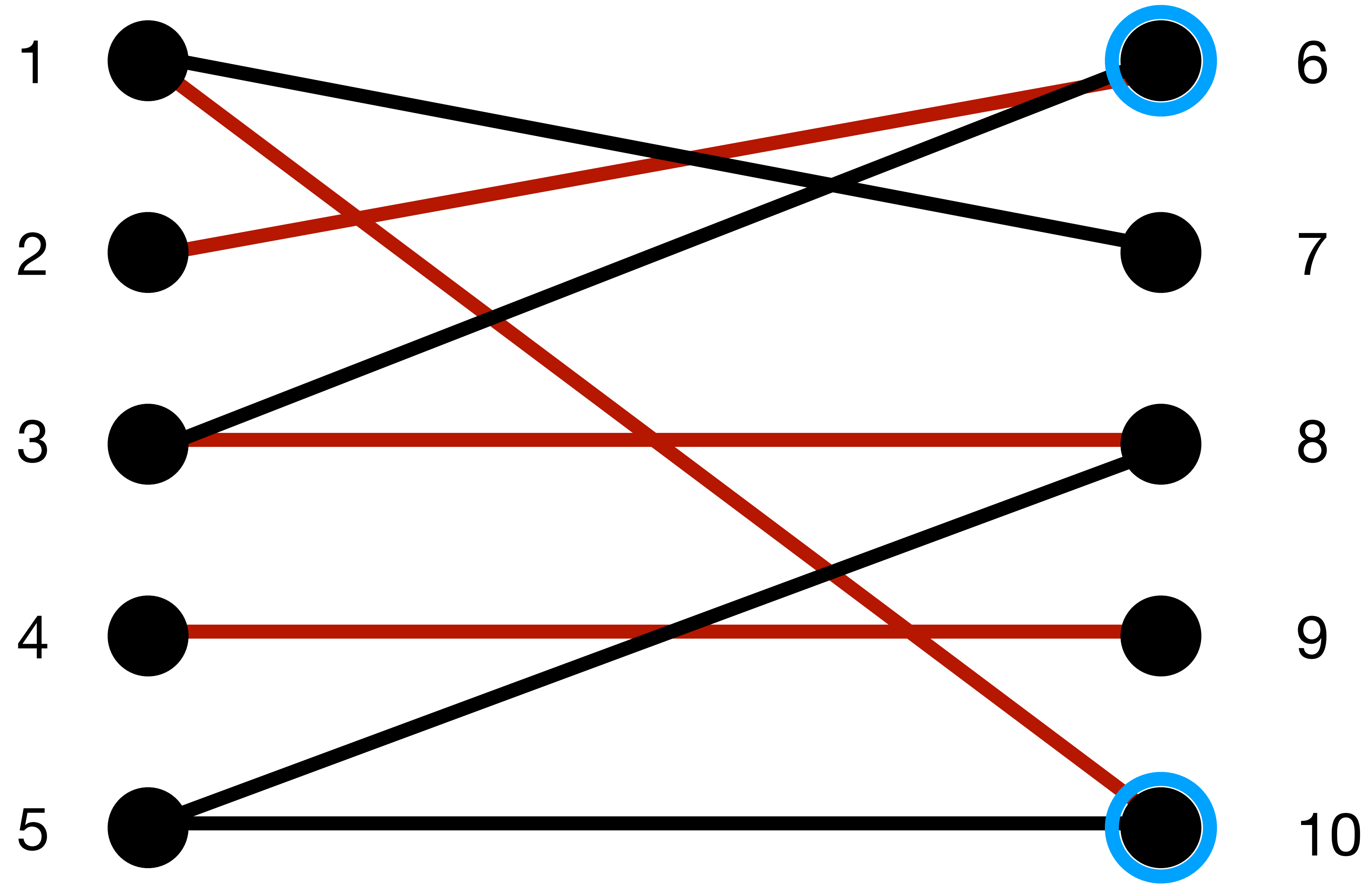
Matchings - Augmentierende Pfade



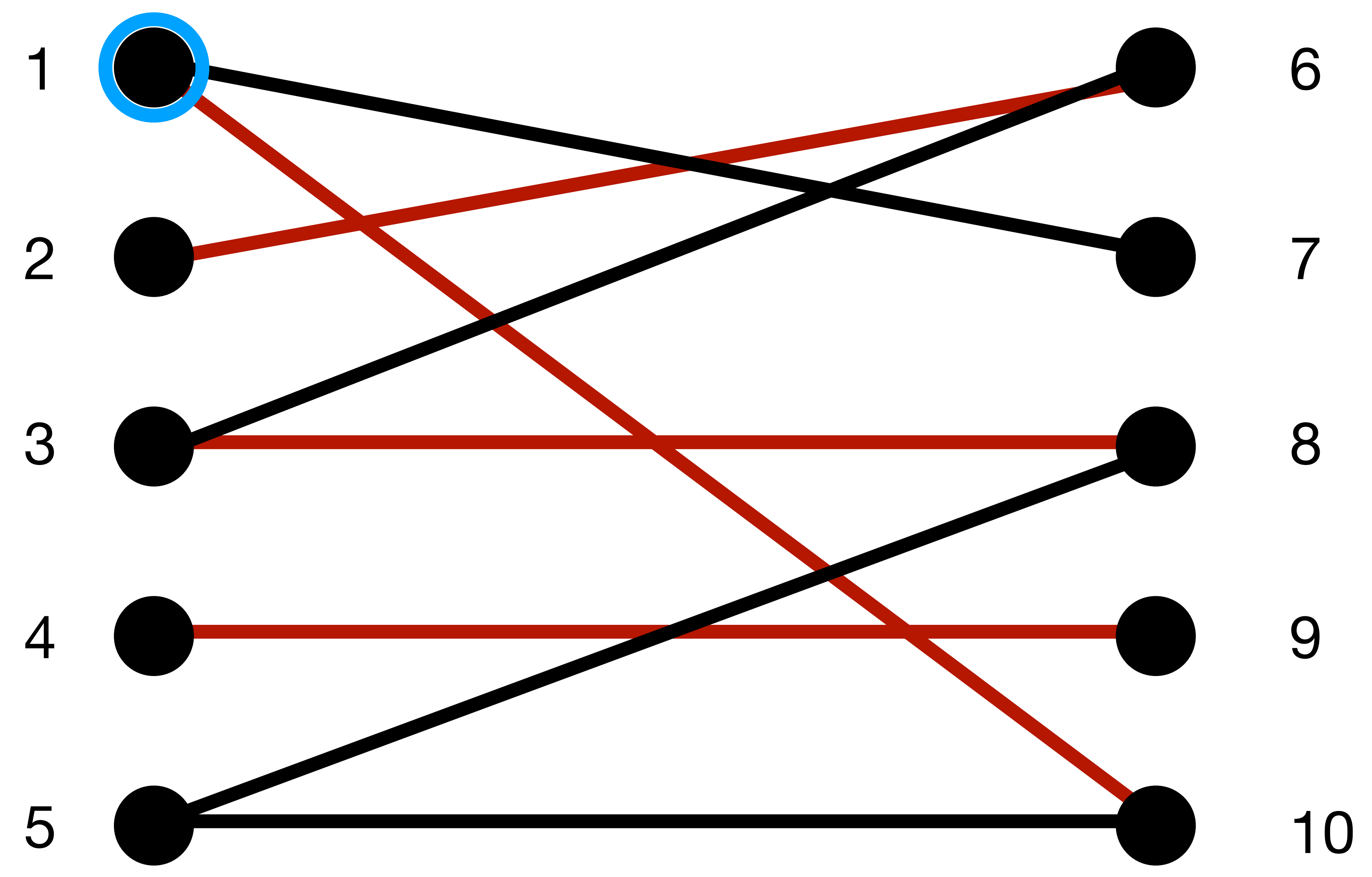
Matchings - Augmentierende Pfade



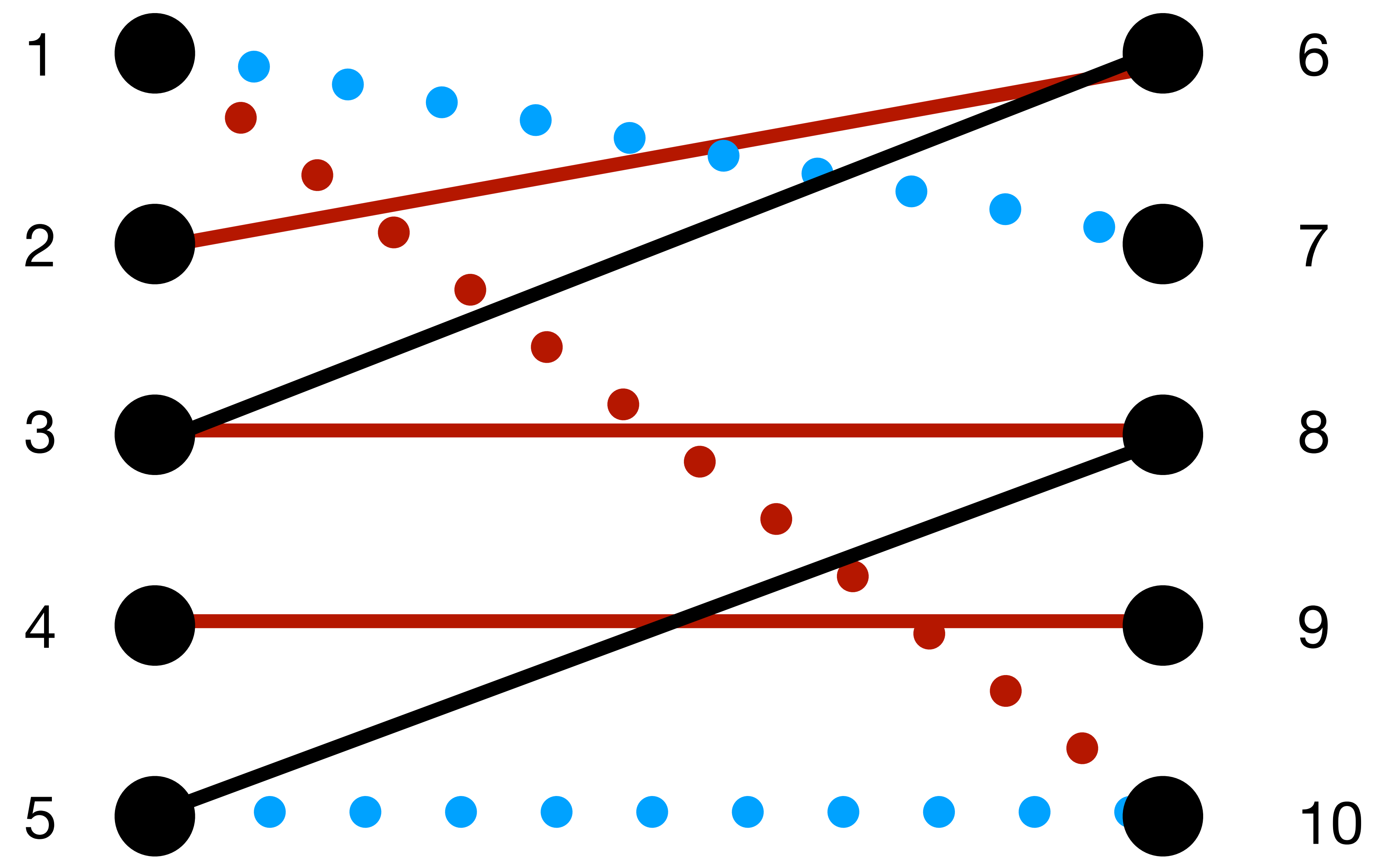
Matchings - Augmentierende Pfade



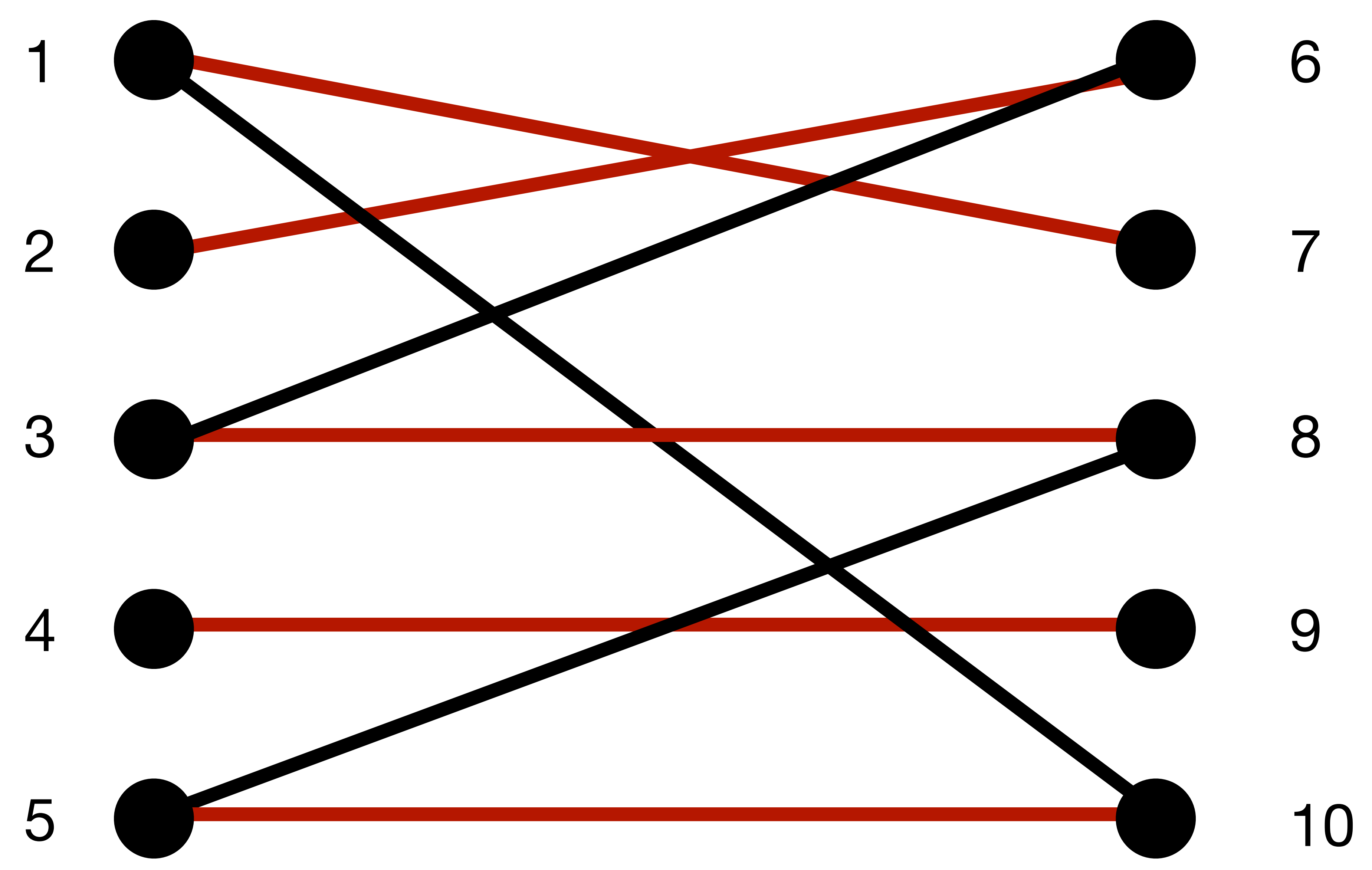
Matchings - Augmentierende Pfade



Matchings - Augmentierende Pfade



Matchings - Augmentierende Pfade



Matchings - Augmentierende Pfade

Algorithmus fürs Finden eines augmentierenden Pfades (bipartit)

Eingabe: ein bipartiter Graph $G = (A \uplus B, E)$, ein Matching M

Ausgabe: kürzester augmentierender Pfad, falls existiert

L_0 = {unüberdeckte Knoten aus A}

Markiere die Knoten in L_0 als *besucht*

for i = 1 … n

  if i ungerade **then**

    $L_i = \{$unbesuchte Nachbarn von L_{i-1} via Kanten in $E \backslash M\}$

  else

    $L_i = \{$unbesuchte Nachbarn von L_{i-1} via Kanten in $M\}$

  Markiere die Knoten in L_i als *besucht*

  if ein Knote v in L_i ist nicht überdeckt $\Longrightarrow$ **return** Pfad zu v

Laufzeit

$O(|E|)$ (BFS)

Algorithmus fürs maximale Matching

Eingabe: $G = (V, E)$

Ausgabe: KM Matching M

Starte mit $M = \emptyset$

  repeat

    Suche augmentierenden Pfad P

    if kein solcher Pfad existiert **then return** M

    else $M = M \oplus P$

Laufzeit

$O(|V| \cdot |E|)$

Hopcroft-Karp

$O(\sqrt{|V|} \cdot |E|)$

Matchings - Christofides Algorithmus

2-Approximation

Eingabe: K_n, metrische Längenfunktion l

Output: Ein Hamiltonkreis, C, sodass $l(C) \leq 2 \cdot \text{opt}(K_n, l)$

1. Finde den **MST** T von G.
2. **Verdopple** alle Kanten in T $\rightarrow$ Denn alle Knoten müssen einen geraden Grad haben für eine Eulertour
3. Bestimmt **Eulertour** W
4. **Kürze** W **ab**, sodass jeder Knoten nur einmal besucht wird $\Longrightarrow$ Hamiltonkreis C

Matchings - Christofides Algorithmus

1.5-Approximation (Christofides)
Eingabe: K_n, metrische Längenfunktion l
Output: Ein Hamiltonkreis, C, sodass $l(C) \leq 1.5 \cdot \text{opt}(K_n, l)$

1. Finde den **MST** T von G.
2. Finde **minimales perfektes Matching** M von $G[U]$ wobei $U := \{v \in T \mid \deg(v) \text{ ungerade}\}$
3. **Füge M zu T hinzu** (Nun haben alle Knoten einen geraden Grad)
4. Bestimmt **Eulertour** W
5. **Kürze W ab**, sodass jeder Knoten nur einmal besucht wird $\Longrightarrow$ Hamiltonkreis C

Analysis

1. Für eine Kante e im Hamiltonkreis H, $H - e$ ist ein Spannbaum. Von daher: $l(T) \leq \text{opt}(K_n, l)$
2. Da H ein Kreis ist, $l(M) \leq \frac{1}{2}\text{opt}(K_n, l)$
3. Eulertour $l(W) = l(T) + l(M) \leq 1.5\text{opt}(K_n, l)$
4. Abkürzen: $l(C) \leq l(W) \leq 1.5\text{opt}(K_n, l)$

Kahoot!

Aufgaben

Aufgabe 1 – *Lateinisches Rechteck*

Ein lateinisches $r \times n$-Rechteck ($r \leq n$) ist eine Anordnung der Zahlen $1, \ldots, n$ in r Zeilen und n Spalten, so dass in jeder Zeile und jeder Spalte jede Zahl höchstens einmal vorkommt. Ein lateinisches n-Quadrat ist ein lateinisches $n \times n$-Rechteck.

$$\begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Beispiel eines lateinischen 3×4-Rechtecks.

Angenommen wir haben ein lateinisches $r \times n$-Rechteck mit $r < n$ gegeben. Wir wollen sehen, ob wir es zu einem lateinischen n-Quadrat erweitern können. Das heisst, wir wollen $n - r$ weitere Zeilen mit Zahlen aus $1, \ldots, n$ zu dem Rechteck hinzufügen, ohne eine Spalte oder eine Zeile in der eine Zahl mehrfach vorkommt zu erhalten.

(a) Angenommen wir haben ein lateinisches $r \times n$-Rechteck wie oben beschrieben. Wir wollen zeigen, wann man dieses zu einem $(r + 1) \times n$-Rechteck erweitern kann. Beschreiben Sie, wie man dieses Problem mit einem bipartiten Graphen $G = (A \uplus B, E)$ modellieren kann und zeigen Sie, dass die Erweiterung genau dann möglich ist, wenn G ein perfektes Matching hat.

Aufgabe 1 – *Lateinisches Rechteck*

Ein lateinisches $r \times n$-Rechteck ($r \leq n$) ist eine Anordnung der Zahlen $1, \ldots, n$ in r Zeilen und n Spalten, so dass in jeder Zeile und jeder Spalte jede Zahl höchstens einmal vorkommt. Ein lateinisches n-Quadrat ist ein lateinisches $n \times n$-Rechteck.

$$\begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Beispiel eines lateinischen 3×4-Rechtecks.

Angenommen wir haben ein lateinisches $r \times n$-Rechteck mit $r < n$ gegeben. Wir wollen sehen, ob wir es zu einem lateinischen n-Quadrat erweitern können. Das heisst, wir wollen $n - r$ weitere Zeilen mit Zahlen aus $1, \ldots, n$ zu dem Rechteck hinzufügen, ohne eine Spalte oder eine Zeile in der eine Zahl mehrfach vorkommt zu erhalten.

(b) Zeigen Sie, dass der in (a) konstruierte Graph regulär ist. Das heisst, zeigen Sie, dass es eine ganze Zahl k gibt, sodass alle Knoten (sowohl die Knoten in A als auch die Knoten in B) Grad genau k haben.

Aufgabe 1 – *Lateinisches Rechteck*

Ein lateinisches $r \times n$-Rechteck ($r \leq n$) ist eine Anordnung der Zahlen $1, \ldots, n$ in r Zeilen und n Spalten, so dass in jeder Zeile und jeder Spalte jede Zahl höchstens einmal vorkommt. Ein lateinisches n-Quadrat ist ein lateinisches $n \times n$-Rechteck.

$$\begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Beispiel eines lateinischen 3×4-Rechtecks.

Angenommen wir haben ein lateinisches $r \times n$-Rechteck mit $r < n$ gegeben. Wir wollen sehen, ob wir es zu einem lateinischen n-Quadrat erweitern können. Das heisst, wir wollen $n - r$ weitere Zeilen mit Zahlen aus $1, \ldots, n$ zu dem Rechteck hinzufügen, ohne eine Spalte oder eine Zeile in der eine Zahl mehrfach vorkommt zu erhalten.

(c) Benutzen Sie ihr Ergebnis aus (b) um zu beschreiben, welche $r \times n$-Rechtecke man zu einem lateinischen n-Quadrat erweitern kann.

Aufgabe 1 – *Lateinisches Rechteck*

Ein lateinisches $r \times n$-Rechteck ($r \leq n$) ist eine Anordnung der Zahlen $1, \ldots, n$ in r Zeilen und n Spalten, so dass in jeder Zeile und jeder Spalte jede Zahl höchstens einmal vorkommt. Ein lateinisches n-Quadrat ist ein lateinisches $n \times n$-Rechteck.

$$\begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Beispiel eines lateinischen 3×4-Rechtecks.

Angenommen wir haben ein lateinisches $r \times n$-Rechteck mit $r < n$ gegeben. Wir wollen sehen, ob wir es zu einem lateinischen n-Quadrat erweitern können. Das heisst, wir wollen $n - r$ weitere Zeilen mit Zahlen aus $1, \ldots, n$ zu dem Rechteck hinzufügen, ohne eine Spalte oder eine Zeile in der eine Zahl mehrfach vorkommt zu erhalten.

(d) Geben Sie einen Algorithmus an, der als Eingabe ein Lateinisches $r \times n$-Rechteck nimmt und es zu einem lateinischen n-Quadrat erweitert (falls ein solches existiert) und ansonsten “Nicht möglich” ausgibt.

a) Zeige, dass jeder Baum höchstens ein perfektes Matching hat.

Hint: Betrachte die Blätter

b) Wie viele perfekte Matchings gibt es in K_{2m}?

c) Seien M und N zwei Matchings im Graphen G mit $|M| > |N|$.

Zeige, dass es zwei Matchings M', N' gibt mit $|M'| = |M| - 1$

und $|N'| = |N| + 1$, s.d. $M \cap N = M' \cap N'$ und $M \cup N = M' \cup N'$.