Algorithmen und Wahrscheinlichkeit

Woche 13

Ilya Maier

Nachbesprechung Serie

- f heisst ganzzahlig, wenn $\forall e \in E : f(e) \in \mathbb{Z}$

Semester Recap

Semester Recap

Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
    4.1. Erwartungswert
    4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Semester Recap

Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

k-Zusammenhang

Ein Graph $G = (V, E)$ ist **zusammenhängend** $\iff \forall u, v \in V, u \neq v : \exists u, v$**-Pfad** in G

<u>Knoten</u>

$X \subseteq V$

k-zusammenhängend
1) $|V| \geq k + 1$
2) $\forall X \subseteq V : |X| < k \implies G[V \backslash X]$ zusammenhängend

Satz von Menger
G k-zusammenhängend
$\iff \forall u, v \in V, u \neq v : \exists k$ intern-knotendisjunkte u, v-Pfade

<u>Kanten</u>

$X \subseteq E$

k-kanten-zusammenhängend
$\forall X \subseteq E : |X| < k \implies (V, E \backslash X)$ zusammenhängend

Satz von Menger
G k-kanten-zusammenhängend
$\iff \forall u, v \in V, u \neq v : \exists k$ kantendisjunkte u, v-Pfade

$\exists v \in V : \deg(v) < k \implies G$ ist nicht k-zusammenhängend

Knotenzusammenhang $\leq$ Kantenzusammenhang $\leq$ minimaler Grad

2-Zusammenhang

Für einen zusammenhängenden Graphen $G = (V, E)$**:**

Knoten

$v \in V$ ist ein **Artikulationsknoten (AK)**

$\iff G - v$ ist **nicht zusammenhängend**

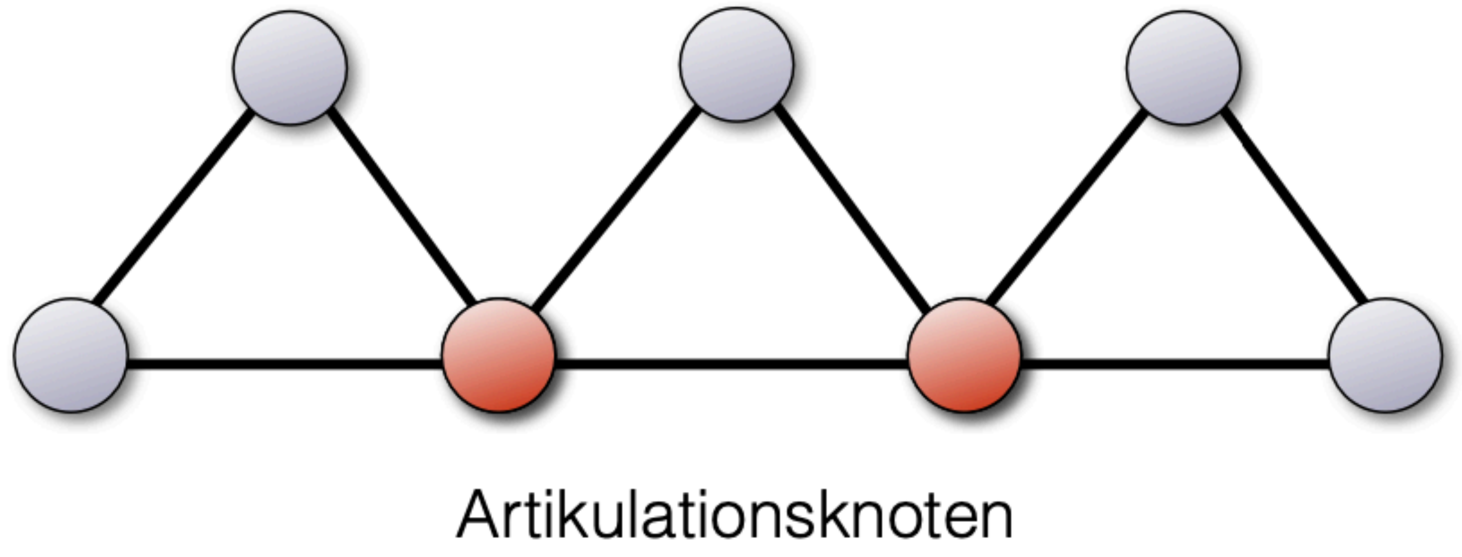
Kanten

$e \in E$ ist eine **Brücke**

$\iff G - e$ ist **nicht zusammenhängend**

$\forall u, v \in V : \{u, v\}$ ist eine **Brücke** $\implies$ $\deg(u) = 1$ oder u ist ein **AK** **und** $\deg(v) = 1$ oder v ist ein **AK**

Umkehrung gilt nicht!



Artikulationsknoten / Brücken finden

low[v]: kleinste **dfs**-Nummer, die man von v aus durch einen gerichteten Pfad aus beliebig vielen Baumkanten und maximal einer Restkante erreichen kann

→ Berechenbar in $O(|V| + |E|)$ mithilfe DP

$$\textbf{low}[v] = \min\left(\operatorname{dfs}[v],\ \min_{(v,w)\in E}\begin{cases}\operatorname{dfs}[w], & \text{falls } (v,w) \text{ Restkante}\\ \operatorname{low}[w], & \text{falls } (v,w) \text{ Baumkante}\end{cases}\right)$$

AK finden

If 1) $v =$ **root** und v hat mind. 2 Kinder in DFS

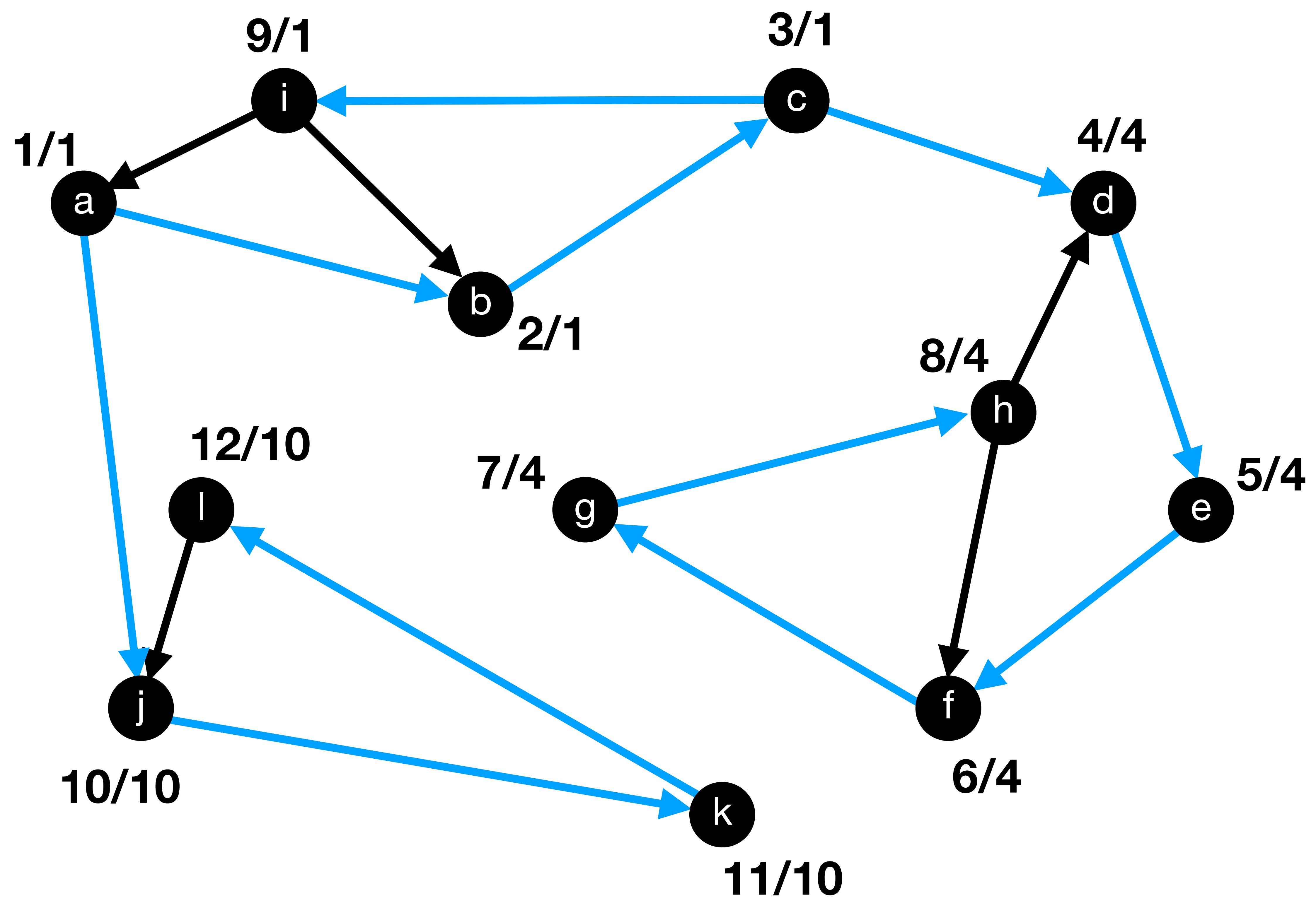
2) $v \neq$ **root** und v hat ein Kind w in DFS s.t. **low[w]** $\geq$ **dfs[v]**

Brücke finden

Eine Baumkante $e = (v, w)$ ist eine Brücke

$\iff$ **low[w]** $>$ **dfs[v]**

Eine Restkante ist nie eine Brücke



Semester Recap

Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

Kreise

Hamiltonkreis: Ein Kreis durch jeden Knoten genau einmal
Eulerzyklus: Ein geschlossener Weg durch jede Kante genau einmal

Ein zsmhgder Graph $G = (V, E)$ hat einen **Eulerzyklus** $\iff \forall v \in V : \deg(v) \equiv_2 0$
Ein zsmhgder Graph $G = (V, E)$ hat einen **Eulerweg** $\iff |\{v \in V | \deg(v) \equiv_2 1\}| \leq 2$

Kann man in $O(|E|)$ finden

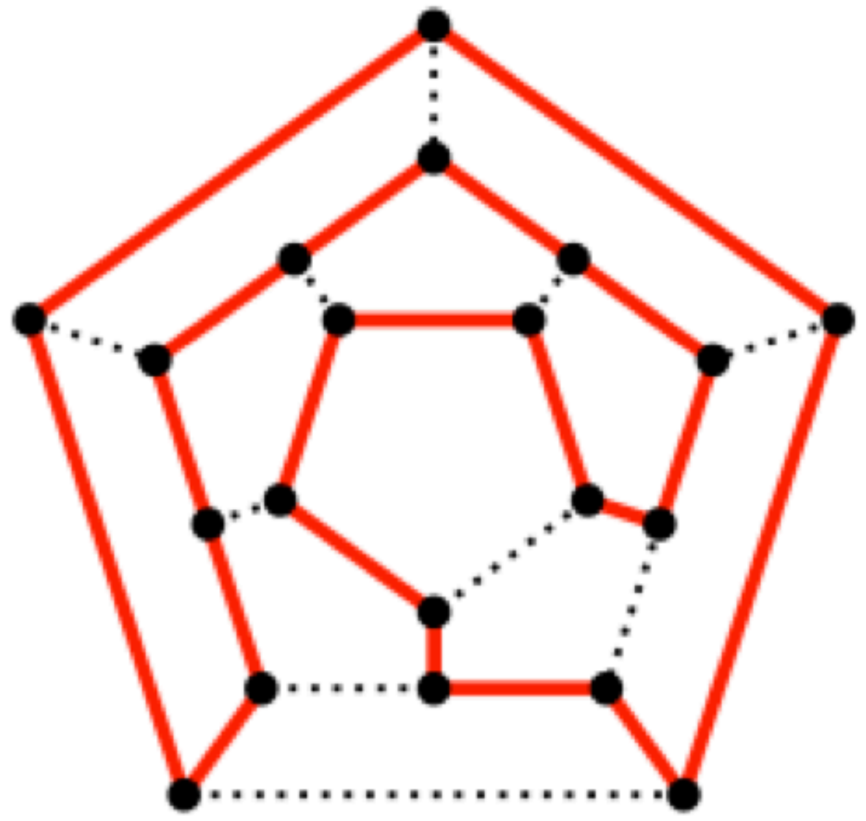
Ein $n \times m$ Gitter hat einen **Hamiltonkreis** $\iff n \times m \equiv_2 0$

d**-dimensional Hyperwürfel** H_d: - Knoten : $\{0,1\}^d$
- Kanten : Jedes Paar von Knoten, die sich nur an einem Ziffer unterscheiden

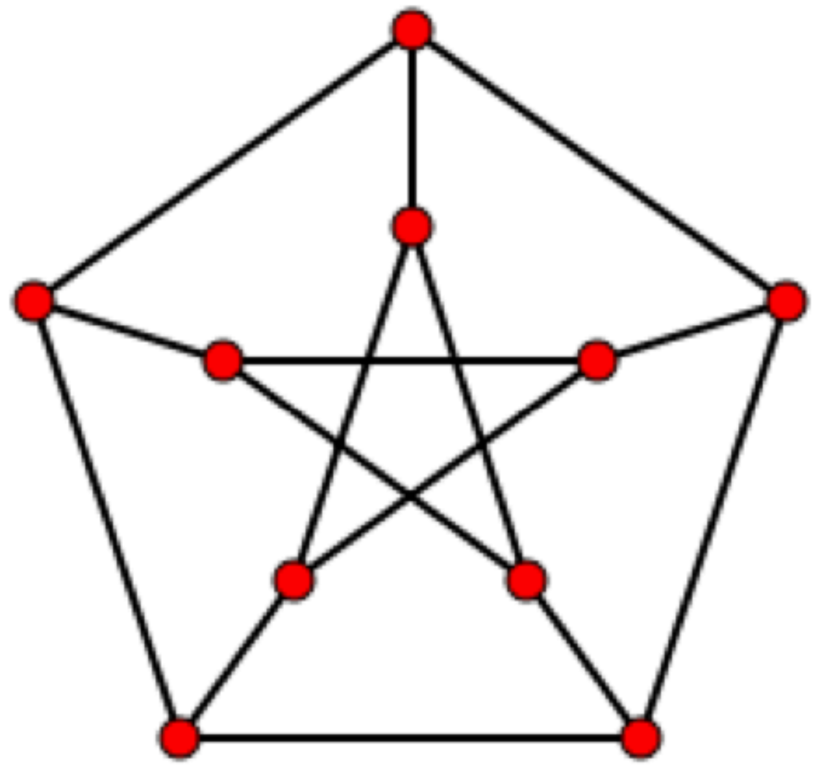
H_d hat einen **Hamiltonkreis**

Satz von Dirac: Jeder Graph $G = (V, E)$ mit $|V| \geq 3$ und Minimalgrad $\delta(G) \geq \frac{|V|}{2}$ enthält einen Hamiltonkreis

Hamilton Kreise



Ikosaeder



Petersengraph

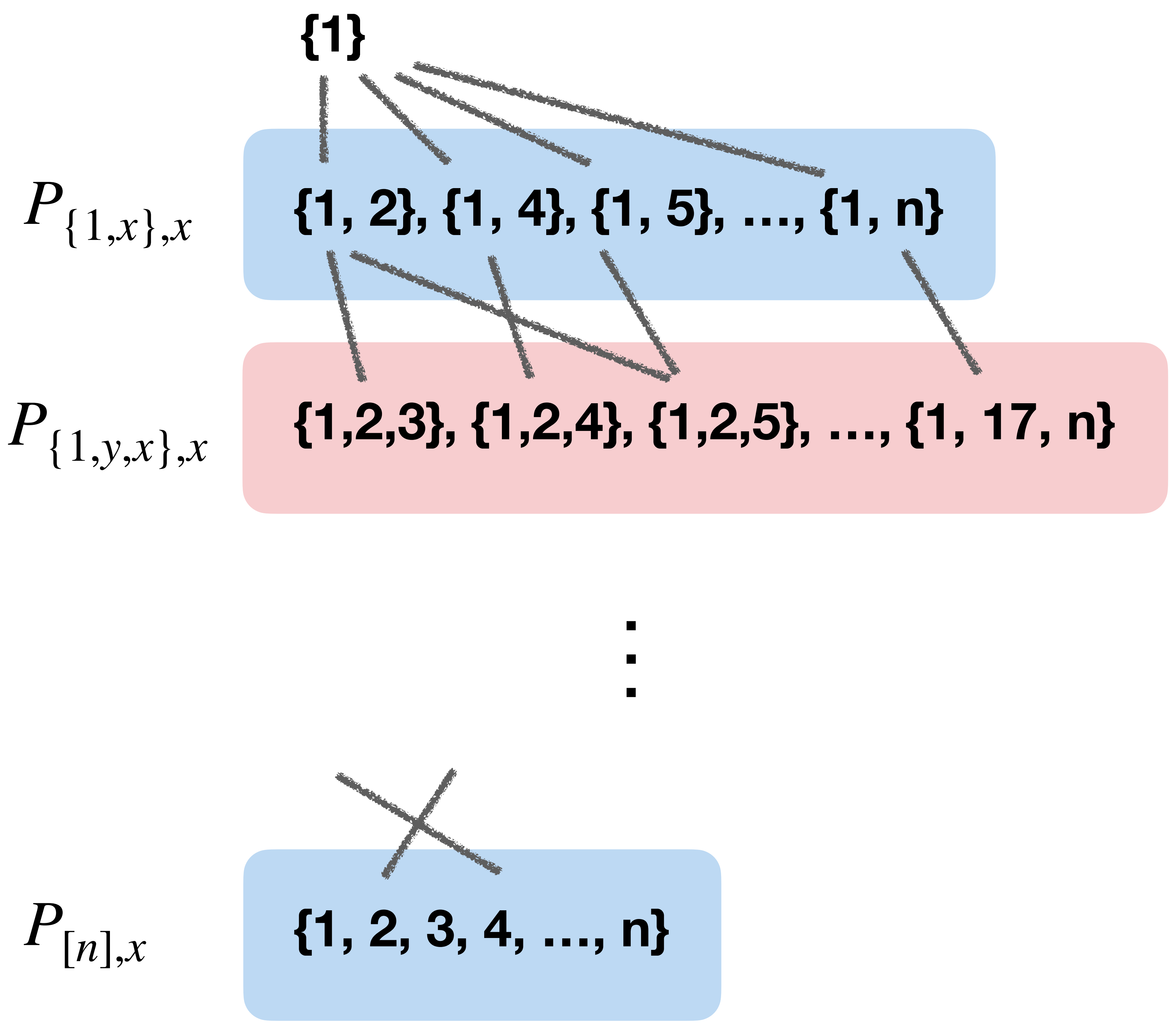
DP-Hamiltonkreis

Für alle $S \subseteq [n]$ mit $1 \in S$ und alle $x \in S$ mit $x \neq 1$:

$P_{S,x} = 1 \iff$

$\exists$ 1-x-Pfad, der genau aus den Knoten von S besteht

Visuelle Darstellung:



Initialisierung:

$\forall x \in \{2,\ldots,n\} : P_{\{1,x\},x} = 1 \iff \{1,x\} \in E$

Berechnung:

for all $s = 3$ to n:
  for all $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$:
    for all $x \in S$ mit $x \neq 1$:
      $P_{S,x} = \max\{P_{S\setminus\{x\},y} : y \in \mathcal{N}(x) \cap S\}$

G hat einen Hamiltonkreis $\iff \exists x \in \mathcal{N}(1)$ und $P_{[n],x} = 1$

Laufzeit: $O(2^n \cdot n^2)$

Speicherplatz: $O(2^n \cdot n)$

Semester Recap

Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

Das Traveling Salesman Problem (TSP)

Gegeben: ein kompletter Graph K_n mit n Knoten & Distanzen zw. je zwei Knoten: $l : \binom{[n]}{2} \to \mathbb{N}_0$

Gesucht: Kürzester Hamiltonkreis: $\underset{C\text{:Hamiltonian cycle}}{\operatorname{argmin}} \sum_{e \in C} l(e)$

Reduktion von HK Problem auf TSP $\implies$ TSP ist **NP-vollständig**

HK reduzierbar auf TSP mit $l(e) = \begin{cases} 0 & \text{falls } e \in E \\ 1 & \text{falls } e \notin E \end{cases}$

α-Approximation: $\sum_{e \in C} l(e) \leq \alpha \cdot \operatorname{opt}(K_n, l)$

Metrisches TSP 2-Approximation

Metrisch: $l(\{x,z\}) \leq l(\{x,y\}) + l(\{y,z\}) \quad \forall x,y,z \in [n]$

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

Laufzeit: $O(n^2)$

1. Finde den **MST** T von G.
2. **Verdopple** alle Kanten in T
3. Bestimmt **Eulertour** W
4. **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. Vom Verdoppeln: $2l(T) \leq 2\text{opt}(K_n, l)$
3. Eulertour $l(W) = 2l(T) \leq 2\text{opt}(K_n, l)$
4. Abkürzen: $l(C) \leq l(W) = 2l(T) \leq 2\text{opt}(K_n, l)$

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 $U := \{v \in T \,|\, \deg(v) \text{ ungerade}\}$
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: $l(C) \leq l(W) \leq 1.5\text{opt}(K_n, l)$

Semester Recap

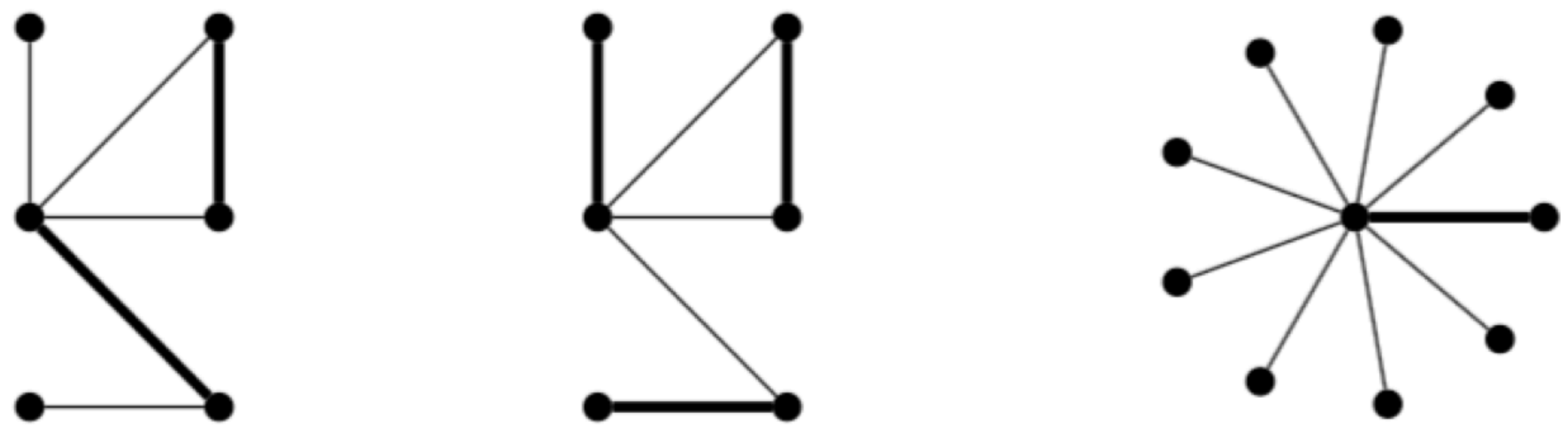
Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

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: $|M_{ink}| \geq \dfrac{|M_{kar}|}{2}$

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)

$\implies |M_{kar}| \leq$ #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 $\mathcal{N}(X) \leq k|\mathcal{N}(X)|$
$\implies$: $\forall X \subseteq A : |X| \leq |\mathcal{N}(X)|$

Matchings - Matching Algorithmen

Greedy
Wähle zufällig eine Kante und lösche sie und die inzidenten Kanten bis $|E| = \emptyset$
$\Longrightarrow$ 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

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 $\implies$ **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

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

Semester Recap

Teil 1: Graphentheorie

1. Zusammenhang
2. Kreise
3. Traveling Salesman Problem (TSP)
4. Matchings
5. Färbungen

Färbungen

Färbung eines Graphen (V, E) **mit** k **Farben**: eine Abbildung $c : V \to [k]$ s.d. $c(u) \neq c(v)$ für alle Kanten $\{u, v\} \in E$

bzw. $V = V_1 \dot{\cup} \ldots \dot{\cup} V_k$, wobei V_i keine Kanten enthält, V_i := Farbklasse

Chromatische Zahl $\chi(G)$: minimale Anzahl Farben, die für eine Färbung von G benötigt wird.

$$\chi(G) \leq k \iff G \text{ ist } k\text{-partit}$$

Gegeben ein Graph $G = (V, E)$**, gilt** $\chi(G) \leq k$?

$\underline{k = 2}$: In $O(|V| + |E|)$ Zeit mit BFS (keine ungeraden Kreise)

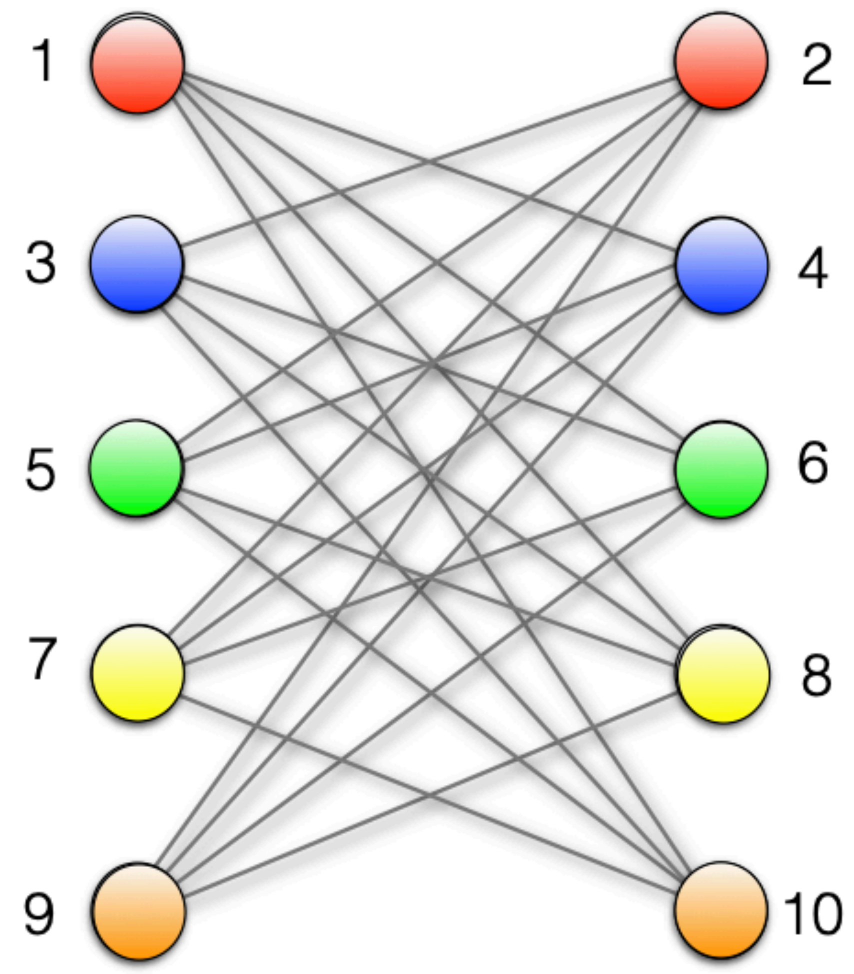
$\underline{k > 2}$: **NP-vollständig**

Farbklassen tauschen:

Falls wir **jeden Block** mit k Farben färben können,

können wir **den ganzen Graphen** mit k Farben färben

Greedy-Färbung



wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, v_2, \ldots, v_n\}$

$c[v_1] \leftarrow 1$

for $i = 2$ **to** $i = n$ **do**

$\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \,|\, k \neq c(u)$ **für alle** $u \in \mathcal{N}(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Für jede Reihenfolge der Knoten braucht der Greedy-Algo höchstens $\Delta(G) + 1$ viele Farben

Es gibt eine Reihenfolge der Knoten, für die der Greedy-Algo nur $\chi(G)$ viele Farben braucht

Es gibt bipartite Graphen und eine Reihenfolge der Knoten, für die der Greedy-Algo $|V|/2$ viele Farben braucht

Heuristik:

v_n = Knote vom kleinsten Grad. Lösche v_n

v_{n-1} = Knote vom kleinsten Grad im Restgraph. Lösche v_{n-1}. Iteriere

Bemerkungen:

1) Die Heuristik findet immer eine Färbung mit 2 Farben für **Bäume**
2) Die Heuristik findet eine Färbung mit 6 Farben für **planare Graphen**
3) Falls $G = (V, E)$ zusammenhängend und $\exists v \in V : \deg(v) < \Delta(G)$: (Alle Graphen außer reguläre Graphen)
   Die Heuristik liefert Reihenfolge, für die der Greedy-Algo höchstens $\Delta(G)$ Farben braucht

3-Färbung

Satz: Einen **3-färbbaren** Graphen kann man in Zeit $O(|V| + |E|)$ mit $O(\sqrt{|V|})$ Farben färben

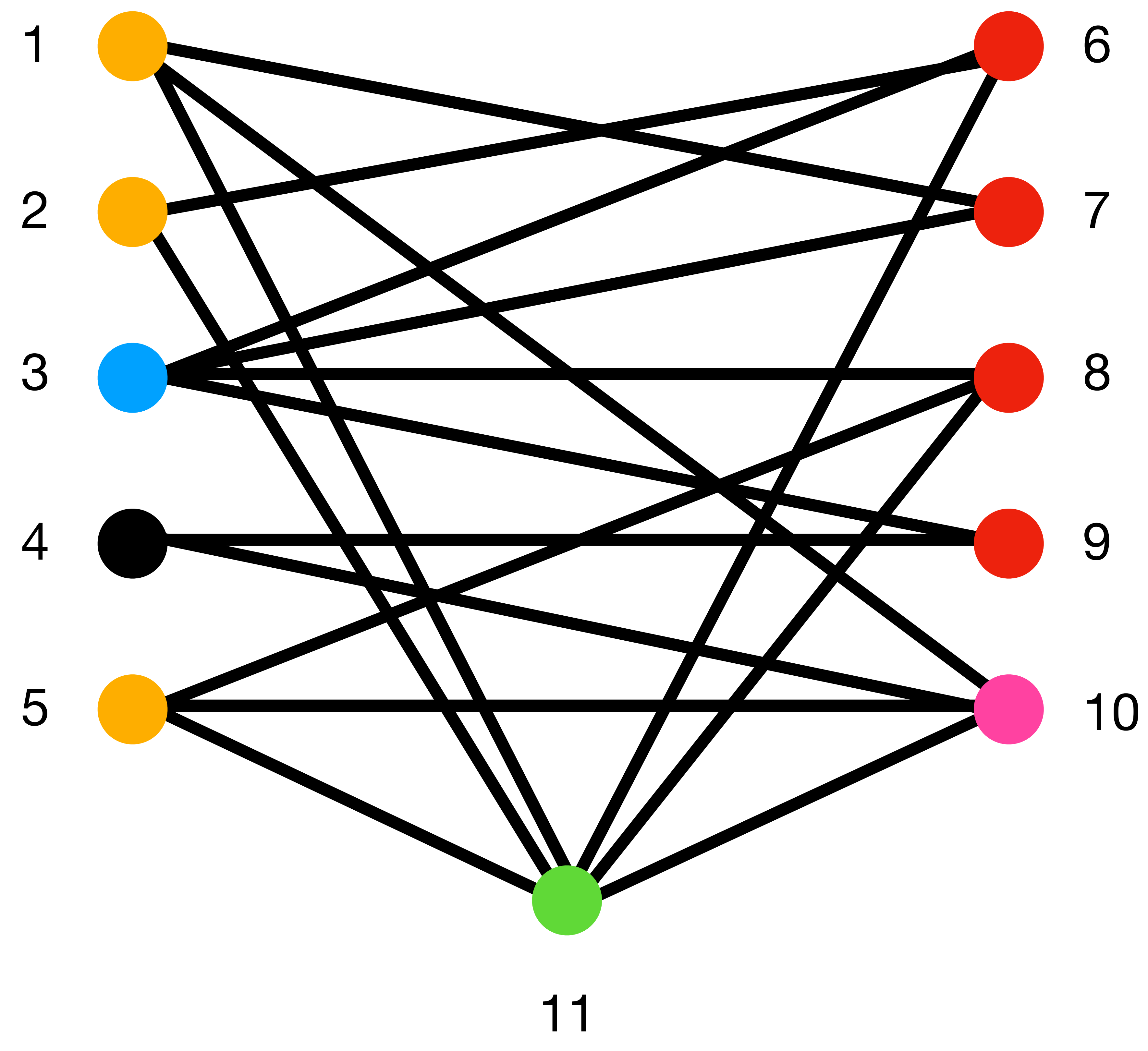
Algorithmus:

while es gibt Knoten v mit $\deg(v) > \sqrt{|V|}$

färbe v mit neuer Farbe und seine Nachbarn mit 2 weiteren neuen Farben

lösche alle gefärbten Knoten, der Restgraph hat $\Delta(G) \leq \sqrt{|V|}$

färbe verbleibende Knoten greedy mit $\Delta + 1$ neuen Farben



Satz von Brooks

$G \neq K_n, G \neq C_{2n+1}, G$ zsmhd $\implies G$ kann in $O(|E|)$ mit $\Delta(G)$ Farben gefärbt werden

- Falls Δ(G

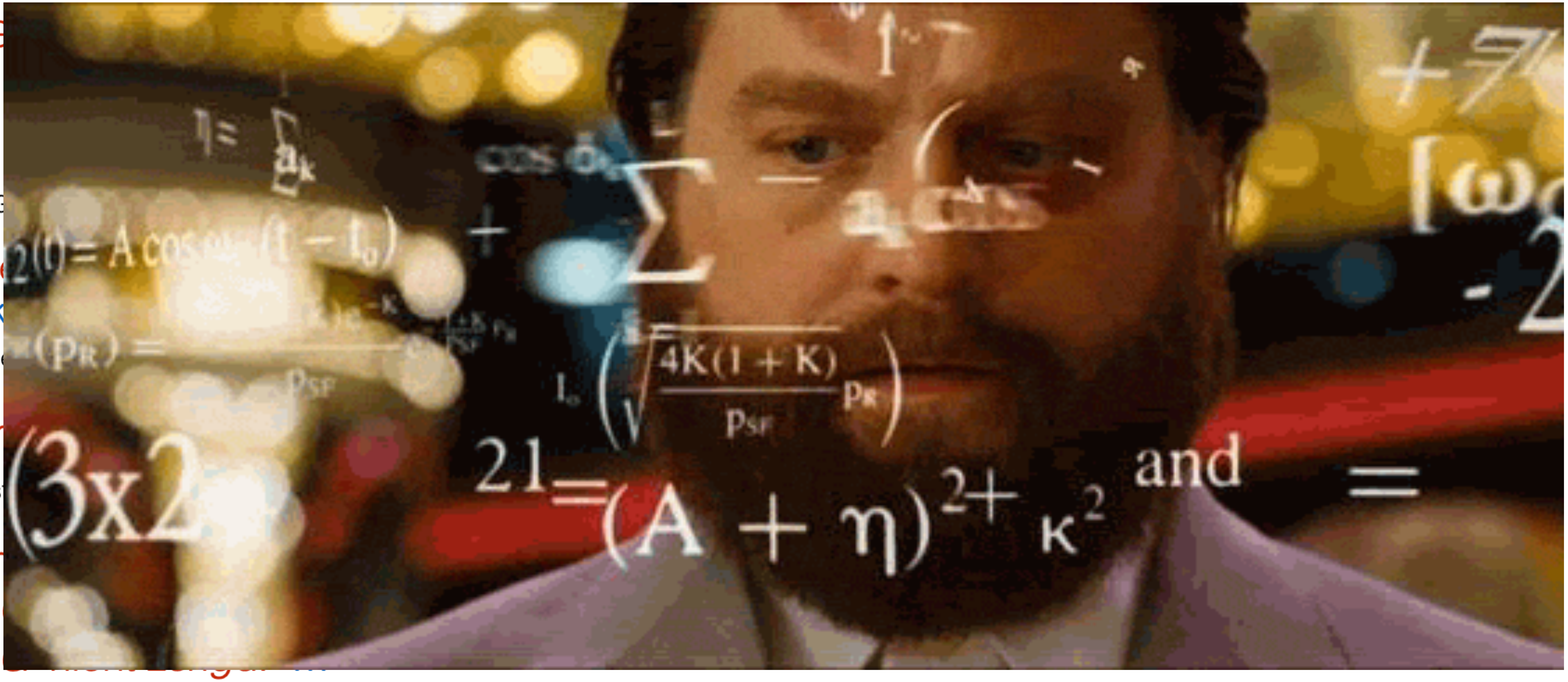
  (da G zshgd
- Falls ∃ v

  (geht mit Δ(G
- Falls es

  Heuristik

  (in allen dies
- Bestimm

  (diese exis
- Betrach
  - *Falls*
  - *Falls*



Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
   4.1. Erwartungswert
   4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Kombinatorik

	Geordnet	Ungeordnet
Mit Zürücklegen	n^k	$\binom{n+k-1}{k}$
Ohne Zürücklegen	$n^{\underline{k}}$	$\binom{n}{k}$

	Geordnet	Ungeordnet
Mit Zürücklegen	n^k	$\frac{(n+k-1)!}{k!(n-1)!}$
Ohne Zürücklegen	$\frac{n!}{(n-k)!}$	$\frac{n!}{k!(n-k)!}$

Beispiele:

1. Anzahl der verschiedenen bit strings der Länge k
   = 2^k
2. Anzahl Möglichkeiten 11 Spieler aus einer Mannschaft von 22 auszuwählen, wobei die Reihenfolge wichtig ist.
   $$= \frac{22!}{(22-11)!}$$
3. Anzahl Möglichkeiten 3 Kugeln aus 5 verschiedenen Kugelfarben zu ziehen, wenn jede Kugel nach dem Ziehen zurückgelegt wird?
   $$= \binom{5+3-1}{3}$$
4. Anzahl der möglichen Kanten im Graphen
   $$= \binom{n}{2}$$
   → ziehe 2 Elemente aus $[n]$ ohne zurücklegen

Wahrscheinlichkeit - Grundbegriffe

<u>Diskreter Wahrscheinlichkeitsraum</u>: $(\Omega, \Pr[\,\cdot\,])$

Ergebnismenge Ω: Menge von <u>Elementarereignissen</u>

Ereignis E: $E \subseteq \Omega$, d.h. eine Menge von Elementarereignissen

Komplementärereignis $\overline{E}$ von E: $\overline{E} := \Omega \backslash E$

1. $\forall \omega \in \Omega : 0 \leq \Pr[\omega] \leq 1$
2. $\sum_{\omega \in \Omega} \Pr[\omega] = 1$
3. $\forall E \subseteq \Omega : \Pr[E] = \sum_{\omega \in E} \Pr[\omega]$

<u>Laplace-Raum</u>: Endlicher W-Raum in dem alle Elementarereignisse gleich wahrscheinlich sind

$$\forall \omega \in \Omega : \Pr[\omega] = \frac{1}{|\Omega|}, \Pr[E] = \frac{|E|}{|\Omega|}$$

<u>Bedingte Wahrscheinlichkeit:</u>

Für Ereignisse A, B s.d. $\Pr[B] > 0$, $\Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]}$

$$\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B] = \Pr[B \mid A] \cdot \Pr[A]$$

Wahrscheinlichkeit - Lemmas

Additionssatz

Für paarweise disjunkte Ereignisse $A_1, \ldots, A_n$

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{i=1}^{n} \Pr[A_i]$$

Boolsche Ungleichung

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \Pr[A_i]$$

Siebformel

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{k=1}^{n} \left((-1)^{k+1} \sum_{S \subseteq [n], |S|=k} \Pr\left[\bigcap_{i \in S} A_i\right]\right)$$

1) $\Pr[\emptyset] = 0, \Pr[\Omega] = 1$
2) $0 \leq \Pr[A] \leq 1$
3) $\Pr[\overline{A}] = 1 - \Pr[A]$
4) $A \subseteq B \implies \Pr[A] \leq \Pr[B]$

Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
   4.1. Erwartungswert
   4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Bedingte Wahrscheinlichkeiten

Multiplikationssatz: Für $A_1, \ldots, A_n$, falls $\Pr[A_1 \cap A_2 \cap \cdots \cap A_n] > 0$,

$$\Pr[A_1 \cap \cdots \cap A_n] = \Pr[A_1] \cdot \Pr[A_2 \mid A_1] \cdot \Pr[A_3 \mid A_1 \cap A_2] \cdots \Pr[A_n \mid A_1 \cap \cdots \cap A_{n-1}]$$

Satz der totalen Wahrscheinlichkeit: Für paarweise disjunkte Ereignisse $A_1, \ldots, A_n$ und B, s.d. $B \subseteq A_1 \cup \cdots \cup A_n$,

$$\Pr[B] = \sum_{i=1}^{n} \underbrace{\Pr[B \mid A_i] \cdot \Pr[A_i]}_{=\Pr[B \cap A_i]}$$

Satz von Bayes: Für paarweise disjunkte Ereignisse $A_1, \ldots, A_n$ und B, s.d. $B \subseteq A_1 \cup \cdots \cup A_n$, $\Pr[B] > 0$,

$$\forall i \in [n] : \Pr[A_i \mid B] = \frac{\Pr[A_i \cap B]}{\Pr[B]} = \frac{\Pr[B \mid A_i] \cdot \Pr[A_i]}{\sum_{j=1}^{n} \Pr[B \mid A_j] \cdot \Pr[A_j]}$$

Geburtstagsproblem

Für m Bälle und n Körbe

A_i = i-ter Ball landet in einem Korb in dem noch kein Ball liegt

→ $\Pr[A_1] = 1$

→ $\Pr[A_2 | A_1] = \dfrac{n-1}{n}$

→ $\Pr[A_3 | A_1 \cap A_2] = \dfrac{n-2}{n}$

→ $\Pr[A_i | A_1 \cap \cdots \cap A_{i-1}] = \dfrac{n-(i-1)}{n} = 1 - \dfrac{i-1}{n}$

$$\implies \Pr[A_1 \cap \cdots \cap A_m] = 1 \cdot (1 - \frac{1}{n}) \cdot (1 - \frac{2}{n}) \cdot \cdots \cdot (1 - \frac{m-1}{n})$$

$$\begin{aligned}
&\Pr[A_1 \cap \cdots \cap A_m] \\
&= 1 \cdot (1 - \frac{1}{n}) \cdot (1 - \frac{2}{n}) \cdots \cdot (1 - \frac{m-1}{n}) \\
&\leq e^{-\frac{1}{n}} e^{-\frac{2}{n}} \cdots e^{-\frac{m-1}{n}} \\
&= e^{-\frac{1}{n}(1+2+3+\cdots+(m-1))} \\
&= e^{-\frac{(m-1)m}{2n}}
\end{aligned}$$

Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
    4.1. Erwartungswert
    4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Unabhängigkeit

Zwei Ereignisse A und B heissen **unabhängig** $\iff$ $\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]$

Ereignisse $A_1, \ldots A_n$ heissen **unabhängig** $\iff$ für jede Teilmenge $I \subseteq [n]$ mit $I = \{i_1, \ldots, i_k\}$

$$\Pr[A_{i_1} \cap \ldots \cap A_{i_k}] = \Pr[A_{i_1}] \cdot \cdots \cdot \Pr[A_{i_k}]$$

Ereignisse $A_1, A_2, \ldots$ heissen **unabhängig** $\iff$ $\forall n \in \mathbb{N} : A_1, \ldots, A_n$ unabhängig sind

Lemmas:

$A_1, A_2, \ldots, A_n$ sind unabhängig $\iff$ $\Pr[A_1^{s_1} \cap \ldots \cap A_n^{s_n}] = \Pr[A_1^{s_1}] \cdot \cdots \cdot \Pr[A_n^{s_n}]$ für alle $(s_1, \ldots, s_n) \in \{0,1\}^n$,
wobei $A_i^0 = \bar{A}_i$ und $A_i^1 = A_i$

$A_1, A_2, B_1, \ldots, B_n$ sind unabhängig $\implies$ $A_1 \cup A_2, B_1, \ldots, B_n$ sind unabhängig

$A_1, A_2, B_1, \ldots, B_n$ sind unabhängig $\implies$ $A_1 \cap A_2, B_1, \ldots, B_n$ sind unabhängig

Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
   4.1. Erwartungswert
   4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Zufallsvariablen

Dichtefunktion: $f_X : \mathbb{R} \to [0,1], x \mapsto \Pr[X = x]$

Eine **Zufallsvariable** ist eine Funktion $X : \Omega \to \mathbb{R}$

Verteilungsfunktion: $F_X : \mathbb{R} \to [0,1], x \mapsto \Pr[X \leq x]$

Erwartungswert: $\mathbb{E}[X] := \sum_{x \in W_x} x \cdot \Pr[X = x] = \sum_{\omega \in \Omega} X(\omega) \cdot \Pr[\omega]$

Lemma: Für eine ZV X mit $W_X \subseteq \mathbb{N}_0$ gilt $\mathbb{E}[X] := \sum_{i=1}^{\infty} \Pr[X \geq i]$

Linearität: Sei $X = a_1 X_1 + \ldots + a_n X_n + b$, dann gilt: $\mathbb{E}[X] := a_1 \mathbb{E}[X_1] + \ldots + a_n \mathbb{E}[X_n] + b$

Indikatorvariablen

Eine **Indikatorvariable** ist eine Funktion $I_E : \Omega \to \{0,1\}$ mit $I_E(\omega) = 1 \iff \omega \in E$

$$\mathbb{E}[I_E] = \Pr[E]$$

Rechnen mit Indikatorvariablen:

Komplement $\bar{A}$ $\quad I_{\bar{A}} = 1 - I_A$

Schnitt $A_1 \cap \ldots \cap A_n = B$ $\quad I_B = I_{A_1} \cdot \cdots \cdot I_{A_n}$

Vereinigung $A_1 \cup \ldots \cup A_n = C$ $\quad I_C = 1 - I_{\bar{C}} = 1 - \prod_{i=1}^{n} I_{\bar{A}_i} = 1 - \prod_{i=1}^{n} (1 - I_{A_i})$

Bedingte Zufallsvariablen

Definition: $\Pr[X = x \,|\, A] = \dfrac{\Pr[\{\omega \in A : X(\omega) = x\}]}{\Pr[A]}$

Erwartungswert: $\mathbb{E}[X \,|\, A] = \sum_{x \in W_X} x \cdot \Pr[X = x \,|\, A] = \sum_{\omega \in A} X(\omega) \cdot \Pr[\omega \,|\, A]$

Totale Wahrscheinlichkeit: Seien $A_1, \ldots, A_n$ disjunkt mit $A_1 \cup \ldots \cup A_n = \Omega$ und $\Pr[A_i] > 0$, dann gilt:

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X \,|\, A_i] \cdot \Pr[A_i]$$

Stabile Mengen

Stabile Menge: $S \subseteq V$, in dem es keine Kanten zwischen den Knoten gibt

Gesucht: Möglichst grosses stabile Menge

Phase 1: Füge jeden Knoten unabhängig mit Wahrscheinlichkeit p zu S hinzu

Phase 2: Für jede noch vorhandene Kante lösche einen zufälligen Knoten

Satz: Für jeden Graphen $G = (V, E)$ mit $|V| = n$ und $|E| = m$,

findet der Algorithmus eine stabile Menge S mit $\mathbb{E}[|S|] \geq np - mp^2$

Korollar: Für jeden Graphen $G = (V, E)$ mit $|V| = n$ und $|E| = m$,

findet der Algorithmus für $p = \dfrac{n}{2m}$ eine stabile Menge S mit $\mathbb{E}[|S|] \geq \dfrac{n^2}{4m}$

Varianz

Sei X ein Zufallsvariable mit $\mu = \mathbb{E}[X]$

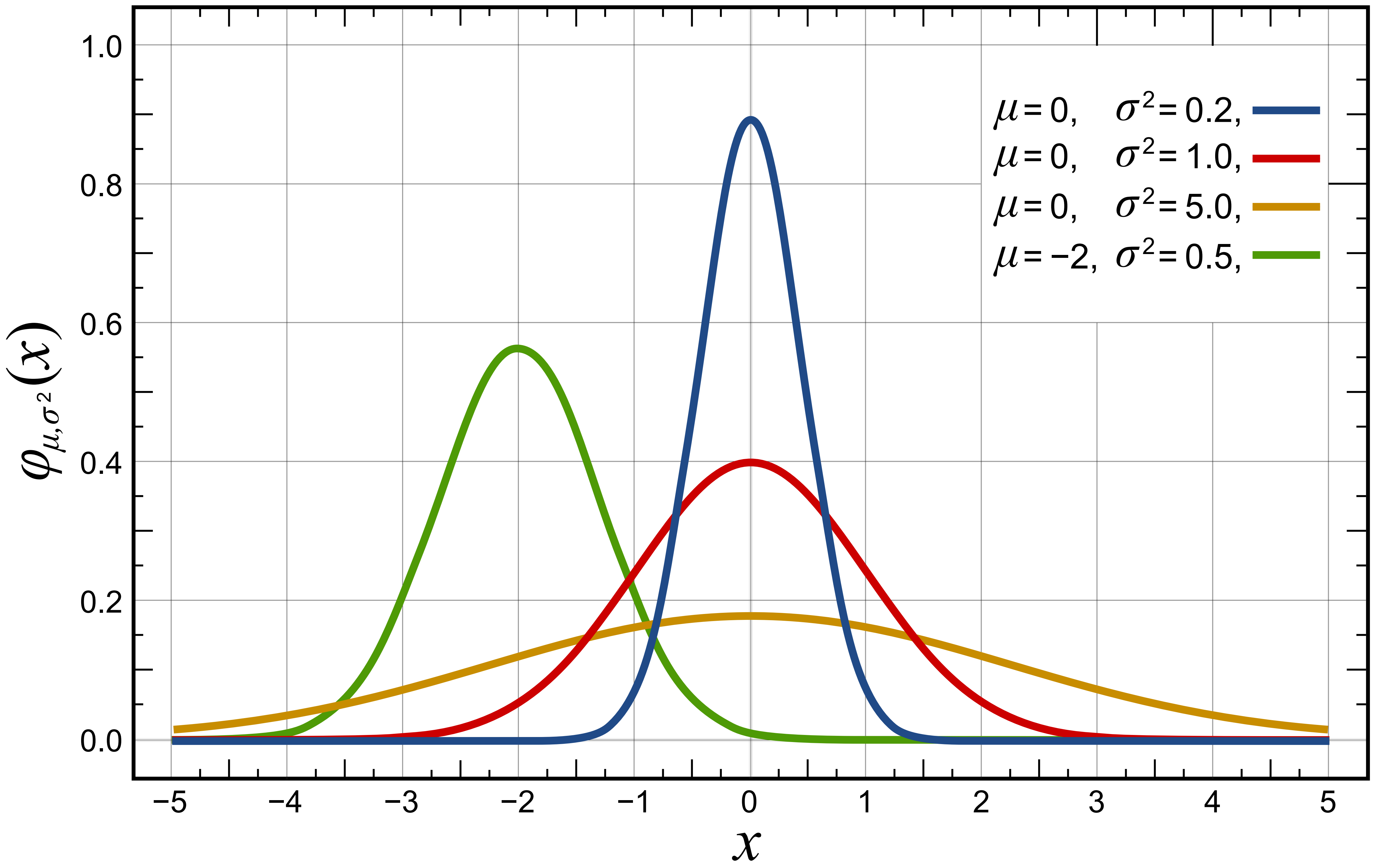
Varianz: $\text{Var}[X] := \mathbb{E}[(X-\mu)^2] = \sum_{x \in W_X} (x-\mu)^2 \cdot \Pr[X = x]$

Standardabweichung von X: $\sigma = \sqrt{\text{Var}[X]}$

Varianz mit Erwartungswert: $\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

Rechenregeln:

1) $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ $\quad \forall X, Y$
2) $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$ $\quad \forall X, Y$ unabhängig
3) $\text{Var}[X+Y] = \text{Var}[X] + \text{Var}[Y]$ $\quad \forall X, Y$ unabhängig
4) $\text{Var}[X \cdot Y] \neq \text{Var}[X] \cdot \text{Var}[Y]$ (in meisten Fällen)
5) $\text{Var}[a \cdot X + b] = a^2 \cdot \text{Var}[X]$ $\quad \forall a, b \in \mathbb{R}$



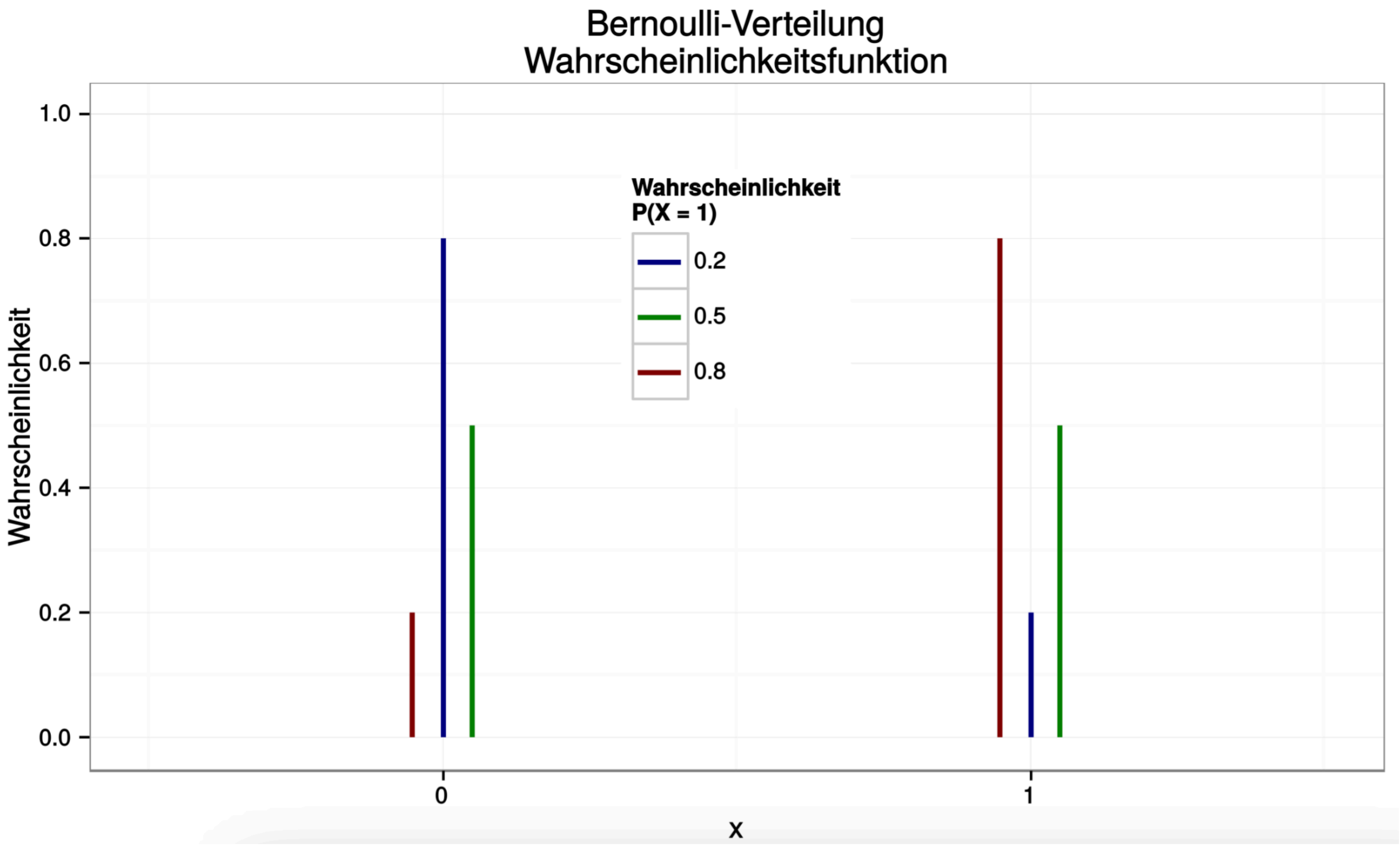
1. Bernoulli-Verteilung

Bezeichnung: $X \sim \text{Bernoulli}(p)$

Wertebereich: $W_X = \{0,1\}$

Dichtefunktion: $f_X(i) = \begin{cases} p & i = 1 \\ 1 - p & i = 0 \\ 0 & \text{sonst} \end{cases}$

Erwartungswert: $\mathbb{E}[X] = p$



Beispiel: Münzenwurf, Indikator für Kopf

2. Binomial-Verteilung

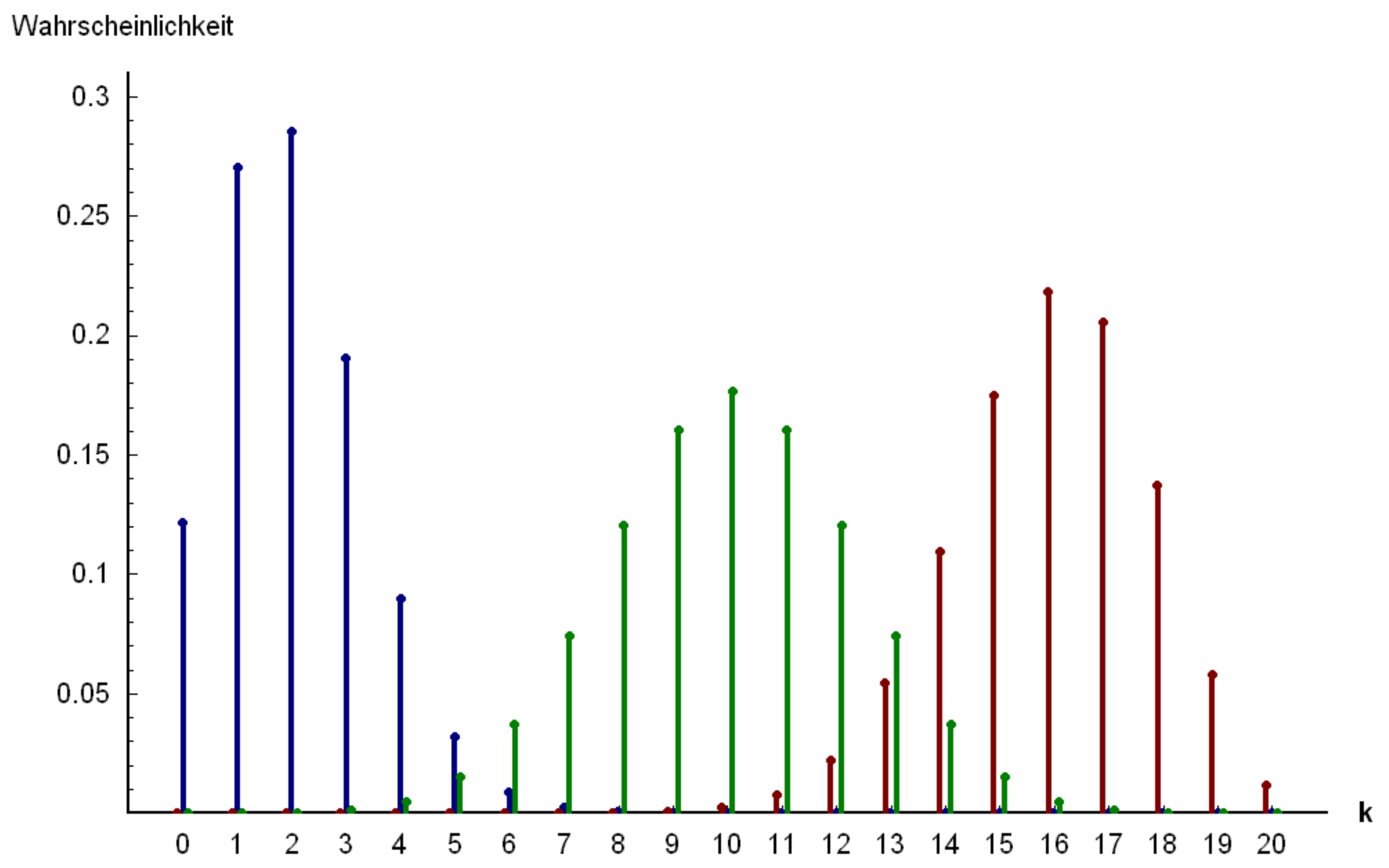
Bezeichnung: $X \sim \text{Bin}(n, p)$

Wertebereich: $W_X = \{0,1,\ldots,n\}$

Dichtefunktion: $f_X(i) = \begin{cases} \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i} & i \in \{0,1,\ldots,n\} \\ 0 & \text{sonst} \end{cases}$

Erwartungswert: $\mathbb{E}[X] = np$

Beispiel: n mal Münzenwurf und wir zählen wie oft Kopf vorkommt



$p = 0.1$ blau

$p = 0.5$ grün

$p = 0.8$ rot

3. Geometrische-Verteilung

Bezeichnung: $X \sim \text{Geo}(p)$

Wertebereich: $W_X = \mathbb{N}$

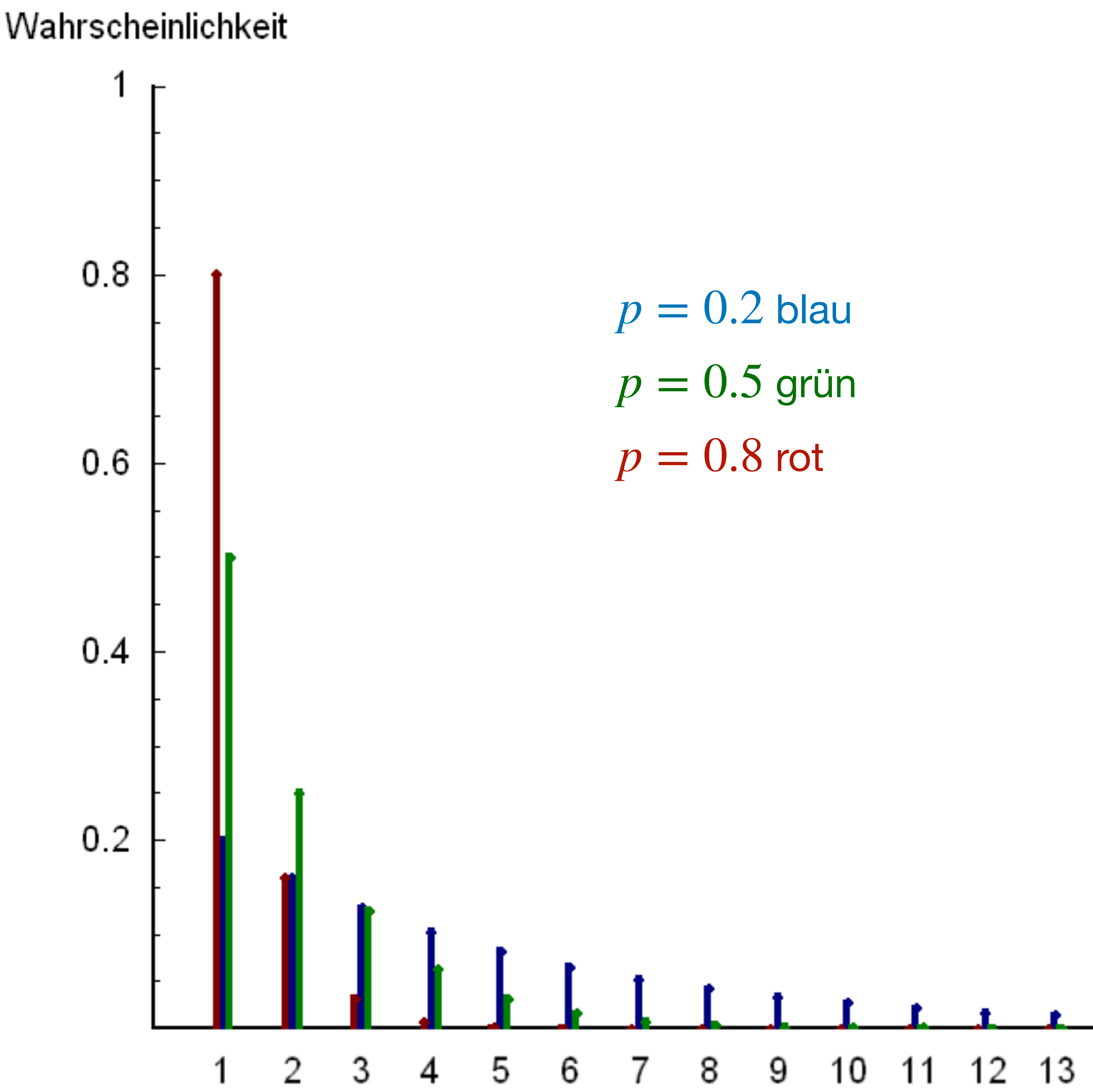
Dichtefunktion: $f_X(i) = \begin{cases} p \cdot (1-p)^{i-1} & i \in \mathbb{N} \\ 0 & \text{sonst} \end{cases}$

Erwartungswert: $\mathbb{E}[X] = \dfrac{1}{p}$

$\Pr[X > t] = (1-p)^t$

Beispiel: Anzahl der Würfe bis das erste Mal Kopf vorkommt

Gedächtnislosigkeit: für alle $s, t \in \mathbb{N}$: $\Pr[X \geq s + t \mid X > s] = \Pr[X \geq t]$



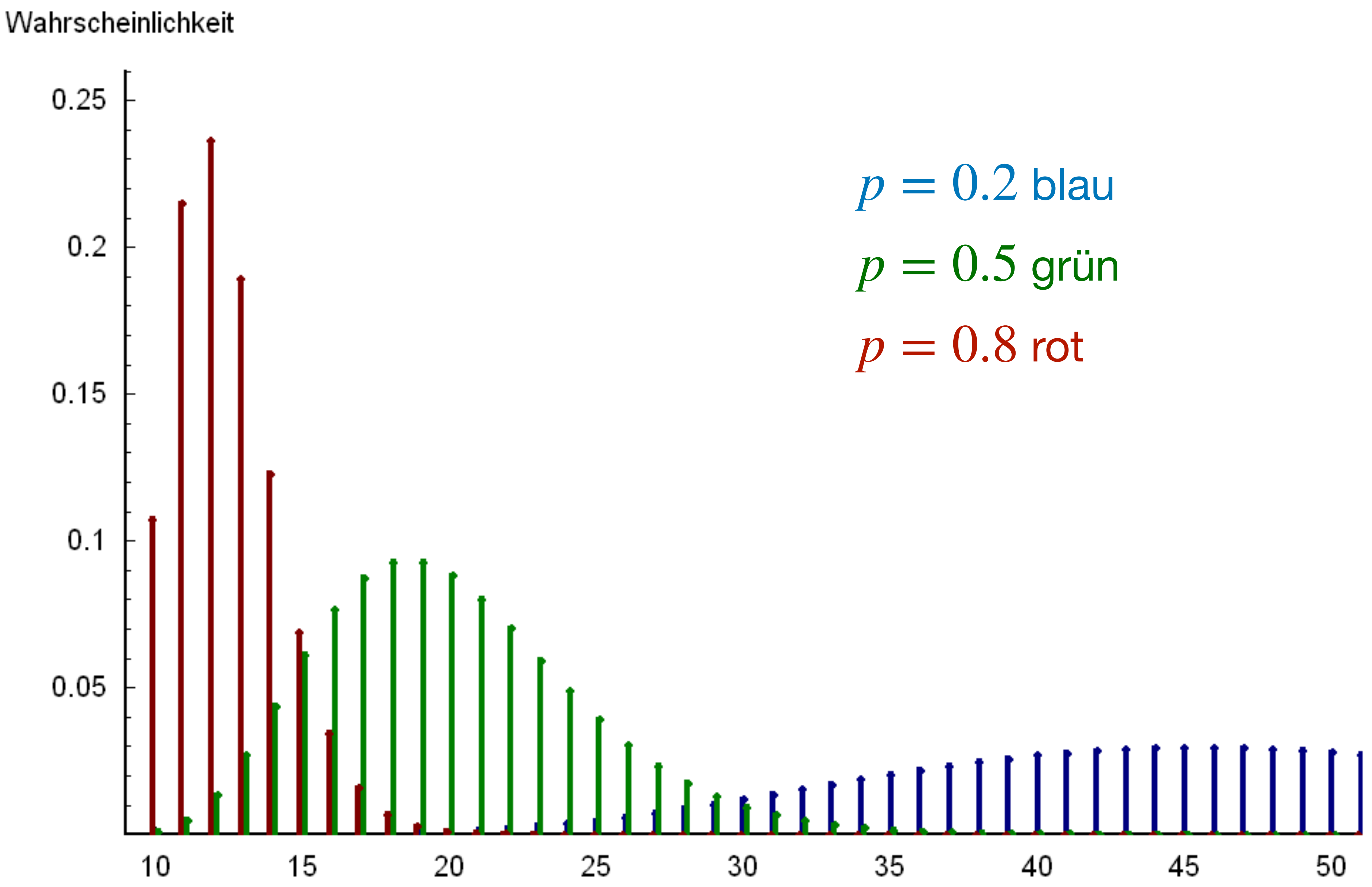
4. Negative Binomial-Verteilung

Bezeichnung: $X \sim$ NegativeBin(n, p)

Wertebereich: $W_X = \mathbb{N}_{\geq n}$

Dichtefunktion: $f_X(i) = \begin{cases} \binom{i-1}{n-1} \cdot p^n \cdot (1-p)^{i-n} & i \geq n \\ 0 & \text{sonst} \end{cases}$

Erwartungswert: $\mathbb{E}[X] = \dfrac{n}{p}$



$p = 0.2$ blau

$p = 0.5$ grün

$p = 0.8$ rot

Beispiel: Anzahl der Versuche bis wir n Mal einen Kopf werfen

5. Poisson-Verteilung

Bezeichnung: $X \sim \text{Po}(\lambda)$

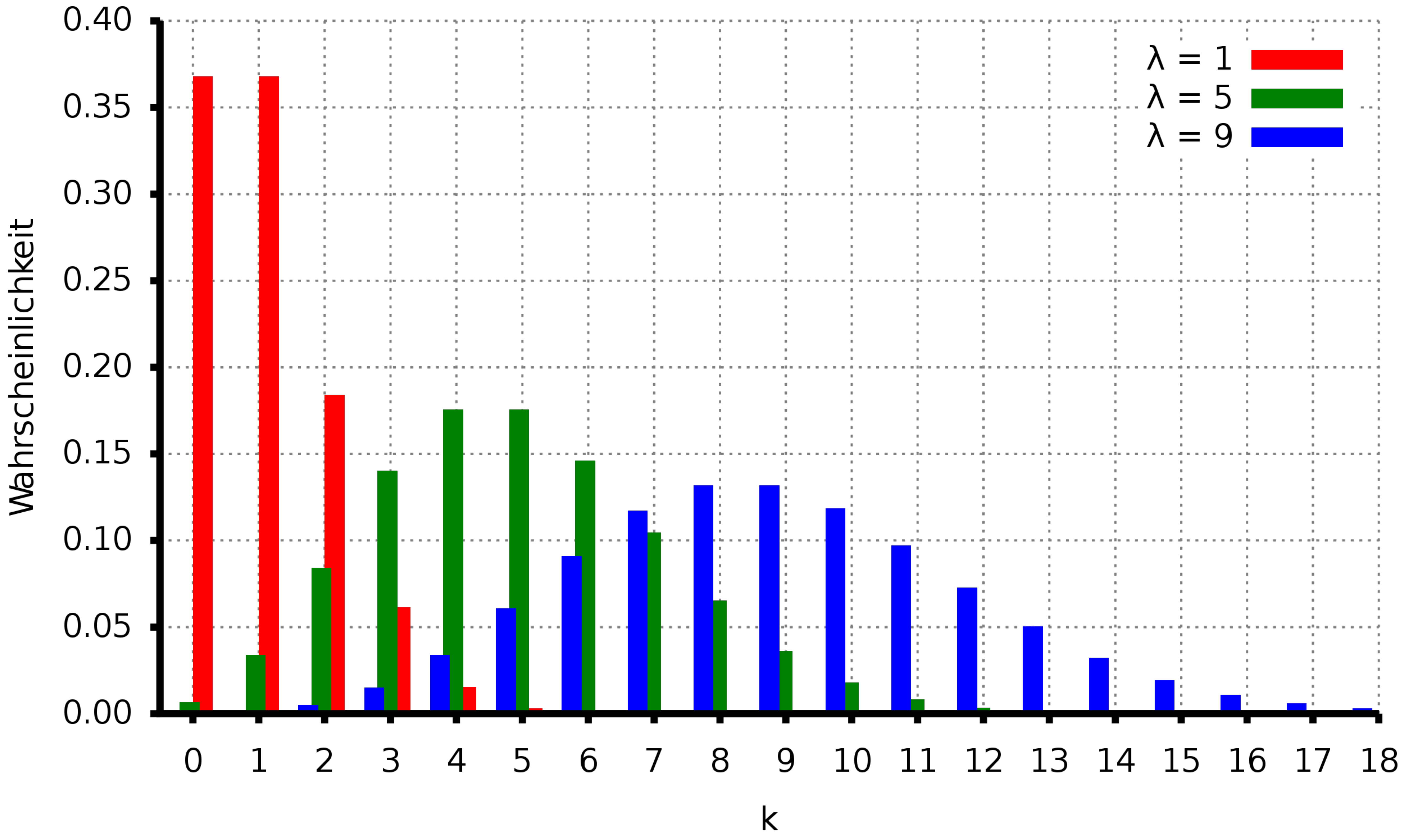
Wertebereich: $W_X = \mathbb{N}_0$

Dichtefunktion: $f_X(i) = \begin{cases} \frac{e^{-\lambda} \cdot \lambda^i}{i!} & i \in \mathbb{N}_0 \\ 0 & \text{sonst} \end{cases}$

Erwartungswert: $\mathbb{E}[X] = \lambda$

Beispiel: Anzahl der Herzinfarkte in der Schweiz, die in einer Stunde auftreten, wenn der gemessene Durchschnitt soweit λ Herzinfarkte pro Stunde ist

Konvergenz: $\text{Bin}(n, \lambda/n)$ konvergiert zu $\text{Po}(\lambda)$ für $n \to \infty$



Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
   4.1. Erwartungswert
   4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Mehrere Zufallsvariablen

Definition: $\Pr[X = x, Y = y] = \Pr[\{\omega \in \Omega : X(\omega) = x, Y(\omega) = y\}]$

Gemeinsame Dichte: $f_{X,Y}(x, y) = \Pr[X = x, Y = y]$

Randdichte: $f_X(x) = \Pr[X = x] = \sum_{y \in W_Y} \Pr[X = x, Y = y]$

Unabhängigkeit:

$X_1, \ldots, X_n$ sind unabhängig $\iff \forall (x_1, \ldots, x_n) \in W_{X_1} \times \ldots \times W_{X_n}$ gilt $\Pr[X_1 = x_1, \ldots, X_n = x_n] = \Pr[X_1 = x_1] \cdot \cdots \cdot Pr[X_n = x_n]$

Korollar: $X_1, \ldots, X_n$ sind unabhängig, so ist dann auch jeder Subset von $X_1, \ldots, X_n$

Mehrere Zufallsvariablen

Satz: Sind $f_1, \dots, f_n$ reellwertige Funktionen $(X_i : \mathbb{R} \to \mathbb{R})$ und seien $X_1, \dots, X_n$ unabhängig,

so sind auch $f_1(X_1), \dots, f_n(X_n)$ unabhängig

Satz: Seien X, Y unabhängig und $Z := X + Y$, dann gilt $f_Z(z) = \sum_{x \in W_X} f_X(x) \cdot f_Y(z - x)$

Waldsche Identität: Seien X, N unabhängig mit $W_N \subseteq \mathbb{N}$. Weiter sei $Z := \sum_{i=1}^{N} X_i$, wobei X_i unabhängige Kopien von X sind.

Dann gilt: $\mathbb{E}[Z] = \mathbb{E}[N] \cdot \mathbb{E}[X]$

Beispiel: N ist die Augenzahl eines Würfels, X Indikator für Kopf. Z ist die Anzahl von Kopf, wenn wir N mal werfen.

$$\mathbb{E}[Z] = \mathbb{E}[N] \cdot \mathbb{E}[X] = \frac{21}{6} \cdot \frac{1}{2} = 1{,}75$$

Semester Recap

Teil 2: Wahrscheinlichkeitstheorie

1. Grundbegriffe, Lemmas
2. Bedingte Wahrscheinlichkeiten
3. Unabhängigkeit
4. Zufallsvariablen
    4.1. Erwartungswert
    4.2. Varianz
5. Mehrere Zufallsvariablen
6. Abschätzen von Wahrscheinlichkeiten

Abschätzen von Wahrscheinlichkeiten

Markovs Ungleichung:

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$$

$\forall X \geq 0$

$\forall t > 0,\ t \in \mathbb{R}$

Chebyshevs Ungleichung:

$$\Pr[\,|X - \mathbb{E}[X]|\, \geq t] \leq \frac{\mathsf{Var}[X]}{t^2}$$

$\forall X$

$\forall t > 0,\ t \in \mathbb{R}$

Chernoffs Ungleichung:

1. $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2\mathbb{E}[X]}$
2. $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{2}\delta^2\mathbb{E}[X]}$
3. $\Pr[X \geq t] \leq 2^{-t}$

$\forall X \sim \mathsf{Bin}(n,p)$

$\forall 0 < \delta \leq 1$

$\forall t \geq 2e\mathbb{E}[X]$

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Randomisierte Algorithmen

Randomisierter Algorithmus : Eingabe I → Algorithmus A mit Zufallszahlen R → Ausgabe $A(I, R)$

- deterministisch: selbe Eingabe, selber Output
- nicht-deterministisch: selbe Eingabe, nicht unbedingt selber Output

Monte Carlo Algorithmus: Primzahltest, Target-shooting
→ Korrektheit ist die Zufallsvariable

- immer gleiche Laufzeit
- manchmal falsches Ergebnis

Las Vegas Algorithmus: Quicksort, Duplikate finden
→ Laufzeit ist die Zufallsvariable
→ Geometrisch verteilt mit $p = \Pr[A(I) \neq \text{"???"}]$

- immer korrekte Antwort
- manchmal dauert zu lange /
  gibt nach einer bestimmter Zeit “???” aus

Fehlerreduktion

Las-Vegas:

Sei A ein Las-Vegas-Algorithmus mit $\Pr[A(I) \neq$ "???"$] \geq \epsilon$

Sei A_δ für $\delta > 0$ ein Algorithmus, der entweder die erste Ausgabe verschieden von $???$ ausgibt
oder der nach $N = \lceil \epsilon^{-1} \cdot \ln(\delta^{-1}) \rceil$ Versuchen $???$ ausgibt

dann gilt $\Pr[A_\delta(I) =$ "???"$] \leq \delta$

ϵ	δ	N
0.1	0.01	47
0.5	0.01	10
0.5	10^{-80}	369
0.9	10^{-30}	77

Fehlerreduktion

Monte-Carlo - Einseitiger Fehler:

$\Pr[A(I) = \text{Ja}] = 1$ für alle Ja-Instanzen I $\implies$ Wenn $A(I) = \text{Ja}$, dann könnte die Ausgabe falsch sein
$\Pr[A(I) = \text{Nein}] \geq \epsilon$ für alle Nein-Instanzen I Wenn $A(I) = \text{Nein}$, dann ist die Ausgabe immer korrekt

Sei A_δ für $\delta > 0$ ein Algorithmus, der entweder Nein ausgibt, sobald das erste Mal Nein vorkommt,

oder der nach $N = \lceil \epsilon^{-1} \cdot \ln(\delta^{-1}) \rceil$ Versuchen Ja ausgibt

dann gilt:

$\Pr[A_\delta(I) = \text{Ja}] = 1$ für alle Ja-Instanzen I

$\Pr[A_\delta(I) = \text{Nein}] \geq 1 - \delta$ für alle Nein-Instanzen I

Monte-Carlo - Zweiseitiger Fehler:

$\Pr[A(I) \text{ ist korrekt}] \geq 0.5 + \epsilon$ für alle Instanzen I

A_δ gibt die meiste Antwort aus nach N Wiederholungen

$\implies$

Für $\delta > 0$, wenn $N = \lceil 4 \cdot \epsilon^{-2} \cdot \ln(\delta^{-1}) \rceil$

$\Pr[A_\delta(I) \text{ ist falsch}] \leq \delta$ für alle Instanzen I

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Quicksort/Quickselect

Quicksort: sortiert den Array
erwartete Laufzeit: $\mathcal{O}(n \log n)$

```
QuickSort(A, ℓ, r)
if ℓ < r then
    p ← Uniform({ℓ, ℓ + 1, ..., r})          ▷ wähle Pivotelement zufällig
    t ← Partition(A, ℓ, r, p)
    QuickSort(A, ℓ, t − 1)
    QuickSort(A, t + 1, r)
```

Quickselect: gibt das k-kleinste Element aus
erwartete Laufzeit: $\mathcal{O}(n)$

```
QuickSelect(A, ℓ, r, k)
p ← Uniform({ℓ, ℓ + 1, ..., r})                  ▷ wähle Pivotelement zufällig
t ← Partition(A, ℓ, r, p)
if t = ℓ + k − 1 then
    return A[t]                                  ▷ gesuchtes Element ist gefunden
else if t > ℓ + k − 1 then
    return QuickSelect(A, ℓ, t − 1, k)           ▷ gesuchtes Element ist links
else
    return QuickSelect(A, t + 1, r, k − t)       ▷ gesuchtes Element ist rechts
```

Target Shooting

Gegeben: endliche Mengen S, U s.d. $S \subseteq U$,

$$I_S : U \to \{0,1\} : I_S(u) = 1 \iff u \in S$$

Gesucht: $|S|/|U|$

Algorithmus

1) wähle $u_1, \ldots, u_N$ aus U zufällig unabhängig und gleichverteilt

2) **return** $Y := \frac{1}{N} \sum_{i=1}^{N} I_S(u_i) \qquad \Longrightarrow \qquad \mathbb{E}[Y] = \frac{|S|}{|U|}$

Satz: (geeignetes N finden)
Für $\delta, \epsilon > 0$:

$$N \geq 3 \frac{|U|}{|S|} \epsilon^{-2} \ln\left(\frac{2}{\delta}\right) \qquad \Longrightarrow \qquad \Pr\left[\left|Y - \mathbb{E}[Y]\right| \geq \epsilon \cdot \mathbb{E}[Y]\right] \leq \delta$$

(Kann man mit Chernoffs Ungleichungen i) und ii) zeigen)

Primzahltest

A. GGT

$\gcd(a, n) > 1$ für $1 \leq a \leq n-1$

$\implies n$ nicht prim

i) "nicht prim" immer richtig

ii) $\Pr[\text{"prim"} \mid \text{nicht prim}] = \dfrac{|Z_n^*|}{n-1}$

$\implies$ Verbesserung durch (viel) Wiederholung

iii) cost$(\gcd(m, n)) = \mathcal{O}((\log nm)^3)$

B. Fermat's little theorem

n ist prim

$\implies \forall a \in [n-1] : a^{n-1} \equiv 1 \mod n$

i) "nicht prim" immer richtig

ii) $\Pr[\text{"prim"} \mid \text{nicht prim}] = \dfrac{|PB_n|}{n-1}$

iii) $PB_n := \{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv 1 \text{ mod } n\}$

iv) Carmichael-Zahl n:

1) n ist nicht prim
2) $PB_n = \mathbb{Z}_n^*$

v) $\Pr[\text{"prim"} \mid \text{nicht prim}] < 0.5$,

falls n nicht Carmichael

$\implies$ Verbesserung durch Wiederholung

vi) cost$(a^{n-1} \mod n) = \mathcal{O}((\log n)^3)$

C. Miller-Rabin

Miller-Rabin-PrimeTest(n)

1) Wähle $1 \leq a \leq n-1$ gleichverteilt zufällig
2) $d, k \in \mathbb{N}$ s.d. $n - 1 = 2^k d$, wobei d ungerade
3) **if** $a^d \mod n \neq 1$ **&&**
   $\nexists i < k : a^{2^i d} \mod n = n-1$ **then**
   return "nicht prim"
4) **else return** "prim"

i) "nicht prim" immer richtig

ii) $\Pr[\text{"prim"} \mid \text{nicht prim}] \leq \dfrac{1}{4}$

$\implies$ Verbesserung durch Wiederholung

iii) Laufzeit: $\mathcal{O}(\text{poly}(\log n))$

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
    1.1. Quicksort/Quickselect
    1.2. Target Shooting
    1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Bunte Pfade

Gegeben: ein Graph G und eine Färbung $\gamma : V \to [k]$

die Färbung muss nicht auf die Kanten achten

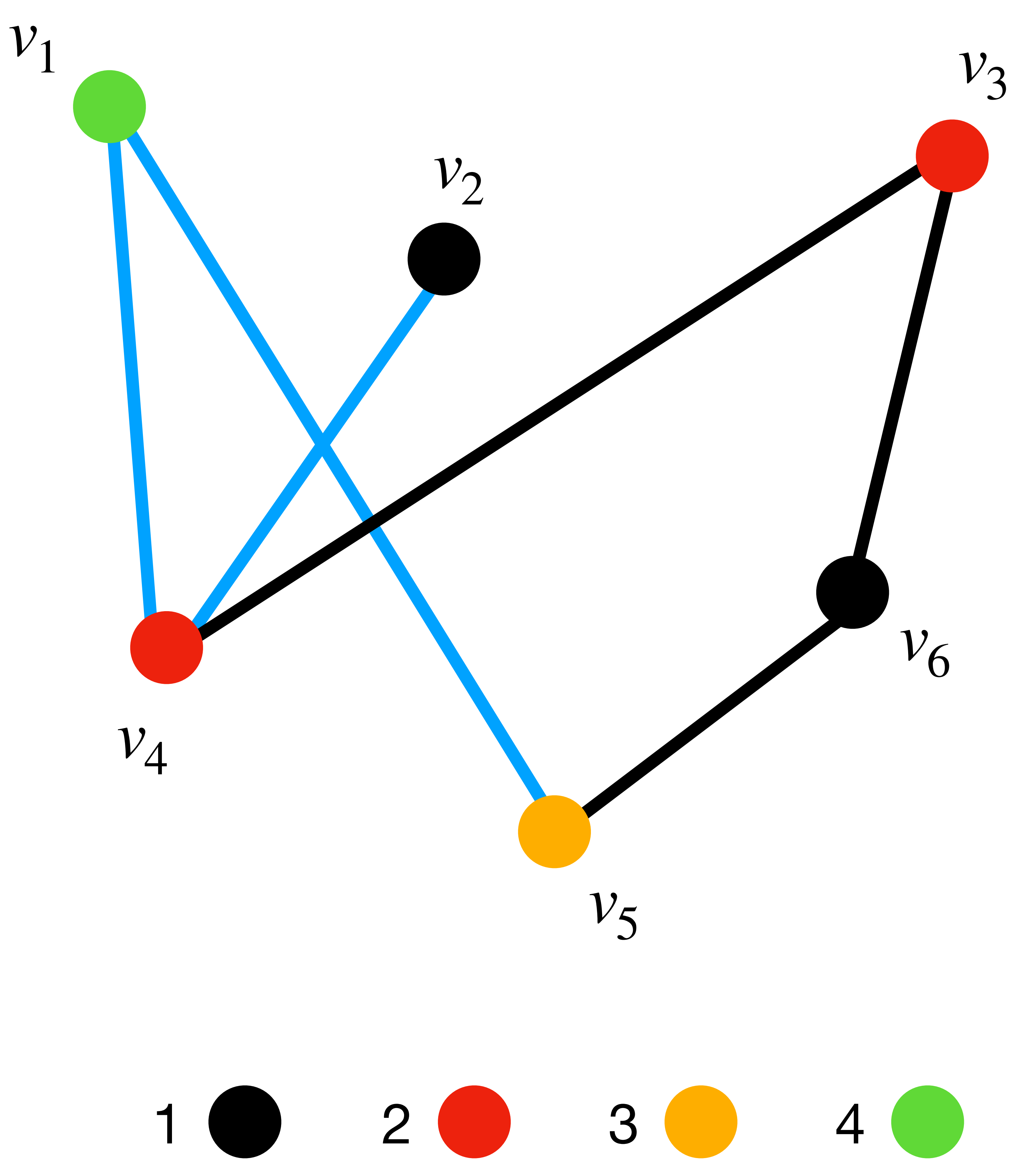
Gesucht: ob G einen <u>bunten</u> Pfad der Länge $k-1$ enthält

Bunter Pfad = alle Knoten haben verschiedene Farben

$P_i(v)$:= alle Bunte Pfade der Länge i, die in v enden

$$P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists \text{bunter Pfad, der in } v \text{ endet und genau mit } S \text{ gefärbt ist}\}$$

Gesucht: ob $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$



$n = 6$

$k = 4$

$P_3(v_5) = \{\{1,2,4,3\}\}$

Bunte Pfade

<u>$\mathrm{Bunt}(G, i)$</u>

for all $v \in V$ **do**

  $P_i(v) \leftarrow \emptyset$

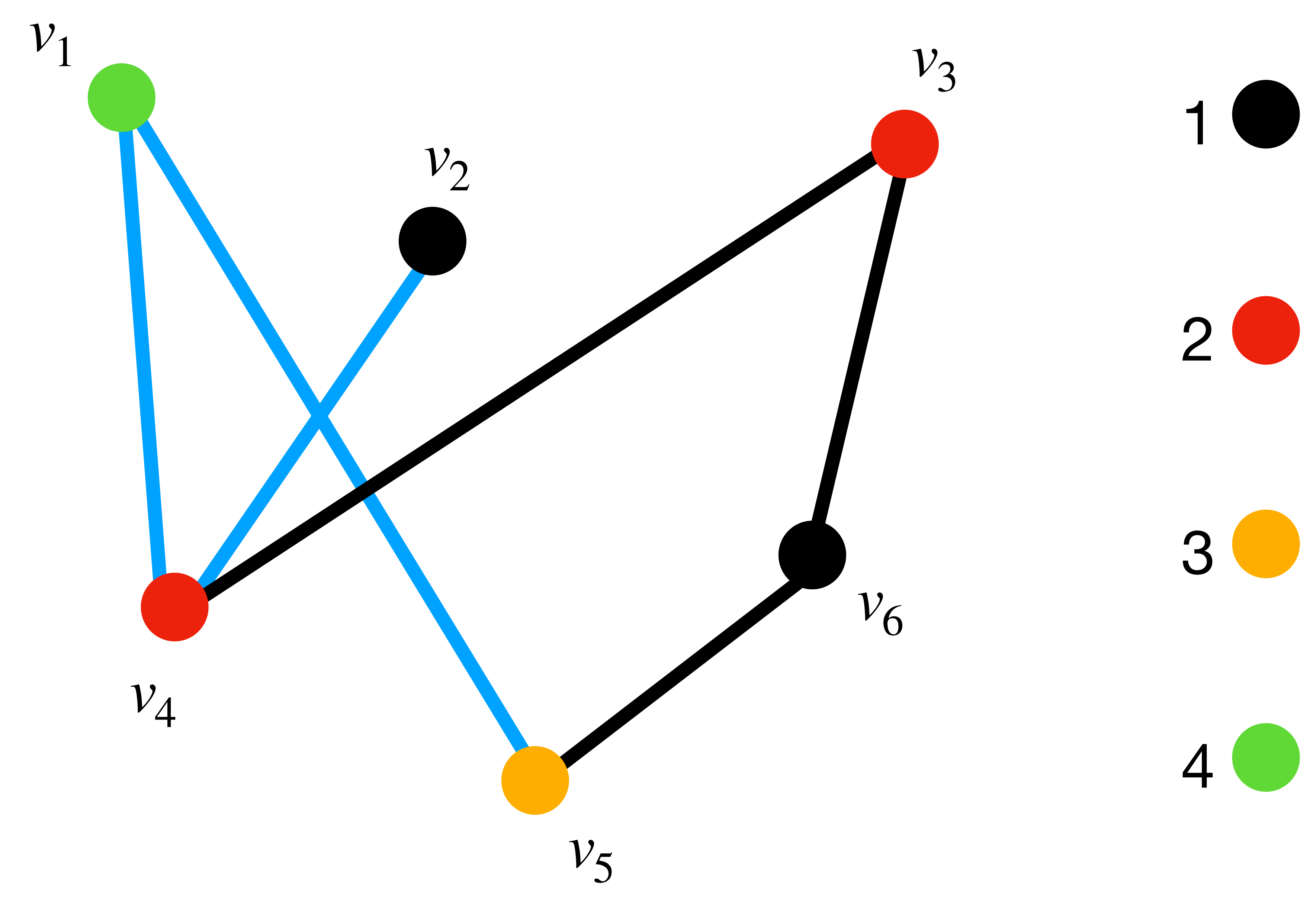
  for all $x \in N(v)$ **do**

    for all $R \in P_{i-1}(x)$ mit $\gamma(v) \notin R$ **do**

      $P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

<u>Beschreibung:</u> findet alle bunte Pfade der Länge i, basierend auf langen Pfaden der Länge $i-1$

<u>Laufzeit:</u> $\mathcal{O}\left(\sum_{v \in V} \deg(v) \cdot \binom{k}{i} \cdot i\right) = \mathcal{O}\left(m \cdot \binom{k}{i} \cdot i\right)$



$P_0(v_1) = \{\{4\}\}$ $P_0(v_2) = \{\{1\}\}$ $P_0(v_3) = \{\{2\}\}$ …

$P_1(v_4) = \{\{4{,}2\}, \{1{,}2\}\}$ …

$P_2(v_1) = \{\{1{,}2{,}4\}, \{1{,}3{,}4\}\}$ …

$P_3(v_5) = \{\{1{,}2{,}4{,}3\}\}$ …

Bunte Pfade

<u>Bunt(G, i)</u>

for all $v \in V$ **do**

$\quad P_i(v) \leftarrow \emptyset$

$\quad$ **for all** $x \in N(v)$ **do**

$\quad\quad$ **for all** $R \in P_{i-1}(x)$ mit $\gamma(v) \notin R$ **do**

$\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

<u>Beschreibung:</u> findet alle bunte Pfade der Länge i, basierend auf langen Pfaden der Länge $i - 1$

<u>Laufzeit:</u> $\mathcal{O}\left(\sum_{v \in V} \deg(v) \cdot \binom{k}{i} \cdot i\right) = \mathcal{O}\left(m \cdot \binom{k}{i} \cdot i\right)$

<u>Regenbogen(G, γ)</u>

for all $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$

for $i = 1,\ldots,k-1$ **do** $\text{Bunt}(G, i)$

return $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$

<u>Beschreibung:</u> findet, ob es einen bunten Pfad der Länge $k - 1$ gibt

<u>Laufzeit:</u> $\mathcal{O}\left(|V| + \sum_{i=1}^{k-1}\left(m \cdot \binom{k}{i} \cdot i\right)\right) = \mathcal{O}(2^k km)$

Lange Pfade

<u>**Long Path Problem**</u>

Gegeben: ein Graph G und $k \in \mathbb{N}_0$

Gesucht: ob G einen Pfad der Länge k enthält

→ <u>NP-schwer</u> (Reduktion von HamiltonPfad-Problem)

Lange Pfade → Bunte Pfade

Für $\mathrm{Longpath}(G, k)$ lösen wir $\mathrm{Regenbogen}(G, \gamma')$

wobei $\gamma' : V \to [k+1]$ ist eine gleichverteilt zufällige Färbung mit $k+1$ Farben

Ein Versuch

Laufzeit: $\mathcal{O}(2^k km)$

$p_{erfolg} \geq e^{-k}$

$\lceil \lambda e^k \rceil$ **Versuche**

Laufzeit: $\mathcal{O}(\lambda e^k \cdot 2^k km)$

$p_{erfolg} \geq 1 - e^{-\lambda}$

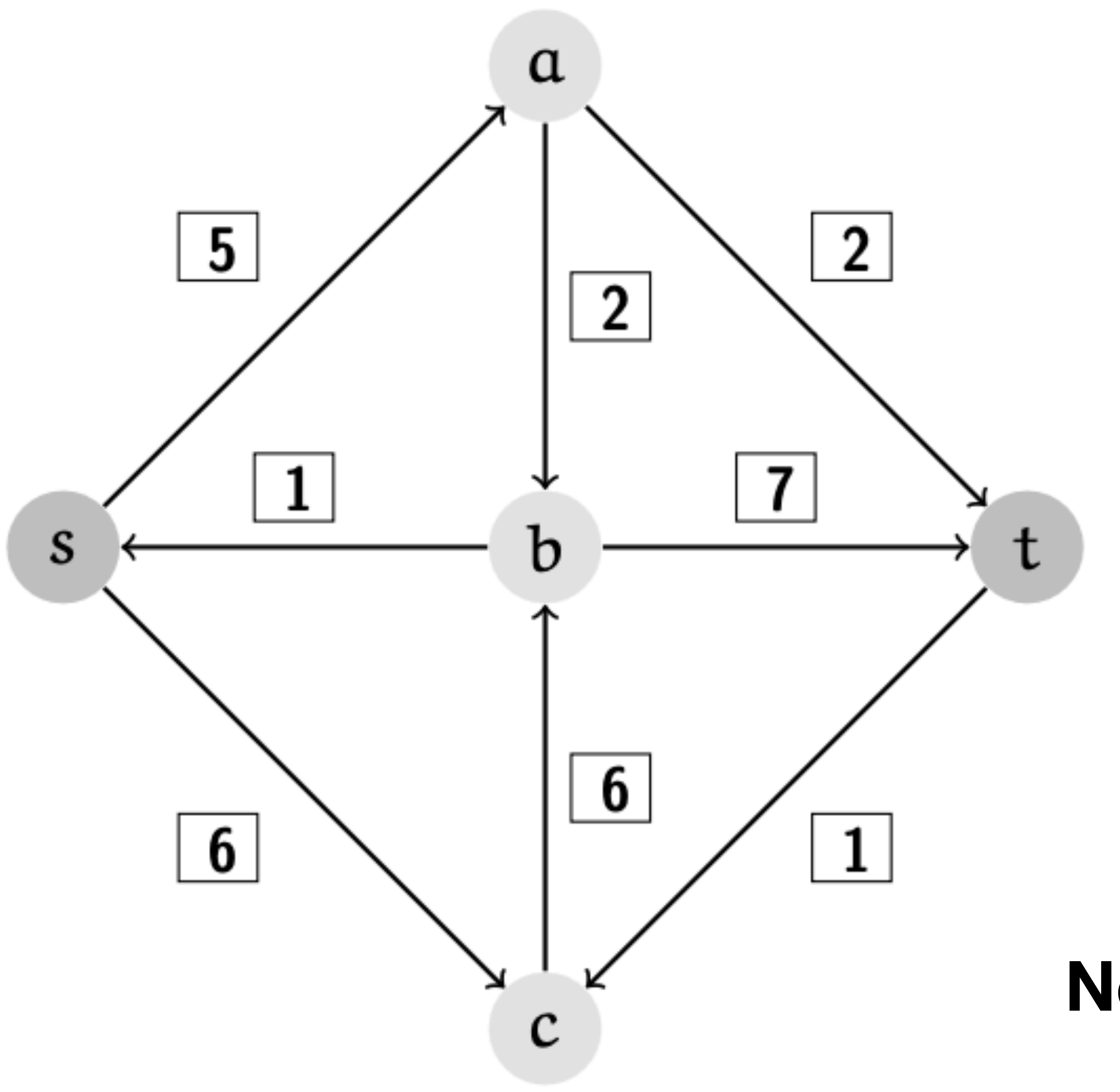
Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Netzwerke und Flüsse

<u>Netzwerk</u> $N := (V, A, c, s, t)$
- (V, A) ist ein <u>gerichteter Graph</u>
- $s \in V$ ist die <u>Quelle</u>
- $t \in V$ ist die <u>Senke</u>
- $c : A \to \mathbb{R}_0^+$ die <u>Kapazitätsfunktion</u>

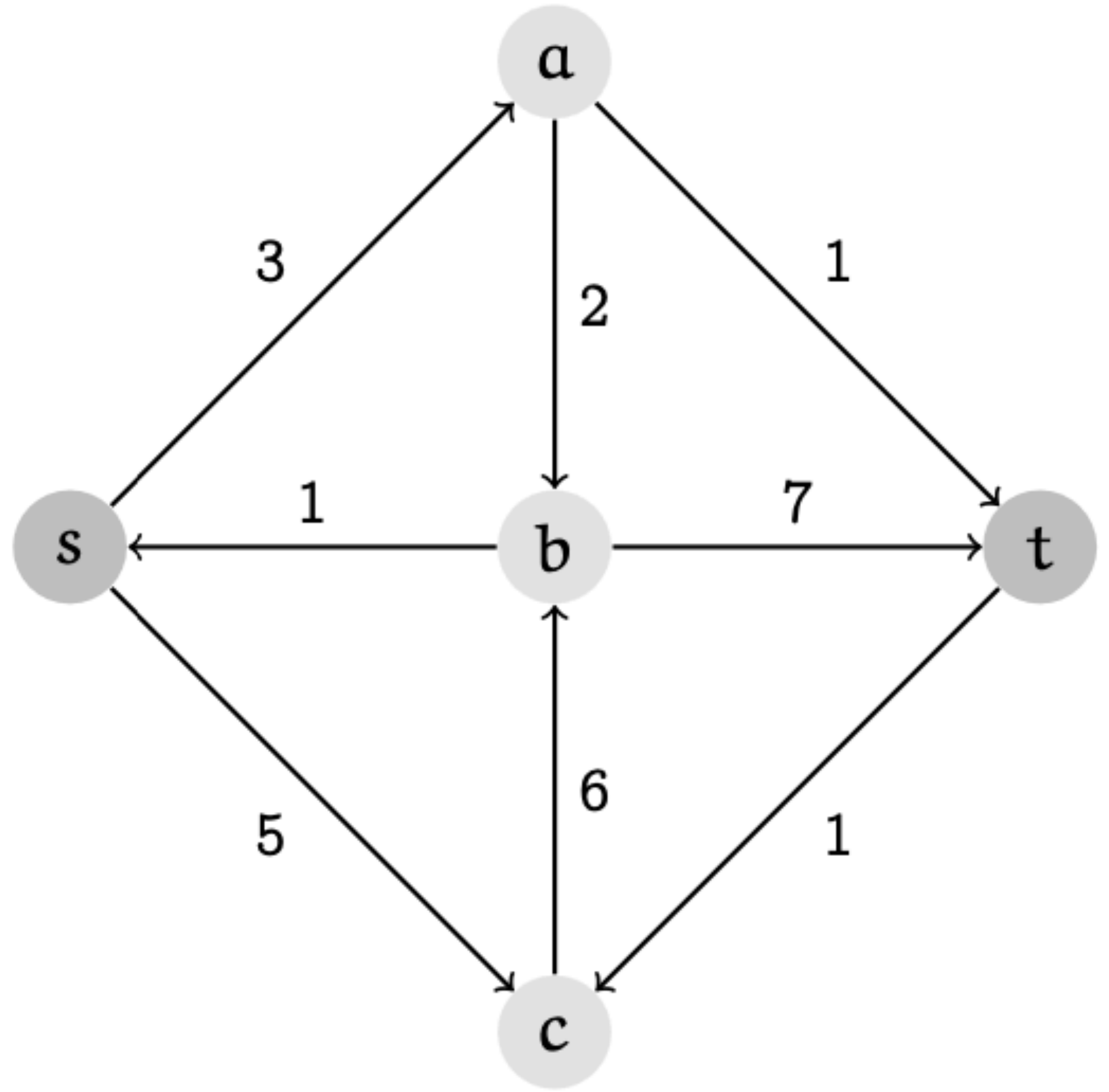


Netzwerk

<u>Fluss</u> f **in** $N : f : A \to \mathbb{R}_0^+$
- **Zulässigkeit**: $0 \leq f(e) \leq c(e)$ für alle $e \in A$
- **Flusserhaltung**: Für alle $v \in V \backslash \{s, t\}$ gilt

$$\sum_{u \in V:(u,v) \in A} f(u, v) = \sum_{u \in V:(v,u) \in A} f(v, u)$$

- **val**(f) := **netoutflow**(s) := $\displaystyle\sum_{u \in V:(s,u) \in A} f(s, u) - \sum_{u \in V:(u,s) \in A} f(u, s)$
- **val**(f) := **netinflow**(t) := $\displaystyle\sum_{u \in V:(u,t) \in A} f(u, t) - \sum_{u \in V:(t,u) \in A} f(t, u)$

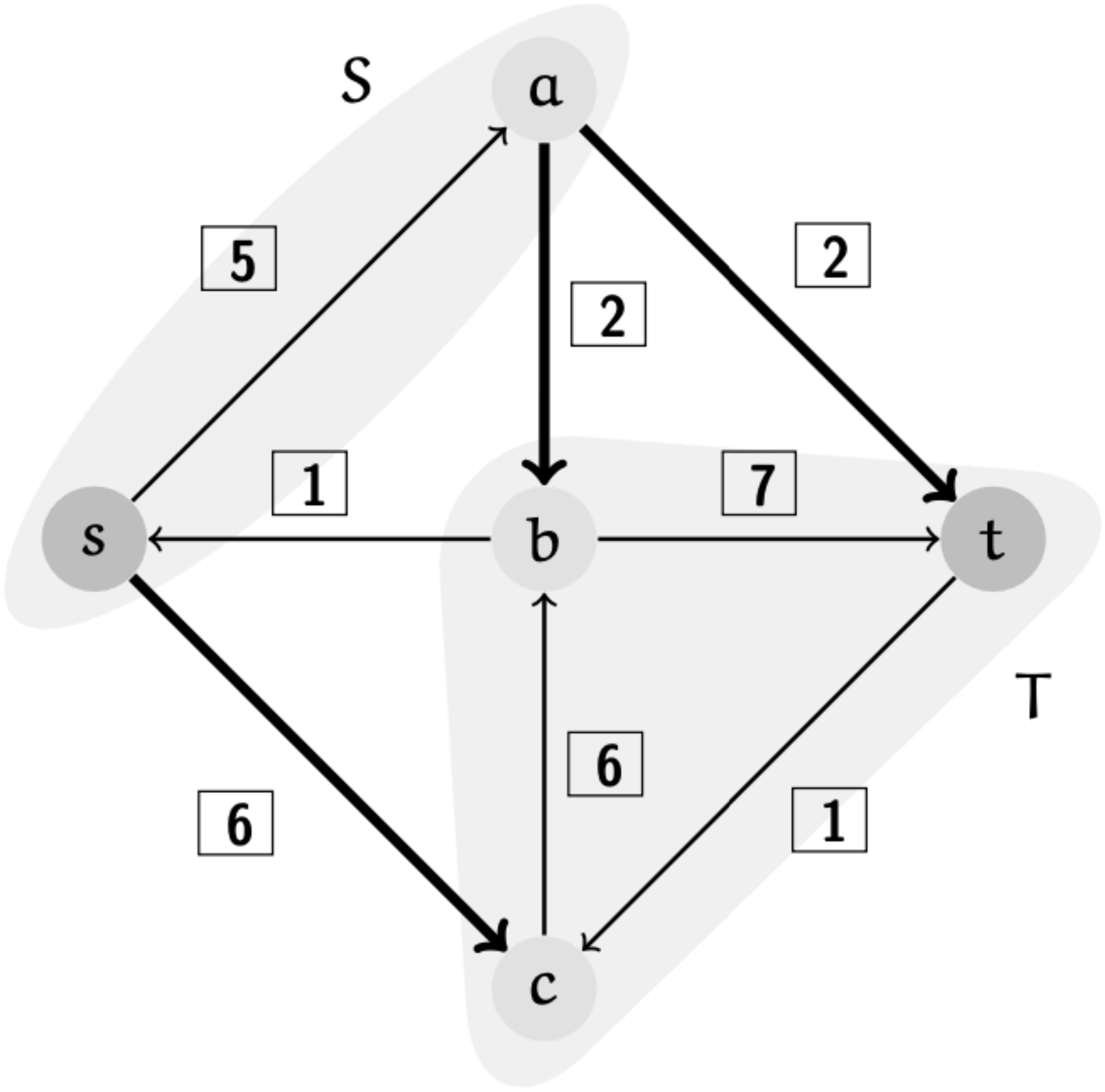


Fluss

Schnitte

s-t-Schnitt in $N := (V, A, c, s, t)$

- eine Partition von V: (S, T)
- $s \in S, t \in T$
- cap$(S, T) = \sum_{(u,w) \in (S \times T) \cap A} c(u, w)$



cap$(S, T) = 2 + 2 + 6 = 10$

Flüsse und Schnitte

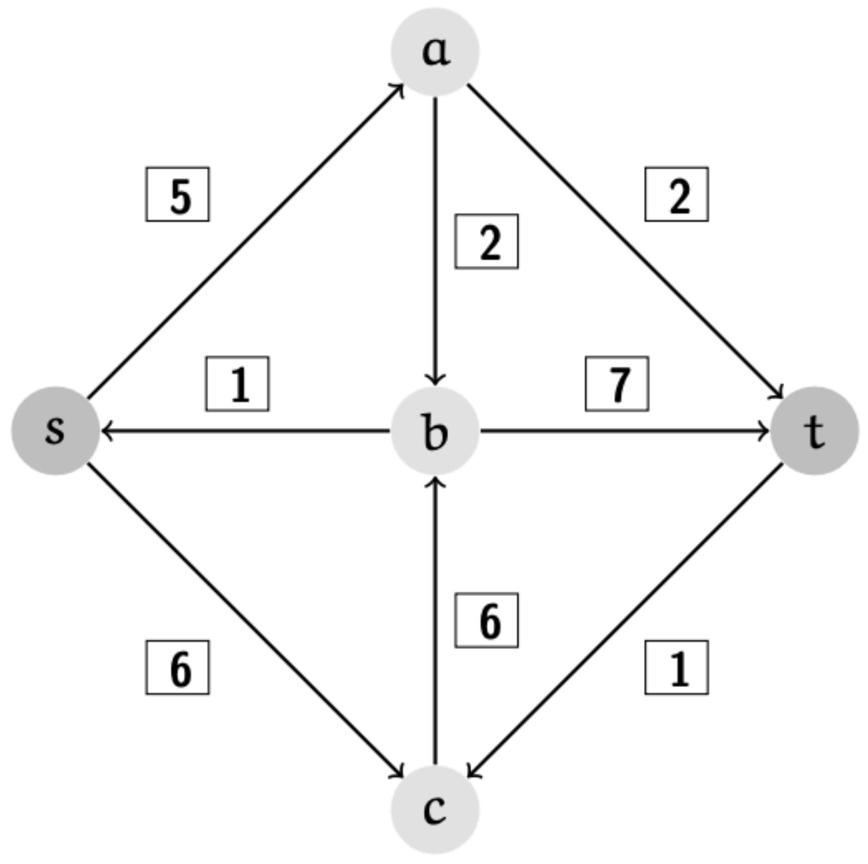
- für einen Fluss f und einen Schnitt (S, T) : **val**$(f) \leq$ **cap**(S, T)
- für einen Fluss f: $\exists (S, T)$: val$(f) =$ cap$(S, T) \implies f$ is maximal
- **MaxFlowMinCut-Theorem:**

$$\max_f \text{val}(f) = \min_{(S,T)} \text{cap}(S, T)$$

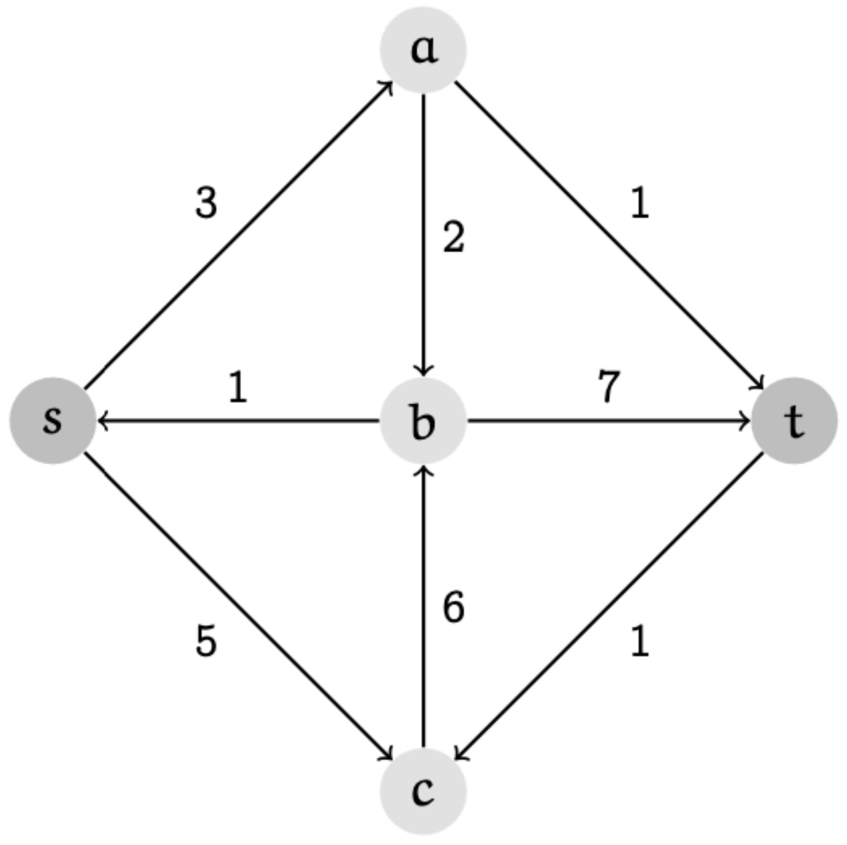
Restnetzwerk

<u>**Restnetzwerk**</u> $N_f = (V, A_f, r_f, s, t)$ **für** N, f

- N hat keine entgegen gerichtete Kanten
- $\forall e \in A : f(e) < c(e) \Longrightarrow e \in A_f \land r_f(e) = c(e) - f(e)$
- $\forall e \in A : f(e) > 0 \Longrightarrow e^{opp} \in A_f \land r_f(e^{opp}) = f(e)$
- A_f hat keine weiteren Kanten



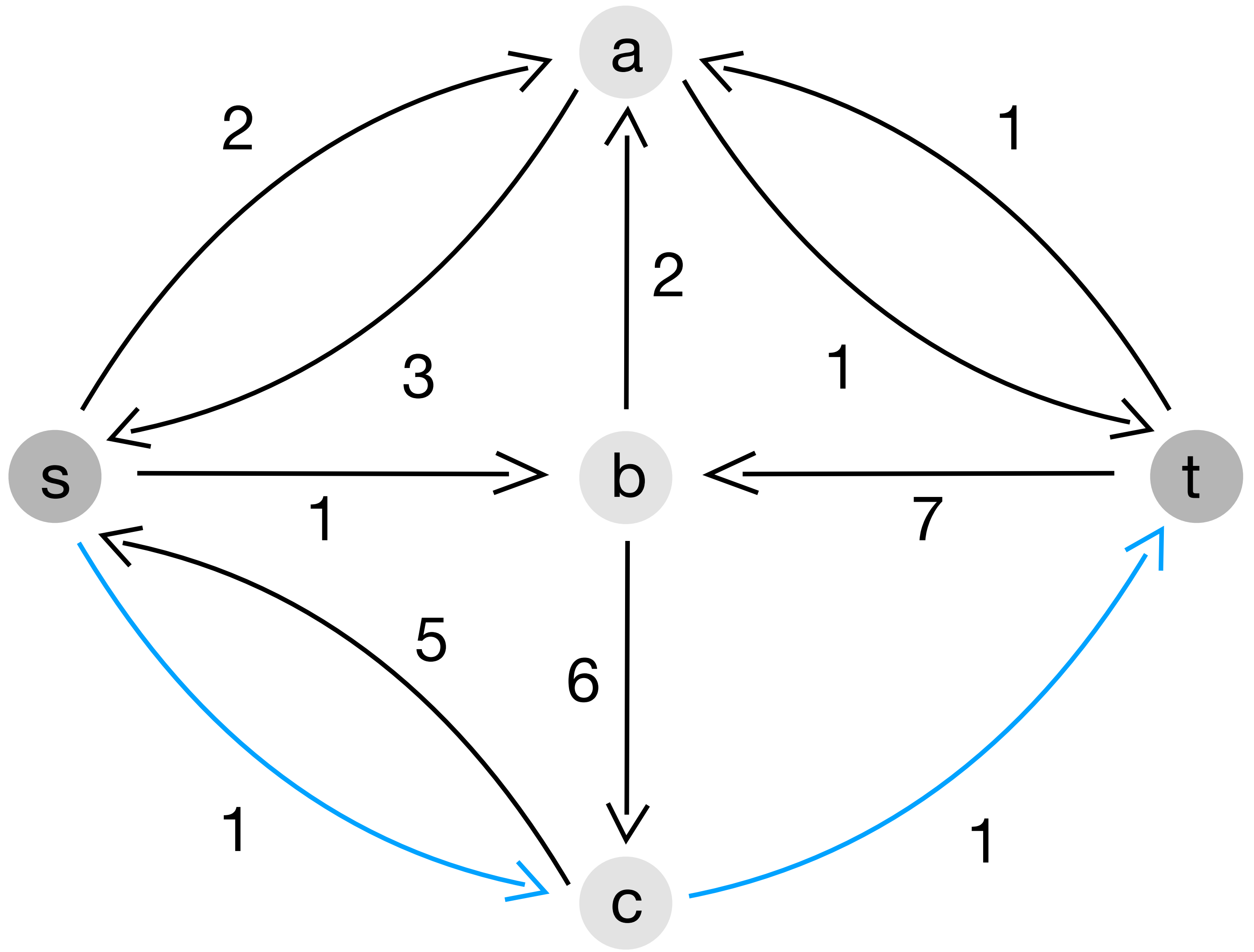
Netzwerk



Fluss

Restnetzwerk und MaxFlow

f ist ein <u>maximaler Fluss</u> $\iff$ $\neg\exists$<u>gerichteter s-t-Pfad</u> in N_f



Algorithmen

Satz für Ford-Fulkerson Algorithmus

Für N mit $c : A \to \mathbb{N}_0^{\leq U}$ gilt:

1) es gibt einen ganzzahligen maximalen Fluss

2) der Algo findet den Fluss in $\mathcal{O}(mnU)$

Ford-Fulkerson Algorithmus

1) $f \leftarrow 0$

2) **while** $\exists$s-t-Pfad P in N_f **do**

3) **Augmentiere** den Fluss entlang P

4) **return** f

→ kann unendlich laufen wenn $c : A \to \mathbb{R}$

→ läuft immer endlich wenn $c : A \to \mathbb{N}_0$

→ Laufzeit: $\mathcal{O}(mnU)$ wobei $c : A \to \mathbb{N}_0$ und $U = \max_{e \in A} c(e)$

Andere Methode/Algorithmen

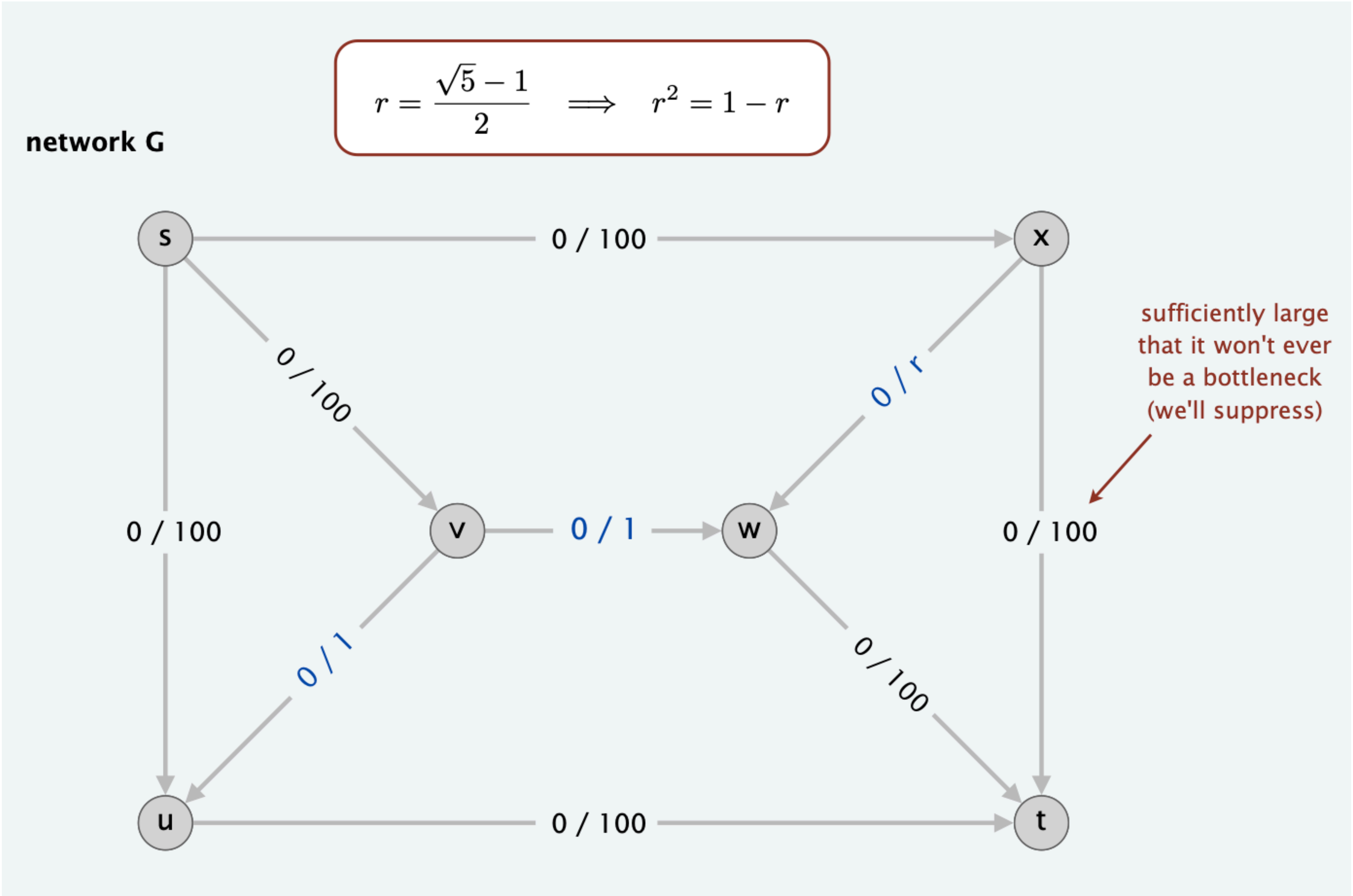
1) Capacity Scaling (Kapazitäten ganzzahlig + höchstens U)

$\mathcal{O}(mn(1 + \log U))$

2) Dynamic Trees

$\mathcal{O}(mn \log n)$

Pathological Example:



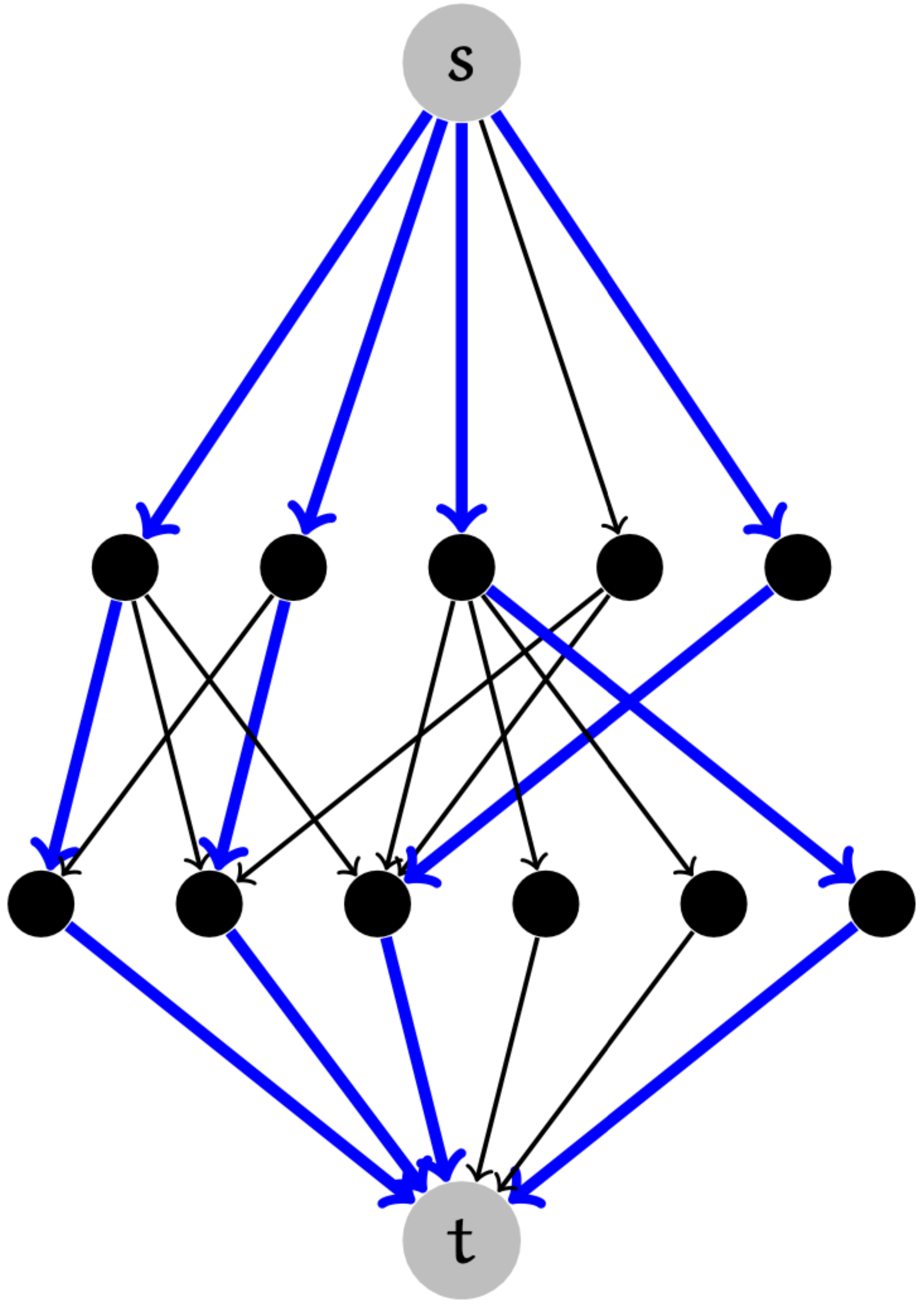
https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07DemoFordFulkersonPathological.pdf

Anwendungen der Flüsse

1. Matchings in bipartiten Graphen

$G = (A \uplus B, E)$
$\mapsto N_G = (V', E', 1, s, t)$
mit $V' = A \uplus B \uplus \{s, t\}$
und $E' = \{s\} \times A \quad \cup \quad \{(a, b) \in A \times B \mid \{a, b\} \in E\} \quad \cup \quad B \times \{t\}$

$$\max_{M \text{ Matching in } G} |M| = \max_{f \text{ Fluss in } N_G} \text{val}(f)$$



Anwendungen der Flüsse

2. Kantendisjunkte Pfade

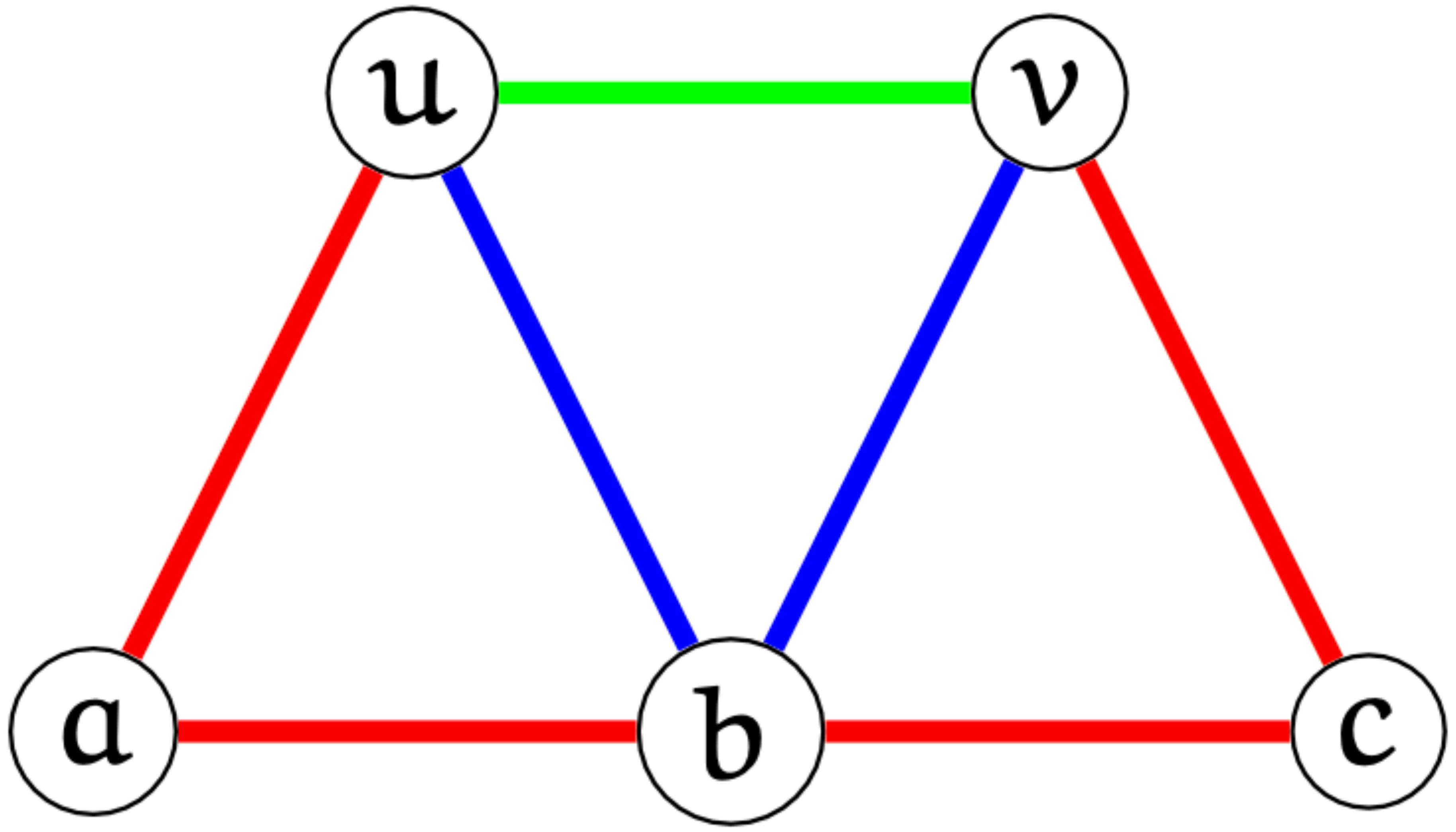
$G = (V, E), u, v \in V$

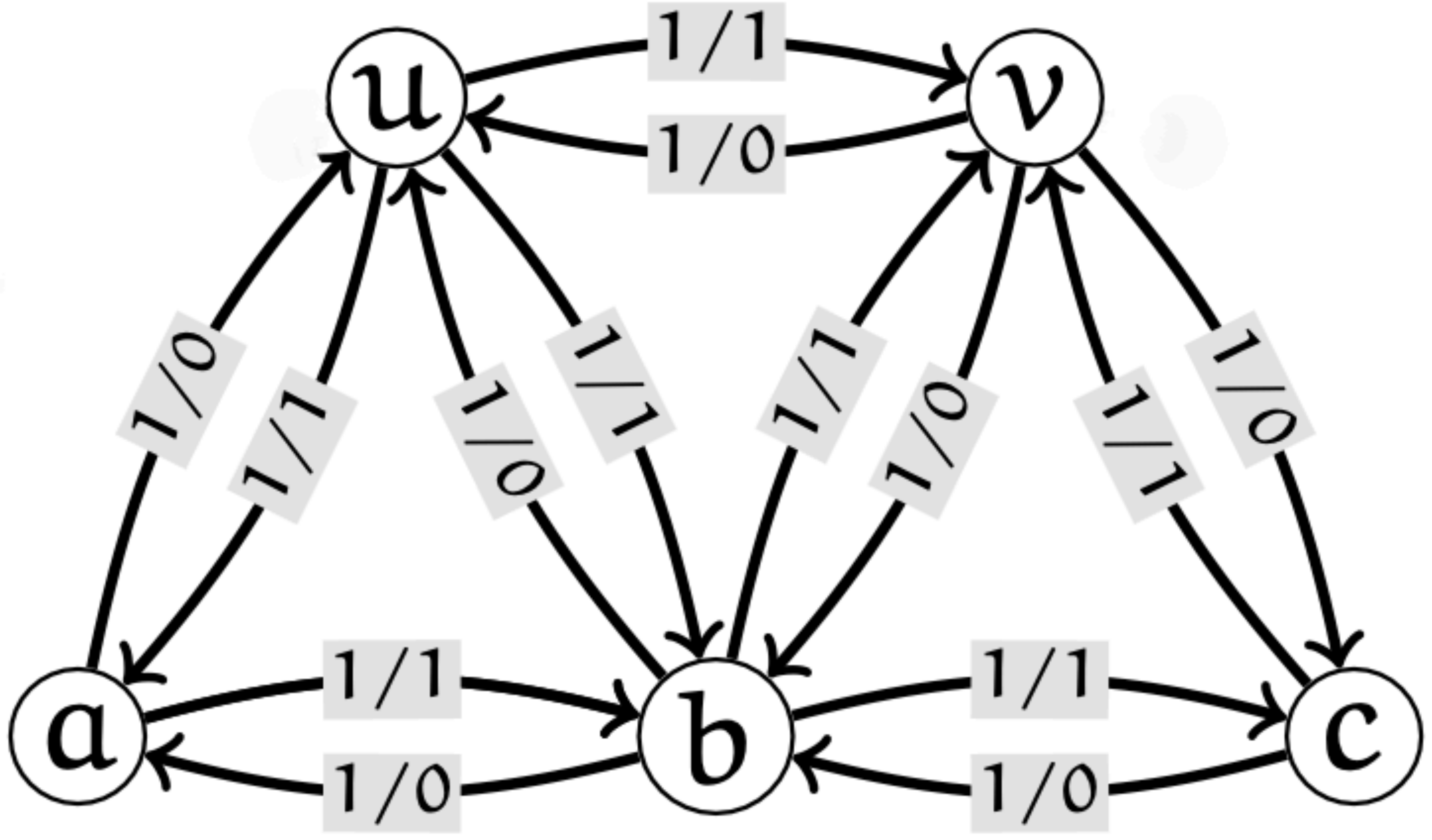
$\mapsto N_G = (V, \{(x, y), (y, x) \mid \{x, y\} \in E\}, 1, u, v)$

1. Finde ganzzahligen maximalen Fluss
2. Starte bei u und finde einen Pfad nach v durch noch nicht besuchte Kanten mit Fluss 1.
3. Markiere die Kanten besucht
4. Wiederhole val(f) mal

$$\max_{f \text{ Fluss in } N_G} \text{val}(f) = \max \text{\#Kantendisjunkter Pfade zwischen } u \text{ und } v \text{ in } G = \min \text{\#Kanten, die } u \text{ und } v \text{ trennen}$$

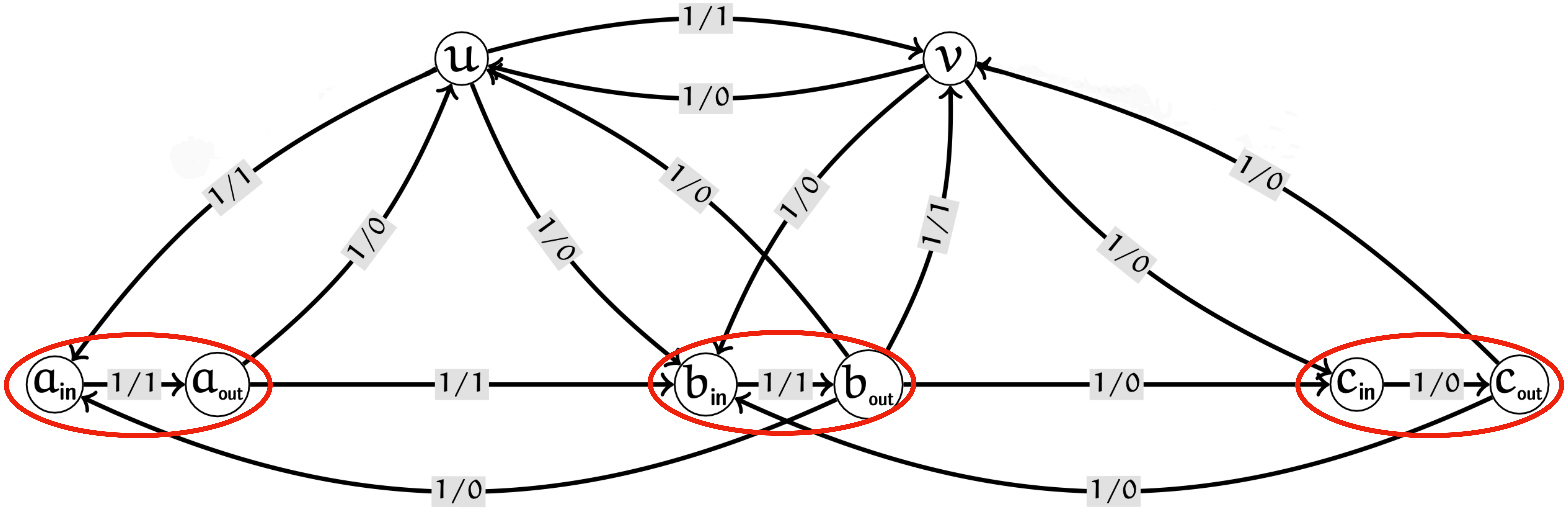
Anwendungen der Flüsse





Anwendungen der Flüsse

Knoten - disjunkte Pfade:



Anwendungen der Flüsse

3. Bildsegmentierung

Bild: ein Graph $G = (P, E)$ mit $\chi : P \to$ Farben

$\alpha : P \to \mathbb{R}_0^+$ α_p größer $\Longrightarrow$ eher im Vordergrund

$\beta : P \to \mathbb{R}_0^+$ β_p größer $\Longrightarrow$ eher im Hintergrund

$\gamma : E \to \mathbb{R}_0^+$ γ_e größer $\Longrightarrow$ eher im gleichen Teil

Qualitätsfunktion: $q(A, B) := \sum_{p \in A} \alpha_p + \sum_{p \in B} \beta_p - \sum_{e \in E, |e \cap A| = 1} \gamma_e$

Gesucht: eine Vorder-/Hintergrundspartition (A, B) die $q(A, B)$ maximiert

Anwendungen der Flüsse

$$
\begin{aligned}
\max q(A,B) &= \max\Big(\sum_{p\in A}\alpha_p + \sum_{p\in B}\beta_p - \sum_{e\in E,|e\cap A|=1}\gamma_e\Big)\\
&= \max\Big(\sum_{p\in P}(\alpha_p+\beta_p) - \sum_{p\in A}\beta_p - \sum_{p\in B}\alpha_p - \sum_{e\in E,|e\cap A|=1}\gamma_e\Big)\\
&= \sum_{p\in P}(\alpha_p+\beta_p) - \min\Big(\sum_{p\in A}\beta_p + \sum_{p\in B}\alpha_p + \sum_{e\in E,|e\cap A|=1}\gamma_e\Big)\\
&=: \sum_{p\in P}(\alpha_p+\beta_p) - \min q'(A,B)
\end{aligned}
$$

Anwendungen der Flüsse

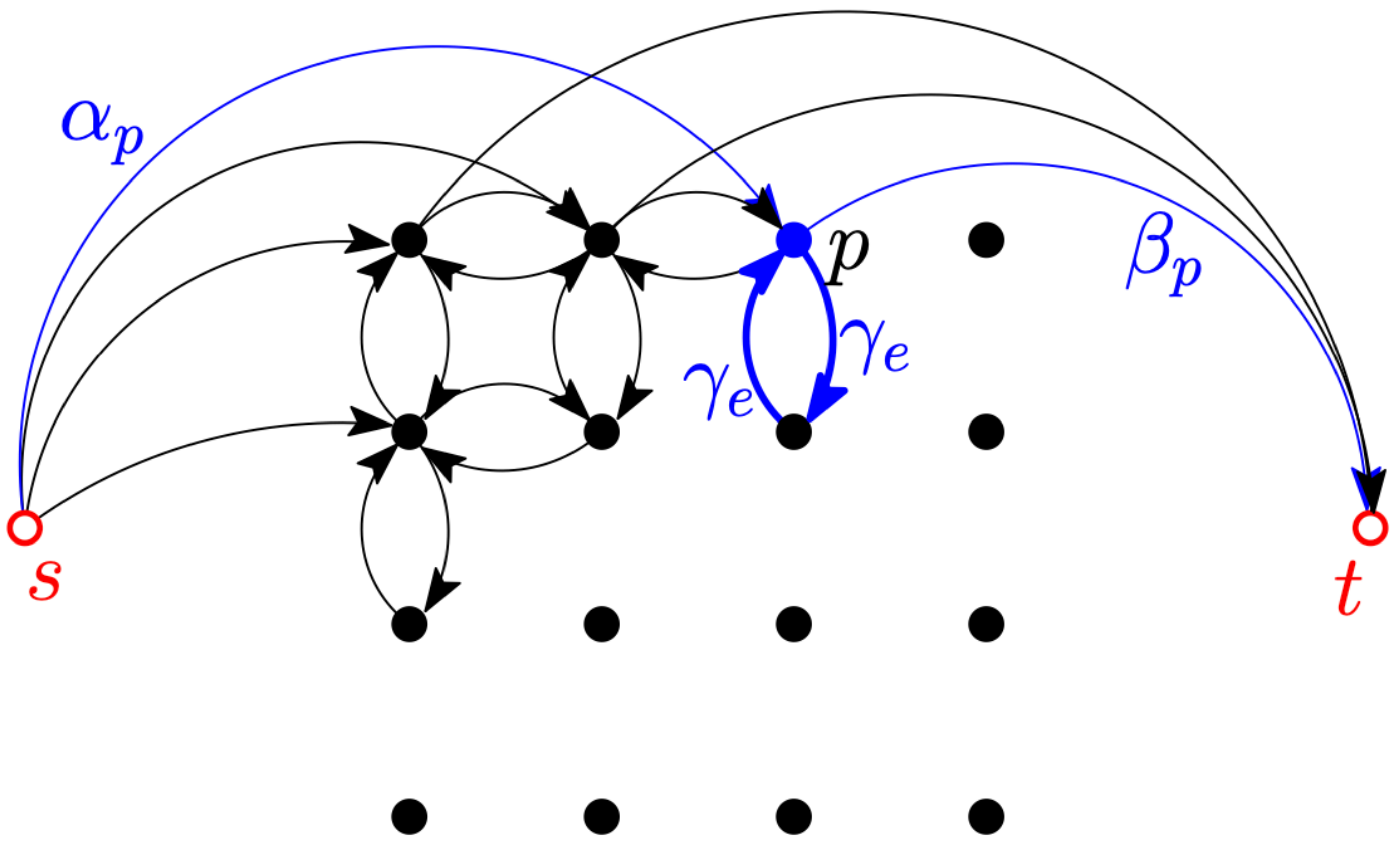
3. Bildsegmentierung

Bild: ein Graph $G = (P, E)$ mit $\chi : P \to$ Farben

$\alpha : P \to \mathbb{R}_0^+$ $\quad \alpha_p$ größer $\Longrightarrow$ eher im Vordergrund

$\beta : P \to \mathbb{R}_0^+$ $\quad \beta_p$ größer $\Longrightarrow$ eher im Hintergrund

$\gamma : E \to \mathbb{R}_0^+$ $\quad \gamma_e$ größer $\Longrightarrow$ eher im gleichen Teil



$\mapsto N = (P \cup \{s, t\}, \overrightarrow{E}, c, s, t)$

$\forall p \in P, \exists (s, p) \in \overrightarrow{E} : c(s, p) = \alpha_p$

$\forall p \in P, \exists (p, t) \in \overrightarrow{E} : c(p, t) = \beta_p$

$\forall e = \{p, p'\} \in E, \exists (p, p'), (p', p) \in \overrightarrow{E} : c(p, p') = c(p', p) = \gamma_e$

für $A := S \backslash \{s\}$ und $B := T \backslash \{t\}$, $q'(A, B) = \text{cap}(S, T)$

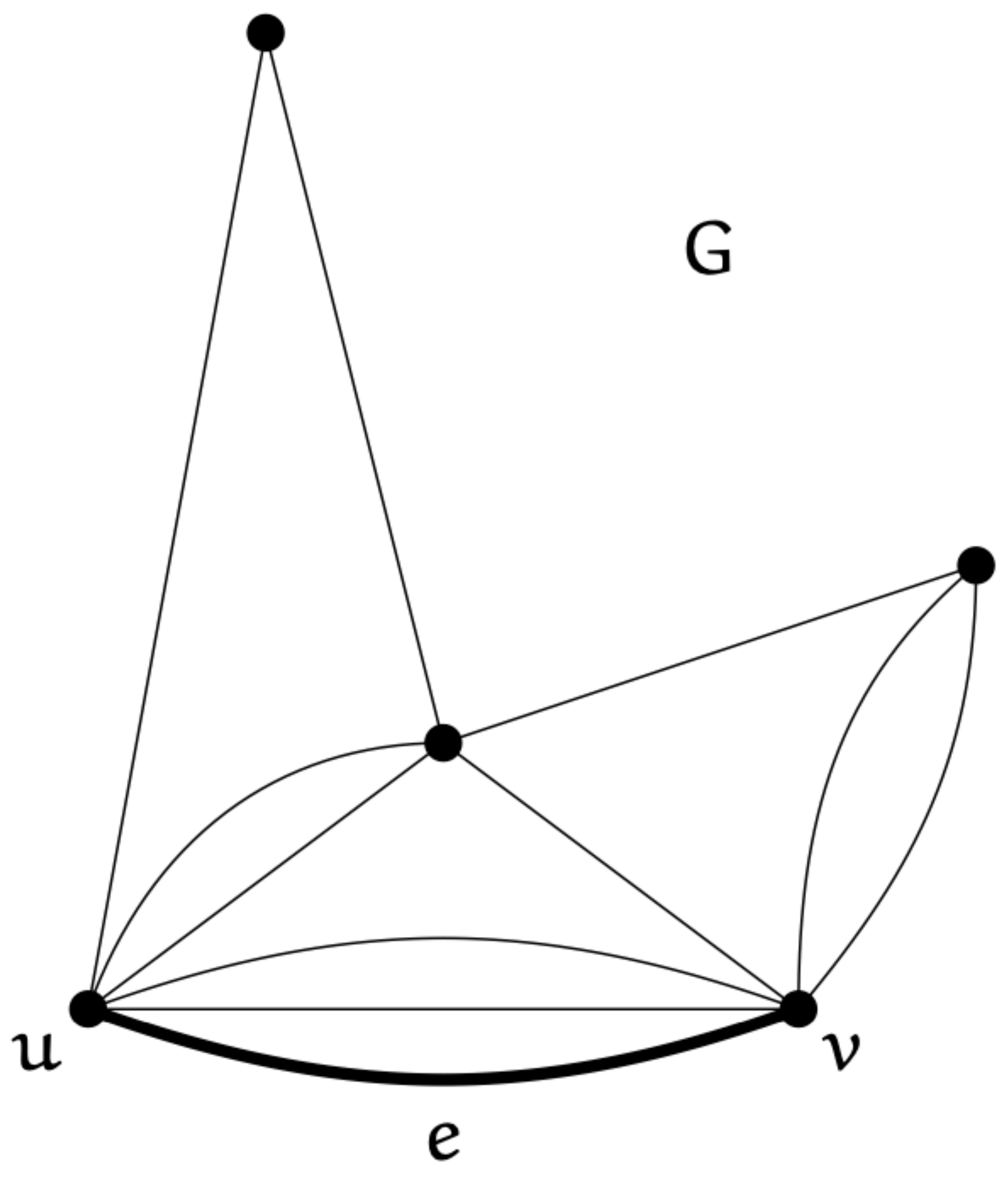
$\to$ Mithilfe Maxflow Mincut finden!

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Min-Cut Problem



<u>Gegeben:</u> Ein Multigraph G

<u>Gesucht</u>: $\mu(G) :=$ die Größe minimales Kantenschnitts

<u>Kantenschnitt:</u> Eine Kantenmenge C, s.d. $(V, E \backslash C)$ nicht zusammenhängend

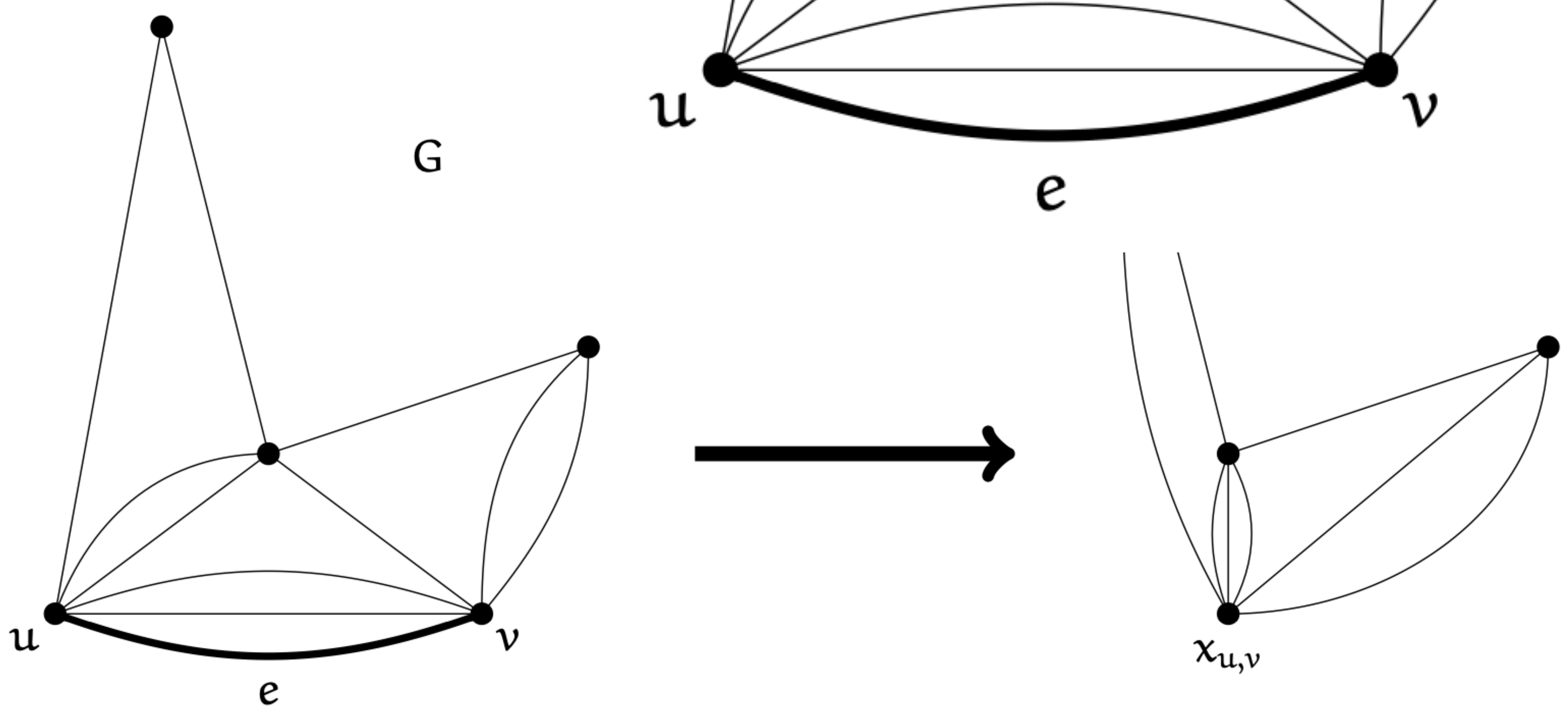
Ansatz 1: Mittels Flüsse (Dynamic trees)

$(n-1) \cdot \mathcal{O}(mn \log n) = \mathcal{O}(mn^2 \log n) = \mathcal{O}(n^4 \log n)$

Ansatz 2: Mittels Kantenkontraktion

<u>Kantenkontrakion:</u> Kontrahiere e von $G \rightarrow G/e$

- $\mu(G/e) \geq \mu(G)$
- $e \notin C \implies \mu(G/e) = \mu(G)$



Min-Cut Problem

<u>*Cut*(G)</u>

1) $G' \leftarrow G$
2) **while** $|V(G')| > 2$ **do**
3) $\quad e \leftarrow$ gleichverteilt zufällige Kante in G'
4) $\quad G' \leftarrow G'/e$
5) **return** Größe des eindeutigen Schnitts in G'

<u>**Laufzeit**</u>**:** $O(n^2)$ wobei $n = |V(G)|$

Sei $p(G)$ die Erfolgswahrscheinlichkeit, also

$$p(G) := \Pr[\text{Cut}(G) = \mu(G)]$$

$$p(n) := \min_{G=(V,E),|V|=n} p(G)$$

$$\rightarrow p(G) \geq p(|V(G)|)$$

<u>**Lemmas**</u>

1) Wenn e gleichverteilt zufällig ist, $\Pr[\mu(G) = \mu(G/e)] \geq 1 - \dfrac{2}{n}$

2) $\forall n \geq 3 : p(n) \geq \left(1 - \dfrac{2}{n}\right) \cdot p(n-1)$

3) $p(n) \geq \dfrac{n-2}{n} \cdot \dfrac{n-3}{n-1} \cdots \dfrac{1}{3} \cdot p(2) = \dfrac{2}{n(n-1)} = \dfrac{1}{\binom{n}{2}}$

Min-Cut Problem

<u>**Monte Carlo Wiederholungen von Cut**</u>

1) Wir wiederholen Cut(G) $\lambda \binom{n}{2}$ mal und nehmen den kleinsten Wert

2) <u>Laufzeit</u>: $\lambda \binom{n}{2} \cdot O(n^2) = O(\lambda n^4)$

3) <u>Fehlerwahrscheinlichkeit</u>: $\left(1 - \frac{1}{\binom{n}{2}}\right)^{\lambda \binom{n}{2}} \leq e^{-\lambda}$

4) Wenn $\lambda = \log n$, ist die Laufzeit $O(n^4 \log n)$ mit Fehlerw-keit $\leq 1/n$

Bootstrapping

n n-1 n-2 n-3 … … 4 3 2

n n-1 n-2 n-3 … … 4 3 2

$\lambda \binom{n}{2}$ mal

⋮

n n-1 n-2 n-3 … … 4 3 2

n n-1 n-2 n-3 … t … 4 3 2

t … 4 3 2

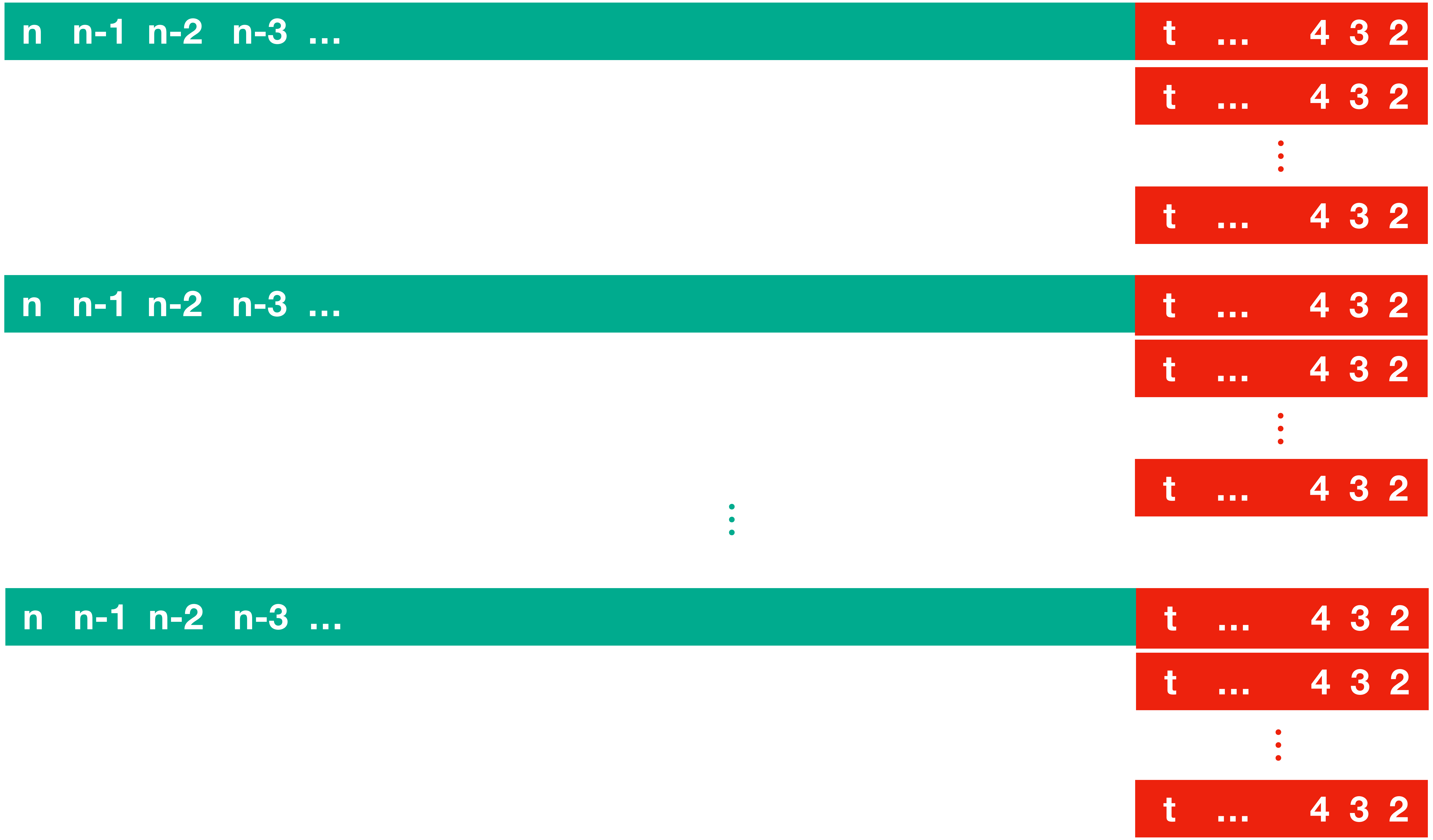
t^4 Algo

⋮

t … 4 3 2

$\rightarrow$ K.W-keit $\geq \dfrac{t(t-1)}{n(n-1)} \dfrac{e-1}{e}$

Bootstrapping



$\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1}$ **mal**

$\rightarrow$ **F.W-keit** $\leq e^{-\lambda}$

Laufzeit: $\overbrace{\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1}}^{\#\text{Wiederholungen}} \mathcal{O}(\underbrace{n(n-t)}_{\text{Reduktion auf } t \text{ Knoten}} + \overbrace{t^4}^{\text{Algo auf } t \text{ Knoten}}) = \mathcal{O}(\lambda(n^4/t^2 + n^2t^2)) \Longrightarrow_{\min} \mathcal{O}(\lambda n^3)$ when $t = \sqrt{n}$

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Kleinster umschliessender Kreis

Gegeben: eine endliche Punktenmenge $P \subseteq \mathbb{R}^2$
Gesucht: der Kreis des kleinsten Radius, der P umschliesst

Lemma: für jede Punktenmenge $P \subseteq \mathbb{R}^2, |P| \geq 3$ gibt es eine Teilmenge $Q \subseteq P$ s.d. $|Q| \leq 3 \wedge C(Q) = C(P)$
Wir verwenden: $T :=$ Anzahl der Runden/Durchläufe der Schleife

Randomized_PrimitiveVersion(P)

1) **wiederhole unendlich**
2) wähle $Q \subseteq P$ mit $|Q| = 3$ gleichverteilt zufällig
3) finde $C(Q)$
4) **if** $P \subseteq C^\bullet(Q)$
5) **return** $C(Q)$

Erwartete Laufzeit: $T \sim \mathrm{Geo}\left(1/\binom{n}{3}\right) \implies \mathcal{O}(n \cdot n^3)$

Randomized_CleverVersion(P)

1) $P' \leftarrow P$
2) **wiederhole unendlich**
3) wähle $Q \subseteq P'$ mit $|Q| = 11$ gleichverteilt zufällig
4) finde $C(Q)$
5) **if** $P \subseteq C^\bullet(Q)$
6) **return** $C(Q)$
7) **else** verdopple alle Punkte von P' ausserhalb von $C(Q)$

Erwartete Laufzeit: $\mathcal{O}(n \cdot T) = \mathcal{O}(n \cdot \log n)$

Lemma 3.28

Lemma 3.28. Sei P eine Menge von n (nicht unbedingt verschiedenen) Punkten und für $r \in \mathbb{N}$, R zufällig gleichverteilt aus $\binom{P}{r}$. Dann ist die erwartete Anzahl Punkte von P, die ausserhalb von $C(R)$ liegen, höchstens $3\frac{n-r}{r+1} \leq 3\frac{n}{r+1}$.

Für $p \in P$ **und** $R, Q \subseteq P$

- $\operatorname{out}(p, R) := \begin{cases} 1 & \text{falls } p \notin C^\bullet(R) \\ 0 & \text{sonst} \end{cases}$

$$\operatorname{out}(p, R) = 1 \iff \operatorname{essential}(p, R \cup \{p\}) = 1$$

- $\operatorname{essential}(p, Q) := \begin{cases} 1 & \text{falls } C(Q - p) \neq C(Q) \\ 0 & \text{sonst} \end{cases}$
- $Y :=$ Anzahl Punkte ausserhalb von $C(R)$

Beweis von Lemma 3.28

$$\mathbb{E}[Y] = \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{p \in P \setminus R} \operatorname{out}(p, R)$$

$$= \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{p \in P \setminus R} \operatorname{essential}(p, R \cup \{p\})$$

$$= \frac{1}{\binom{n}{r}} \sum_{Q \in \binom{P}{r+1}} \underbrace{\sum_{p \in Q} \operatorname{essential}(p, Q)}_{\leq 3}$$

$$\leq \frac{1}{\binom{n}{r}} \sum_{Q \in \binom{P}{r+1}} 3 = 3 \cdot \frac{\binom{n}{r+1}}{\binom{n}{r}} = 3 \cdot \frac{n-r}{r+1}$$

1. gehe über alle $r-$elementige Subsets durch und zähle die Punkte ausserhalb, am Ende bestimme den Durchschnitt
2. $\operatorname{out}(p, R) = 1 \iff \operatorname{essential}(p, R \cup \{p\}) = 1$
3. im Schritt davor haben wir uns sowieso schon Mengen mit $r+1$ Elementen angeschaut
4. es kann höchstens 3 essentielle Punkte geben letzte Umformungen sind einfach Arithmetik

Anzahl Punkte - Upper Bound

- $X_k :=$ Anzahl Punkte nach k Iterationen
- $X_0 = n$

$$\begin{aligned}
\mathbb{E}[X_k] &= \sum_{i=0}^{\infty} \mathbb{E}[X_k \,|\, X_{k-1} = i] \cdot \Pr[X_{k-1} = i] \\
&\leq \sum_{i=0}^{\infty} \left(1 + \frac{3}{r+1}\right) \cdot i \cdot \Pr[X_{k-1} = i] \\
&= \left(1 + \frac{3}{r+1}\right) \cdot \sum_{i=0}^{\infty} i \cdot \Pr[X_{k-1} = i] \\
&= \left(1 + \frac{3}{r+1}\right) \cdot \mathbb{E}[X_{k-1}]
\end{aligned}$$

$$\Longrightarrow \quad \mathbb{E}[X_k] \leq \left(1 + \frac{3}{r+1}\right)^k \cdot n$$

1. Def. bedingter Erwartungswert
2. Abschätzung durch Lemma 3.28

   $\rightarrow$ höchstens $\dfrac{3i}{r+1}$ ausserhalb
3. die Klammer rausziehen
4. Def. Erwartungswert
5. Teleskopieren

Anzahl Punkte - Lower Bound

- $X_k :=$ Anzahl Punkte nach k Iterationen
- $T :=$ Anzahl der Runden/Durchläufe der Schleife

$$\begin{aligned}\mathbb{E}[X_k] &= \mathbb{E}[X_k \mid T \geq k] \cdot \Pr[T \geq k] + \mathbb{E}[X_k \mid T < k] \cdot \Pr[T < k] \\ &\geq \mathbb{E}[X_k \mid T \geq k] \cdot \Pr[T \geq k] \\ &\geq 2^{k/3} \cdot \Pr[T \geq k]\end{aligned}$$

1. Def. bedingter Erwartungswert
2. Erwartungswert und Wahr-keit nicht negativ
3. mindestens einer der Punkte ist in mindesten $k/3$ der Runden ausserhalb
   $\rightarrow$ hat mindestens $2^{k/3}$ Kopien

Laufzeit

Zusammen mit Lower und Upper bound:

$$2^{k/3} \cdot \Pr[T \geq k] \leq \mathbb{E}[X_k] \leq \left(1 + \frac{3}{r+1}\right)^k \cdot n$$

$$\implies \Pr[T \geq k] \leq \min\left\{1, \left(\frac{1+\frac{3}{12}}{2^{1/3}}\right)^k \cdot n\right\} \leq \min\{1, 0.995^k \cdot n\}$$

Mit $k_0 := -\log_{0.995} n$

$$\mathbb{E}[T] = \sum_{k \geq 1} \Pr[T \geq k]$$

$$\leq \sum_{k=1}^{k_0} 1 + \sum_{k > k_0} 0.995^k \cdot n$$

$$= \sum_{k=1}^{k_0} 1 + \sum_{k' \geq 1} 0.995^{k'} \cdot 0.995^{k_0} \cdot n$$

$$= k_0 + \mathcal{O}(1) \leq 200 \ln n + \mathcal{O}(1)$$

1. Def. Erwartungswert, T nichtnegativ
2. 0.995^k dominiert erst ab $k > k_0$
3. $k = k_0 + k'$
4. $0.995^{k_0} \cdot n = 1$ $\quad \sum_{k' \geq 1} 0.995^{k'}$ ist eine geom. Reihe

Semester Recap

Teil 3: Algorithmen

1. Randomisierte Algorithmen
   1.1. Quicksort/Quickselect
   1.2. Target Shooting
   1.3. Primzahltests
2. Bunte/Lange Pfade
3. Flüsse
4. Min-Cut
5. Kleinster umschliessender Kreis
6. Konvexe Hülle

Konvexe Hülle

Gegeben: eine endliche Punktenmenge $P \subseteq \mathbb{R}^d$
Gesucht: die konvexe Hülle von P, $\operatorname{conv}(P)$

Konvexe Hülle: die minimale konvexe Menge, die P beinhaltet
Randkante: geordnetes Paar qr für $q, r \in P$ mit $q \neq r$ s.d. alle Punkte von P links von qr liegen

<u>*JarvisWrap*(P)</u>
1) $h \leftarrow 0$
2) $p_{now} \leftarrow$ Punkt mit kleinster $x-$Koordinate
3) **repeat**
4) $\quad q_h \leftarrow p_{now}$
5) $\quad p_{now} \leftarrow \text{FindNext}(q_h)$
6) $\quad h \leftarrow h + 1$
7) **until** $p_{now} = p_0$
8) **return** $(q_0, \ldots, q_{h-1})$

<u>*FindNext*(q)</u>
1) Wähle beliebig $p_0 \in P \setminus \{q\}$
2) $q_{next} \leftarrow p_0$
3) **for all** $p \in P \setminus \{q, p_0\}$ **do**
4) $\quad$ **if** p **rechts** von qq_{next} **then**
5) $\quad\quad q_{next} \leftarrow p$
6) **return** q_{next}

Laufzeit: $\mathcal{O}(n \cdot h)$, wobei $h :=$ #Ecken in $\operatorname{conv}(P)$
$\rightarrow \mathcal{O}(n^2)$ worst case
$\rightarrow \mathcal{O}(n)$ wenn h konstant

p rechts von qq_{next}
$$\iff (q_x - p_x)(q_{next,y} - p_y) < (q_y - p_y)(q_{next,x} - p_x)$$

Konvexe Hülle - Lower Bound der Laufzeit

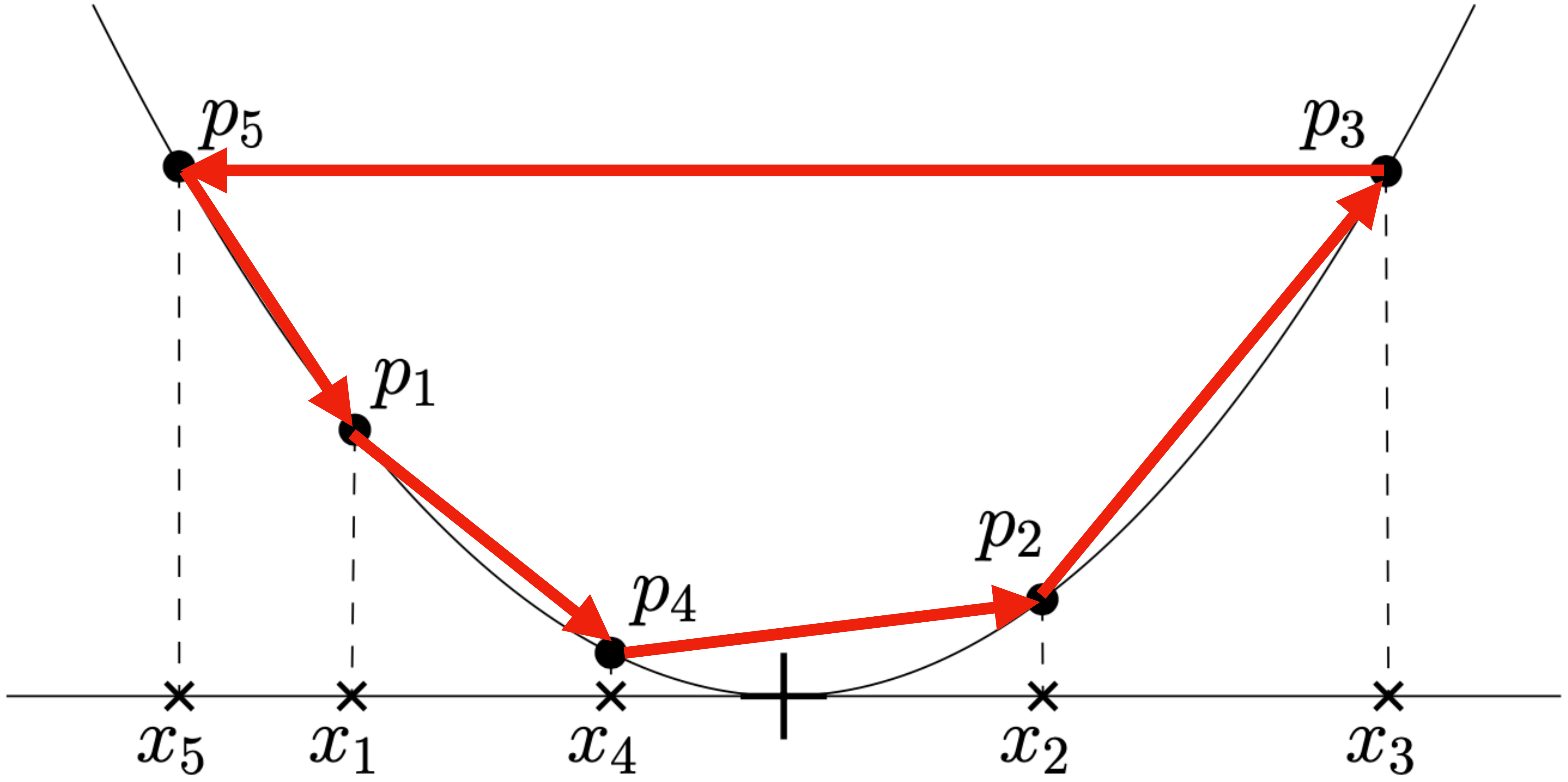
Reduktion vom Sortieren:

$(x_1, \ldots, x_n) \rightarrow ((x_1, x_1^2), \ldots, (x_n, x_n^2))$

können wir Konvexe Hülle in $t(n)$ bestimmen

so können wir in $\mathcal{O}(t(n) + n)$ sortieren

$\implies t(n) \in \Omega(n \cdot \log n)$



Konvexe Hülle - Lokal Verbessern

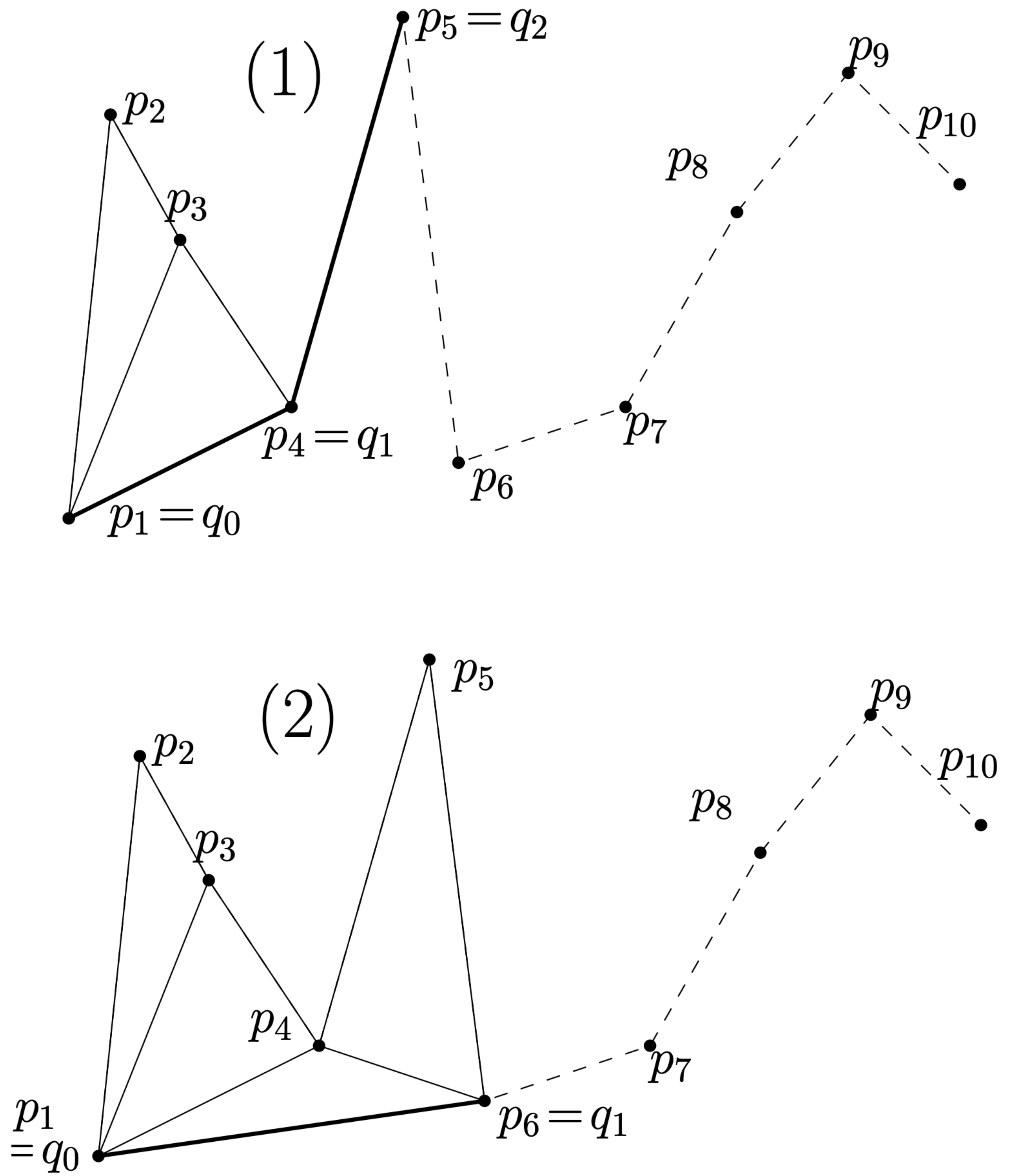
LocalRepair($p_1, p_2, \ldots, p_n$) — ($p_1, p_2, \ldots, p_n$) sortiert

$q_0 \leftarrow p_1$; $h \leftarrow 0$
for $i \leftarrow 2$ **to** n **do** ▷ unterer Rand, links nach rechts
  while $h > 0$ und q_h links von $q_{h-1}p_i$ **do**
    $h \leftarrow h - 1$
  $h \leftarrow h + 1$; $q_h \leftarrow p_i$
    ▷ $(q_0, \ldots, q_h)$ untere konvexe Hülle von $\{p_1, \ldots, p_i\}$
$h' \leftarrow h$
for $i \leftarrow n - 1$ **downto** 1 **do** ▷ oberer Rand, rechts nach links
  while $h > h'$ und q_h links von $q_{h-1}p_i$ **do**
    $h \leftarrow h - 1$
  $h \leftarrow h + 1$; $q_h \leftarrow p_i$
return $(q_0, q_1, \ldots, q_{h-1})$

Analyse

- Sortieren -> $\mathcal{O}(n \log n)$
- $2n - 2 - h$ Mal erfolgreiche Tests (neues Dreieck) -> $\mathcal{O}(n)$
- $2n - 2$ erfolglose Tests (p_i wird zu einer Kante) -> $\mathcal{O}(n)$

→ Wenn die Punkte schon sortiert sind -> $\mathcal{O}(n)$ statt $\mathcal{O}(n \log n)$



ML D28