# AnW – Woche 13

23.05.2024
Nicolas Wehrli

Heute:

I. Minitest
II. Theory Recap
   - Randomized MinCut
   - Recap etwas anderes ?
III. Aufgaben
   - Arbeitsgruppen
   - Prüfung FS17

## I. Minitest

Password: outplayed

Für jeden 2-färbbaren Graph $G$ können wir mithilfe eines Flussproblems herausfinden ob $G$ ein perfektes Matching hat.

Bitte wählen Sie eine Antwort:

Wahr ⊗

Falsch ○

2-färbbar $\Longrightarrow$ bipartit

Matchings per Flussalgo – letzten Donnerstag

Sei $N = (V, A, c, s, t)$ ein Netzwerk. Wenn $c$ nur ganzzahlige Kantengewichte hat, so ist jeder maximale Fluss in $N$ ganzzahlig.

Bitte wählen Sie eine Antwort:

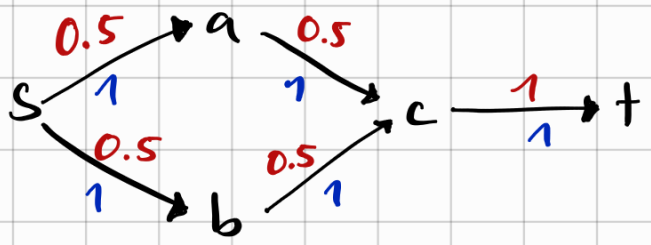
Wahr ○

Falsch ⊗

Common mistake.

Bemerke: Fluss $f$ ganzahlig $\Leftrightarrow$ $f(e) \in \mathbb{N}_0$ $\forall e \in E$

und nicht ~~$val(f) \in \mathbb{N}_0$~~




Kapazität c

Fluss f

f ist maximal, aber nicht ganzahlig!

Sei $N = (V, A, c, s, t)$ ein Netzwerk, $f$ ein Fluss in $N$ und $(S, T)$ ein $s-t$-Schnitt in $N$.

Dann gilt val(f) ≤ cap(S,T)

Lemma, für Beweis siehe "AnW_week 11"

Sei $N = (V, A, c, s, t)$ ein Netzwerk ohne entgegen gerichtete Kanten und $f$ ein Fluss so dass $val(f) > 0$. Sei $N_f$ das Restnetzwerk.

Dann enthält $N_f$ einen (gerichteten) Weg von $t$ nach $s$.

Bitte wählen Sie eine Antwort:

Wahr ⊗

Falsch ○

$val(f) > 0 \Rightarrow \exists$ s-t Pfad mit Kanten $P \subseteq A$ in $N$, so dass

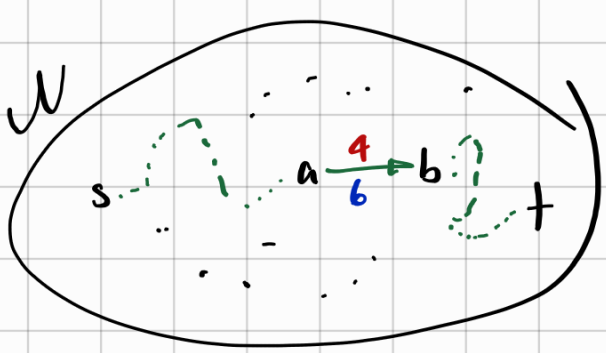
$$f(e) > 0 \quad \forall e \in P$$

Per Konstruktion von $N_f$ gilt

$$\forall e \in P: \; e^{opp} \in A_f \text{ mit } r_f(e^{opp}) = f(e)$$

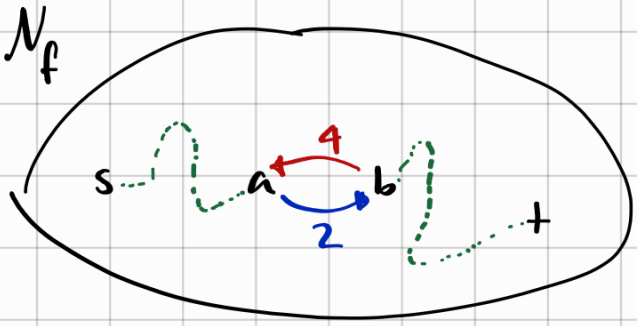
$\Rightarrow \exists$ t-s Pfad in $N_f$ ☒

Betrachte eine beliebige Kante $(a,b)$ im s-t Pfad $P$:




Kapazität s-t Pfad in $N$ mit $f(e) > 0$

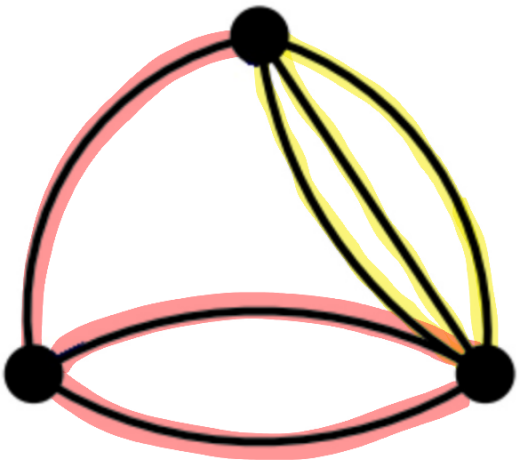
Fluss




t-s Pfad in $N_f$

$\Rightarrow (b,a) \in A_f$

Betrachten Sie den folgenden Multigraphen $G$. Was ist die Wahrscheinlichkeit, dass Cut(G) die richtige Antwort zurück gibt (nach einem Aufruf)?



Wir wählen eine Kante zufällig gleichverteilt und kontrahieren diese.
Wählen wir eine rot markierte Kante erhöht sich der minimale Schnitt.
Bei einer gelb markierten Kante würde der minimale Schnitt erhalten bleiben.

(Nur 1 Kontraktion, da nur 3 Knoten)

$\Rightarrow \quad \frac{3}{6} = 0.5$

Sei $N = (V, A, c, s, t)$ ein Netzwerk ohne entgegen gerichtete Kanten. Sei zudem die Kapazität jeder Kante höchstens $U$. Dann berechnet der Ford-Fulkerson Algorithmus in Zeit $O(mnU)$ einen maximalen Fluss.

Bitte wählen Sie eine Antwort:

Wahr ○

Falsch ⊗

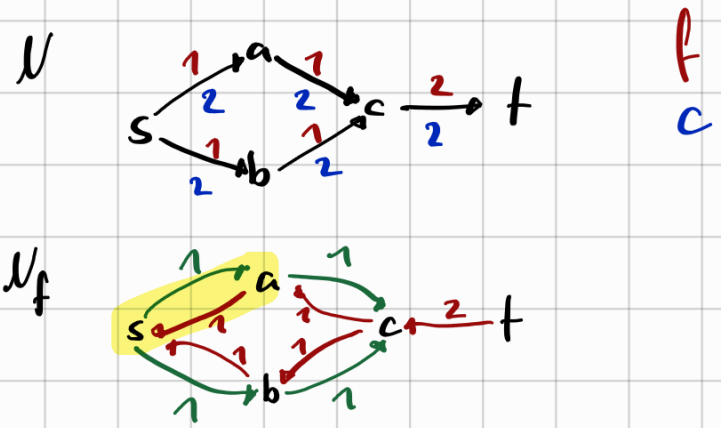
Falls die Kapizitäten in $N$ irrational sind, könnte Ford-Fulkerson nicht terminieren.

Sei $f$ ein maximaler Fluss in einem Netzwerk $N = (V, A, c, s, t)$. Dann gibt es keine zwei Knoten $u, v$ in $V$, sodass sowohl die Kante $(u, v)$ als auch die Kante $(v, u)$ im Restnetzwerk $N_f$ vorkommt.

Bitte wählen Sie eine Antwort:

Wahr ○

Falsch ✗







Betrachten Sie ein Netzwerk $N = (V, A, c, s, t)$ ohne entgegen gerichtete Kanten, einen Fluss $f$ auf $N$, und das dazugehörige Restnetzwerk $N_f$. Wir sagen, dass ein Knoten $v \in V$ erreichbar ist, wenn es einen Pfad von $s$ zu $v$ in $N_f$ gibt. Wir bezeichnen $X$ die Menge der erreichbaren Knoten. Welche der folgenden Aussagen treffen immer zu?

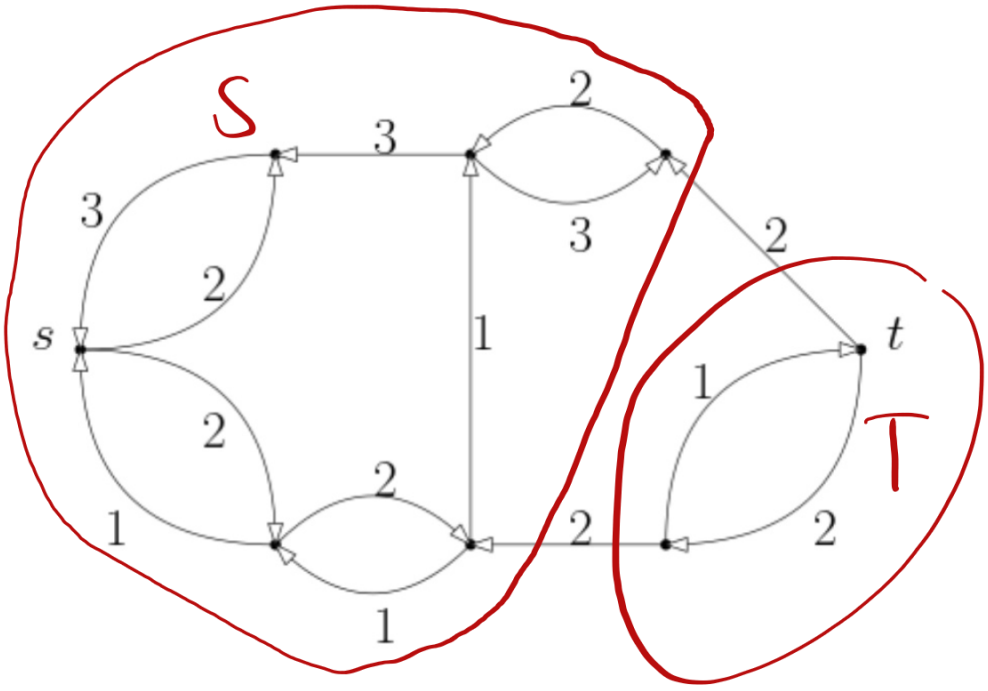
| Wahr | Falsch | |
|---|---|---|
| ✗ | ○ | Falls $(X, V \setminus X)$ ein s-t Schnitt ist, dann gilt $\mathrm{val}(f) = \mathrm{cap}(X, V \setminus X)$. |
| ○ | ✗ | $(X, V \setminus X)$ ist ein s-t Schnitt. |
| ○ | ✗ | $t \in X$. |

1. Folgt vom Maxflow-Mincut Theorem und der Existenz eines maximalen Flusses.
2. t könnte erreichbar sein $(t \in X) \Rightarrow s, t \in X,\ t \notin X \setminus V$
   $\Rightarrow$ kein s-t Schnitt

3. Wenn f maximal, dann
ist t von s aus nicht erreichbar in $N_f$ (i.e. $t \notin X$)
sonst gäbe es einen augmentierenden s-t-Pfad

$\Rightarrow$ Widerspruch zu f maximal ↯

Sei $N = (V, A, c, s, t)$ ein Netzwerk ohne entgegen gerichtete Kanten und $f$ ein Fluss. Das Restnetzwerk $N_f$ ist wie folgt gegeben:




Ist der Fluss maximal?

Bitte wählen Sie eine Antwort:

- [x] Wahr
- [ ] Falsch

Tipp bei solchen Aufgaben, beginnt bei t zu schauen.

Welche der folgenden Punkte treffen zu?

| Richtig | Falsch | |
|---|---|---|
| ☒ | ○ | Es gibt einen s-t-Schnitt $(S,T)$ und einen Fluss $f$, so dass $\text{val}(f) \geq \text{cap}(S,T)$ |
| ○ | ☒ | Es gibt einen s-t-Schnitt $(S,T)$ und einen Fluss $f$, so dass $\text{val}(f) > \text{cap}(S,T)$ |
| ☒ | ○ | Jedes Netzwerk erfüllt $\max_{f \text{ Fluss}} \text{val}(f) = \min_{(S,T) \text{ s-t-Schnitt}} \text{cap}(S,T)$ |
| ○ | ☒ | Jedes Netzwerk erfüllt $\min_{f \text{ Fluss}} \text{val}(f) = \max_{(S,T) \text{ s-t-Schnitt}} \text{cap}(S,T)$ |

1.) Dies kann erfüllt sein, wenn $\text{val}(f) = \text{cap}(S,T)$.

Möglich für maximaler Fluss f

2.) Wir haben letzte Woche (siehe Notizen)

$\text{val}(f) \leq \text{cap}(S,T)$

bewiesen. (für einen beliebigen Fluss f und s-t Schnitt in N)

3.) Maxflow - Mincut Theorem (siehe Notizen Woche 11)

4.) $s \xrightarrow[2]{0} t$ Fluss / Kapizität

# Minimale Schnitte in Multigraphen

**MIN-CUT Problem**. Gegeben ein Multigraph $G$, bestimme die Kardinalität eines *minimalen Kantenschnitts*.

**Multigraph**: Ungerichteter, ungewichteter Graph $G = (V, E)$ ohne Schleifen, aber möglicherweise mit mehreren Kanten zwischen demselben Knotenpaar. (Kann auch durch positiv ganzzahlige Kantengewichte realisiert werden.)

Entweder E ist kein einfaches Set mehr oder Kantengewichte.

Ein Kantenschnitt in einem Multigraph $G=(V,E)$, ist eine Menge von Kanten $C$ s.d. $(V, E \setminus C)$ nicht zsh. ist.

Mit $\mu(G)$ bezeichnen wir die Kardinalität eines kleinstmöglichen Kantenschnitts in $G$, d.h.

$$\mu(G) := \min_{C \subseteq E, (V, E \setminus C) \text{ unzusammenhängend}} |C|$$

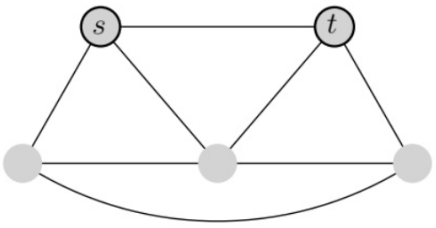
Modelliertes Problem: Gegeben $G$, finde $\mu(G)$.

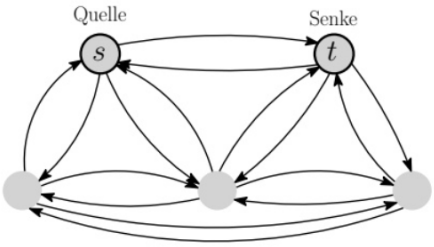
In einem Multigraph $G = (V, E)$ ist der **Grad** $\deg(v) = \deg_G(v)$ eines Knoten $v$ die #inzidenter Kanten (nicht die #Nachbarn). So gilt

$$|E| = \frac{1}{2} \sum_{v \in V} \deg(v) \quad \text{und} \quad \mu(G) \leq \min_{v \in V} \deg(v) .$$

Äquivalent zum Handschlaglemma in simplen Graphen

Wir könnten den Knoten isolieren.




Wir können bereits den kleinsten $s$-$t$-Schnitt berechnen, d.h. die kleinste #Kanten, die man entfernen muss, um $s$ von $t$ zu trennen.

Zeit: $O(mnU)$, bzw. $O(mn(1+\log U))$ oder $O(mn\log n)$.

Für unser Problem fixieren wir ein $s$ und betrachten alle $t \in V \setminus \{s\}$. Jeder Schnitt „ist" ein $s$-$t$-Schnitt, für ein $t \in V \setminus \{s\}$.

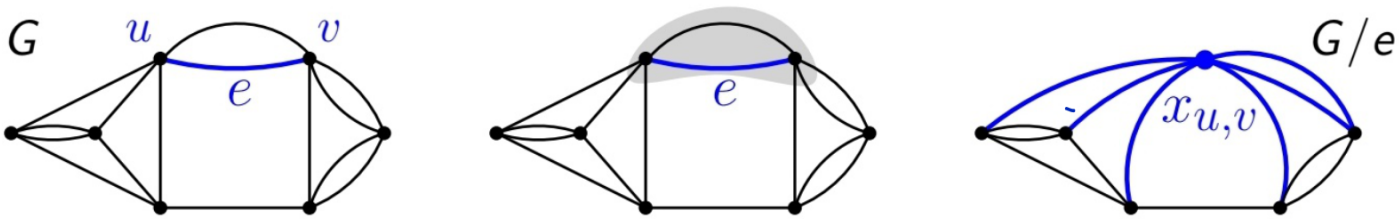
$(n-1)$ Mal $O(mn\log n)$, gibt $O(mn^2\log n) = O(n^4\log n)$.

$O(n^4\log n)$ ist unsere Messlatte, die wir unterbieten wollen.

## Kantenkontraktion:

Sei $G=(V,E)$ mit $e=\{u,v\}\in E$

Die Kontraktion von $e$ verschmilzt $u$ und $v$ zu einem neuen (Super-) Knoten $x_{u,v}$. Die Kanten zwischen $u$ und $v$ werden entfernt. Alle anderen zu $u$ oder $v$ inzidenten Kanten werden nun mit $x_{u,v}$ verbunden.




Den entstehenden Graph bezeichnen wir mit $G/e$.

Sei $k :=$ # Kanten zw. $u$ und $v$

$\deg_{G/e} x_{u,v} := \deg_G u + \deg_G v - 2k$

$|E(G/e)| := |E(G)| - k$

Sei $e = \{u, v\}$. Es gibt eine natürliche Bijektion

$$E(G) \text{ ohne Kanten zw. } u \text{ und } v \quad \longrightarrow \quad E(G/e) \ .$$

Für $w, w' \in V(G) \setminus \{u, v\}$:

$$\{w, w'\} \mapsto \{w, w'\}, \ \{w, u\} \mapsto \{w, x_{u,v}\}, \ \{w, v\} \mapsto \{w, x_{u,v}\}$$

Dadurch können wir eine Bijektion zwischen
Schnitte $C$ in $G$ mit $e \notin C$ $\longmapsto$ Schnitte in $G/e$
induzieren.

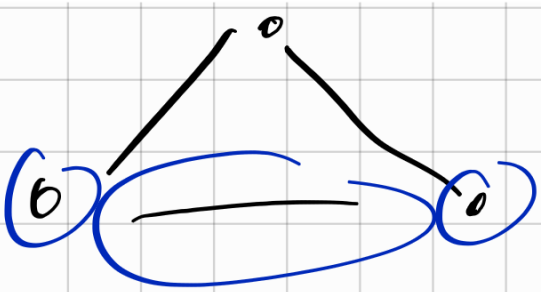
Daraus folgt unmittelbar:

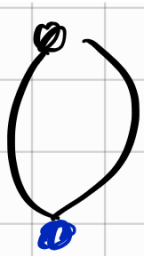
**Lemma**
*Sei $G = (V, E)$ ein Multigraph, $e \in E$. Dann gilt*

$$\mu(G/e) \geq \mu(G) \ .$$

*Falls $G$ einen minimalen Schnitt $C$ mit $e \notin C$ hat, dann gilt*

$$\mu(G/e) = \mu(G) \ .$$





Randomisiertes Verfahren:

**Cut($G$)** — $G$ zusammenhängender Multigraph

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 $G'$

Sei $n := |V(G)|$. Wir setzen voraus:

- Kantenkontraktion in $O(n)$ Zeit.
- Gleichverteilt zufällige Kante in $G$ in $O(n)$ Zeit.

(Erfordert Darstellung der Mehrfachkanten durch Kantengewichte.)

$\Rightarrow$ Cut(G) läuft in $O(n^2)$

Aus dem Lemma (letzte Seite) folgt:

- Ergebnis des Algorithmus ist nie kleiner als $\mu(G)$.
- Falls es einen minimalen Schnitt $C$ gibt, aus dem nie eine Kante kontrahiert wird, so gibt der Algorithmus $\mu(G)$ aus.

Dann schätzen wir mal die Fehlerw'keit für eine einzige Kantenkontraktion ab.

**Lemma**
*Sei $G = (V, E)$, $n := |V|$. Für Kante $e$ gleichverteilt zufällig unter den Kanten in $G$ gilt* $\Pr[\mu(G) = \mu(G/e)] \geq 1 - \frac{2}{n}$.

Beweis. Sei $C$ ein minimaler Schnitt in $G$ und $k := |C| = \mu(G)$.

Wir wissen, dass $\forall v \in V : \deg_G(v) \geq k$ und daher gilt

$$|E| = \frac{1}{2} \sum_{v \in V} \deg_G(v) \geq \frac{kn}{2} .$$

Wegen „$e \notin C \Rightarrow \mu(G/e) = \mu(G)$“ gilt

$$\begin{aligned} \Pr\left[\mu(G) = \mu(G/e)\right] &\geq \Pr[e \notin C] = 1 - \frac{|C|}{|E|} \\ &\geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n} \, . \end{aligned}$$

$\square$

$\hat{p}(G) :=$ Wahrscheinlichkeit, dass Cut$(G)$ den Wert $\mu(G)$ ausgibt

$$\hat{p}(n) := \inf_{G=(V,E),\,|V|=n} \hat{p}(G) \, .$$

Lemma

*Es gilt für alle $n \geq 3$*

$$\hat{p}(n) \geq \left(1 - \frac{2}{n}\right) \cdot \hat{p}(n-1) \, .$$

Beweis. Sei $G = (V, E)$, $n := |V|$. Damit Cut$(G)$ tatsächlich $\mu(G)$ ausgibt, müssen die beiden folgenden Ereignisse eintreten:

- $E_1 :=$ Ereignis $\mu(G) = \mu(G/e)$.
- $E_2 :=$ Ereignis, dass Cut$(G/e)$ den Wert $\mu(G/e)$ ausgibt.

Es gilt nun

$$\hat{p}(G) = \Pr[E_1 \wedge E_2] = \Pr[E_1] \cdot \Pr[E_2 \mid E_1] \geq (1 - 2/n) \cdot \hat{p}(n-1) \, .$$

Da dies für jeden Multigraph $G$ mit $n$ Knoten gilt, folgt auch $\hat{p}(n) \geq (1 - \frac{2}{n}) \cdot \hat{p}(n-1)$.

$\square$

Es gilt also $\hat{p}(n) \geq \frac{n-2}{n} \cdot \hat{p}(n-1)$. Wir erhalten so

$$\hat{p}(n) \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \cdot \underbrace{\hat{p}(2)}_{=1} = \frac{2}{n(n-1)}$$

## Lemma

*Für alle $n \geq 2$ gilt*

$$\hat{p}(n) \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$$

Aus der Geometrischen Verteilung folgt
$\mathbb{E}[\#\text{Aufrufe bis Cut}(G)\ \mu(G)\ \text{findet}] = \binom{n}{2}$

Wir wiederholen den Algorithmus Cut($G$) $\lambda\binom{n}{2}$ mal, für ein $\lambda > 0$, und geben dann kleinsten je erhaltenen Wert aus.

Zur Erinnerung: Ein Aufruf von Cut($G$) benötigt Zeit $O(n^2)$.

## Satz

*Für den Algorithmus der $\lambda\binom{n}{2}$-maligen Wiederholung von* Cut($G$) *gilt:*

(1) *Der Algorithmus hat eine Laufzeit von $O(\lambda n^4)$.*

(2) *Der kleinste angetroffene Wert ist mit einer Wahrscheinlichkeit von mindestens $1 - e^{-\lambda}$ gleich $\mu(G)$.*

Mit $\lambda := \ln n$, haben wir **Zeit** $O(n^4 \log n)$ mit **Fehlerw'keit** $\leq 1/n$.

Das hatten wir ja schon!

Für $\lambda\binom{n}{2}$-Wiederholungen:

$(\forall x \in \mathbb{R}: 1+x \leq e^x)$

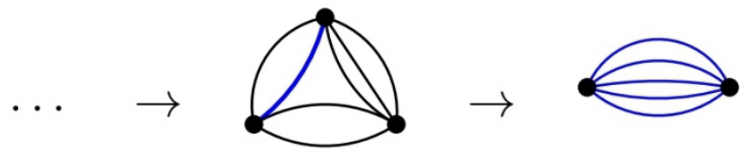
$$\Pr[\text{Success}] = 1 - \Pr[\text{Fail}] \geq 1 - e^{-\lambda}$$

$$\Pr[\text{Fail}] = \left(1 - \hat{p}(n)\right)^{\lambda\binom{n}{2}} \leq \left(1 - \frac{1}{\binom{n}{2}}\right)^{\lambda\binom{n}{2}} \leq \left(e^{-\frac{1}{\binom{n}{2}}}\right)^{\lambda\binom{n}{2}}$$

$$= e^{-\lambda}$$

# Bootstrapping:

Die letzten Schritte haben die grösste Fehlerw'keit



$$\hat{p}(n) \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \underbrace{\frac{3}{5}}_{0.6} \cdot \underbrace{\frac{2}{4}}_{0.5} \cdot \underbrace{\frac{1}{3}}_{0.33} \cdot \underbrace{\hat{p}(2)}_{=1} = \frac{2}{n(n-1)}$$

Wir sollten Schritt von 3 auf 2 Knoten sorgfältiger machen
... und am besten den Schritt von 4 auf 3 auch
...

Die Idee ist, dass wir die Rekursion bei $t$-Knoten abbrechen und die letzten Schritte im kleineren Multigraphen mit einem anderen Algorithmus lösen. (z. Bsp. deterministisch mit Flüssen in $O(t^9 \log t)$)

In der Vorlesung verwenden wir den Cut$(G')$ Algorithmus mit

$\binom{t}{2}$ Wiederholungen mit Erfolgsw'keit $p^*(t) \geq 1 - e^{-1}$ in $O(t^9)$.

Der Unterschied zum vorherigen Vorgehen:

## Altes Vorgehen:

In jedem Schritt wählen wir eine zufällige Kante $e$ und kontrahieren bis wir 2 Knoten haben.

## Neues Vorgehen:

In jedem Schritt wählen wir eine zufällige Kante $e$

und kontrahieren bis wir $t$ Knoten haben

Ab $t$ Knoten simulieren wir die restliche Kantenkontraktion $\binom{t}{2}$-mal und wählen das beste Resultat.

## Neue W'keitsanalyse:

Note: In den Slides verwenden sie $\hat{p}_t(t)$. Es ist auch richtig. Die Abschätzung von $\hat{p}_t(t)$ geschieht dann bei (*)

$$\hat{p}_t(n) \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \ldots \cdot \frac{t}{t+2} \cdot \frac{t-1}{t+1} \cdot p^*(t) \overset{(*)}{\geq} \underbrace{\frac{t(t-1)}{n(n-1)} \cdot \left(1 - e^{-1}\right)}_{\overline{p}_t(n)}$$

$\lambda \frac{1}{\overline{p}_t(n)}$-maliges Wiederholen gibt Fehlerw'keit $\leq e^{-\lambda}$ und Laufzeit

$$\overbrace{\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1}}^{\#\text{Wiederholungen}} \cdot O\Big( \underbrace{n(n-t)}_{\text{Reduktion auf } t \text{ Knoten}} + \overbrace{t^4}^{\text{Alg. auf } t \text{ Knoten}} \Big) = O\left(\lambda\left(\frac{n^4}{t^2} + n^2 t^2\right)\right).$$

Erfolgsw'keit $\geq 1 - e^{-\lambda}$ und Laufzeit

$$O\Big(\lambda \underbrace{\left(\frac{n^4}{t^2} + n^2 t^2\right)}_{\to \min}\Big) \overset{t=\sqrt{n}}{=} O(\lambda n^3)$$

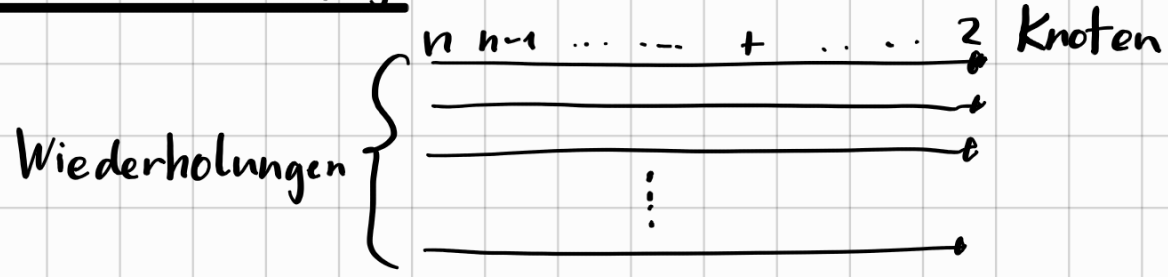
**Bootstrapping:** Es bietet sich an, die gleiche Methode nun mit dem neuen $O(n^3)$ Algorithmus statt dem $O(n^4)$ Algorithmus zu versuchen, und tatsächlich bekommen wir einen noch besseren, etc.

Im „Limit“ entwickelt sich so ein $O(n^2 \text{polylog}(n))$-Algorithmus.

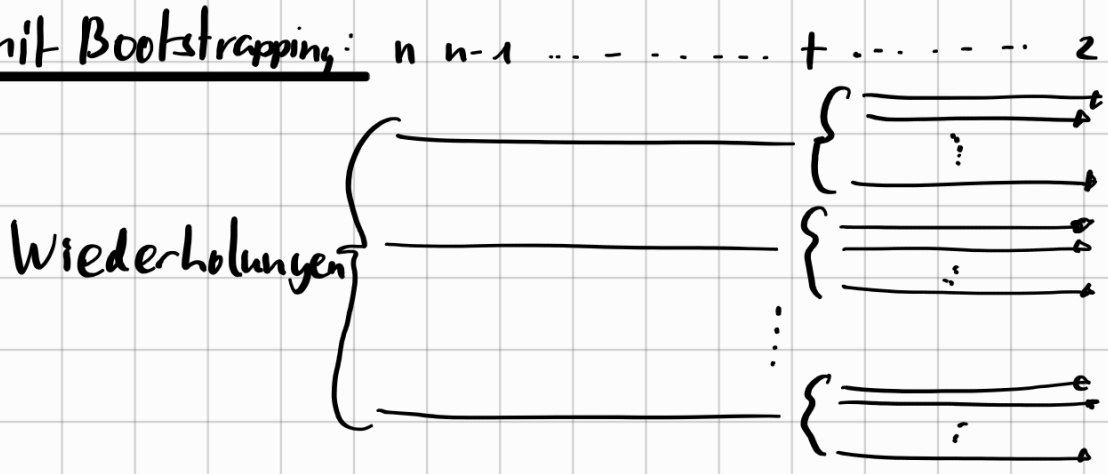
[Karger&Stein'96]

✔ Bootstrapping wird erreicht, dadurch dass man die Kanten im (Multi-)Graph kontrahiert, bis $t$ Knoten übrig bleiben und dann wiederholt den Algorithmus mit diesem Multigraph mit $t$ Knoten als Startpunkt ausführt.

Algo ohne Bootstrapping:




mit Bootstrapping:




✖ Bootstrapping wird benutzt um eine Erfolgswahrscheinlichkeit von mindestens $1 - e^{-1}$ zu garantieren.

War schon ohne Bootstrapping mit $1 \cdot \binom{n}{2}$ Wiederholungen garantiert.

✖ Bootstrapping wird erreicht, dadurch dass man Kanten im (Multi-)Graph kontrahiert, bis $t$ Knoten übrig bleiben und dann nur weiter macht, falls der minimale Schnitt bis hier hin erhalten wurde.

Nein, Bootstrapping führt die Simulation immer zu Ende. Wenn wir effizient überprüfen könnten, ob der minimale Schnitt bei einer Kontraktion erhalten bleibt, könnten wir einen schnellen deterministischen Algorithmus für das Problem finden.

## Aufgabe 1 – *Arbeitsgruppen*

Zu einem Kongress werden Mitarbeiter von $m$ Firmen gesandt. Dabei sendet die $i$-te Firma $c_i$ viele Mitarbeiter. Während des Kongresses sollen die anwesenden Personen in bis zu $r$ verschiedene Arbeitsgruppen mit jeweils höchstens 5 Mitgliedern aufgeteilt werden, wobei keine zwei Mitarbeiter derselben Firma in derselben Arbeitsgruppe sein sollen.

Wir sollen eine solche Aufteilung mit Hilfe von Fluss-Algorithmen finden.

(a) Definieren Sie dazu ein geeignetes Netzwerk $N = (V, A, c, s, t)$ und zeigen Sie, dass es genau dann eine mögliche Aufteilung wie oben gibt, wenn maxflow($N$) eine gewisse (welche?) Eigenschaft hat.

(b) Angenommen, wir erhalten als Eingabe die Zahlen $m, r, c_1, \ldots, c_m$ sowie die Listen $L_1, \ldots, L_m$ mit den Namen aller Teilnehmer des Kongresses. Wir können davon ausgehen, dass keine zwei Teilnehmer den gleichen Namen haben. Geben Sie einen Algorithmus an, der dies als Eingabe verwendet, und als Ausgabe entweder $r$ Listen, $G_1, ..., G_r$ gibt, wobei $G_i$ die Namen aller Teilnehmer der $i$:ten Gruppe in einer gültigen Zuordnung enthält, oder antwortet „Zuordnung nicht möglich“.

(a) Die Aufgabe besteht darin, Mitarbeiter von Firmen auf Gruppen aufzuteilen mit den folgenden Kriterien:

(i) Jede Firma $i$ hat genau $c_i$ Mitarbeiter auf die Gruppen verteilt.

(ii) Jede Gruppe hat höchstens 5 Mitarbeiter.

(iii) Keine Firma verteilt mehr als einen Mitarbeiter auf eine Gruppe.

(iv) Es gibt höchstens $r$ viele Gruppen.

Wir modellieren das Problem als Flussproblem auf einem Netzwerk $N$ wie folgt: $N$ hat eine Quelle $s$ und eine Senke $t$. Darüber hinaus fügen wir für jede Firma $i$ ($1 \leq i \leq m$) einen Knoten $f_i$ und für jede der $r$ (potentiell leeren) Arbeitsgruppen $j$ einen Knoten $a_j$ ein. Die Knoten $f_i$ sind durch einer Kante $(s, f_i)$ mit der Quelle verbunden, die Kapazität $c_i$ hat (intuitiv wird der Fluss entlang dieser Kanten bestimmen, wie viele Mitarbeiter der Firma an Arbeitsgruppen teilnehmen können). Des Weiteren verbinden wir jede Firma $f_i$ mit jeder Arbeitsgruppe $a_j$ durch eine Kante $(f_i, a_j)$ der Kapazität 1 (diese Kante stellt sicher, dass aus jeder Firma nur ein Mitarbeiter pro Arbeitsgruppe vertreten ist). Schlussendlich verbinden wir jede Arbeitsgruppe durch eine Kante $(a_j, t)$ mit Kapazität 5 mit der Senke (diese Kante stellt sicher, dass jede Arbeitsgruppe höchstens 5 Teilnehmer hat).

Formal bedeutet dies, dass unser Netzwerk $N = (V, A, c, s, t)$ wie folgt definiert ist:

$V := \{s, f_1, \ldots, f_m, a_1, \ldots, a_r, t\}$
$A := \{(s, f_i) \mid i \in [m]\} \cup \{(f_i, a_k) \mid i \in [m], k \in [r]\} \cup \{(a_k, t) \mid k \in [r]\}$

$$c(u,v) := \begin{cases} c_i, & \text{falls } u = s \text{ und } v = f_i, i \in [m] \\ 1, & \text{falls } u = f_i \text{ und } v = a_k, i \in [m], k \in [r] \\ 5, & \text{falls } u = a_k \text{ und } v = t, k \in [r]. \end{cases}$$

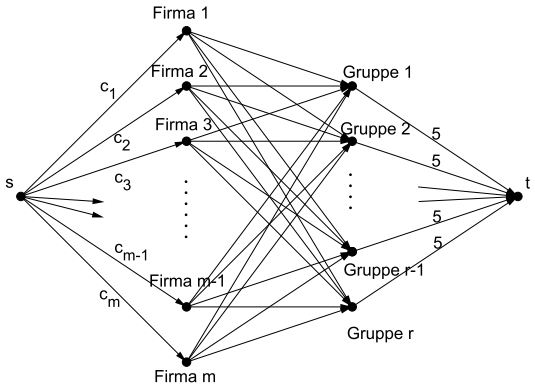



Abbildung 1: Die unbeschrifteten Kanten haben Kapazität 1.

Wir wollen im Folgenden zeigen, dass es eine zulässige Aufteilung gibt genau dann, wenn es in $N$ einen maximalen Fluss mit Wert $\text{maxflow}(N) \geq c := \sum_{i=1}^{m} c_i$ gibt. Dazu müssen wir einerseits zeigen, wie wir aus einer zulässigen Aufteilung einen Fluss der Kapazität $c$ konstruieren können und wie wir andererseits aus einem Fluss der Kapazität $c$ eine solche Aufteilung konstruieren konnen.

Zunächst sehen wir, dass $\text{maxflow}(N) \leq c$, da die Summe aller von $s$ ausgehenden Kapazitäten genau $c$ ist. Daher gilt $\text{maxflow}(N) \geq c \iff \text{maxflow}(N) = c$.

Aufteilung $\implies$ Fluss: Nehmen wir zunächst an, dass es eine zulässige Aufteilung gibt (und fixieren wir eine solche Aufteilung). Dann konstruieren wir einen Fluss, indem wir entlang jeder Kante $(s, f_i)$ den Fluss $c_i$ schicken und von jedem Knoten $f_i$ Fluss 1 entlang $(f_i, a_j)$ genau dann, wenn ein Mitarbeiter von $f_i$ an der Gruppe $a_j$ teilnimmt - und Fluss 0 sonst. Schliesslich schicken wir entlang $(a_j, t)$ den Fluss, der der Anzahl der Teilnehmer in Gruppe $j$ entspricht. Man sieht leicht, dass dieser Fluss zulässig ist. Die Kapazität enlang der Kanten $(f_i, a_j)$ wird nicht überschritten, da in jeder Gruppe höchstens ein Mitarbeiter pro Firma vertreten ist; die Kapazität entlang der Kanten $(a_j, t)$ wird respektiert, da es pro Gruppe höchstens 5 Teilnehmer gibt. Aufgrund der Flusserhaltung $\text{inflow}(a_j) = \text{outflow}(a_j)$ gilt an jedem Firmen- und Arbeitsgruppenknoten per Konstruktion ($\text{inflow}(f_i) = c_i = \text{outflow}(f_i)$, da $c_i$ Mitarbeiter von $f_i$ auf die Gruppen verteilt werden. Schliesslich wollen wir noch zeigen, dass dieser Fluss maximal ist. Dafür genügt es, zu bemerken, dass $\text{val}(f) = \text{outflow}(s) = \sum_{i=1}^{m} c_i = c$.

Fluss $\implies$ Aufteilung: Sei nun $f$ ein maximaler Fluss der Kapazität $c$. Bevor wir die Aufteilung konstruieren, machen wir zwei Beobachtungen: Aus Satz 3.12 folgt, dass wir annehmen können, dass dieser Fluss ganzzahlig ist, da alle Kapazitäten ganzzahlig sind. Insbesondere ist der Fluss entlang jeder Kante $(f_i, a_j)$ entweder 0 oder 1. Des Weiteren wissen wir, dass entlang jeder Kante $(s, f_i)$ der Fluss $c_i$ fliesst, da jede Firma $c_i$ Mitarbeiter zur Konferenz schickt. Wir weisen nun einen Mitarbeiter der Firma $f_i$ der Gruppe $a_j$ zu, genau dann wenn $f((f_i, a_i)) = 1$. Da $\text{inflow}(f_i) = c_i$ gilt auch $\text{outflow}(f_i) = c_i$ und deshalb werden $c_i$ Mitarbeiter der Firma auf die Arbeitsgruppen aufgeteilt. Man sieht ebenfalls leicht, dass pro Arbeitsgruppe nur ein Mitarbeiter aus jeder Firma vertreten ist und dass in jeder Arbeitsgruppe höchstens 5 Teilnehmer sind (da $\text{inflow}(a_j) = \text{outflow}(a_j) \leq 5$). Damit haben wir gezeigt, dass aus der Existenz eines maximalen Flusses mit Wert $c$ folgt, dass es eine zulässige Aufteilung gibt und den Beweis abgeschlossen.

(b) Wir konstruieren unser Netzwerk wie in a). Die Strategie ist nun, einen maximalen ganzzahligen Fluss $f$ zu finden. Da alle Kapazitäten ganzzahlig sind, ist dies z.B. mittels Ford-Fulkerson möglich. Dann prüfen wir, ob $\text{val}(f) = c$.
Falls nein, dann geben wir "Zuordnung nicht möglich" aus.
Falls ja, dann nutzen wir die Richtung Fluss $\implies$ Aufteilung aus a), um eine gültige Aufteilung zu finden. Durch den aus a) erhaltenen Fluss $f$ können wir für jede Arbeitsgruppe $k$ alle Firmen finden als $F_k := \{i \in [m] | f(f_i, a_k) = 1\}$, die Mitarbeiter in die jeweilige Gruppe schicken. Umgekehrt können wir für jede Firma $i$ die zugehörigen Arbeitsgruppen finden als $Z_i := \{k \in [r] | f(f_i, a_k) = 1\}$ mit $|Z_i| = c_i$. Sei nun $\phi_i : Z_i \to [c_i]$ eine beliebige bijektive Abbildung für jedes $i \in [m]$, z.B. die kanonisch kleinste solche Abbildung. $\phi_i$ bildet eine Arbeitsgruppe auf einen Mitarbeiter der Firma $i$ ab. Nun können wir die Liste der Teilnehmer der Gruppe definieren als $G_k := (L_{i,\phi_i(k)})_{i \in F_k}$.

Die Korrektheit des Algorithmus folgt direkt aus Aufgabe a). Die Laufzeit des Algorithmus ist dominiert duch die Subroutine den maximalen Fluss zu finden, für Ford-Fulkerson also $\mathcal{O}(|A| \cdot |V| \cdot U)$ wobei $|A| \in \Theta(m \cdot r)$ die Anzahl der Kanten des konstruierten Netzwerks ist, $|V| \in \Theta(m+r)$ die Anzahl der Knoten und $U := \max(c_1, ..., c_m, 5)$ die maximale Kapazitat der Kanten. Die restliche "Buchhaltung" des Algorithmus (das Erstellen der Listen $G_k$) benötigt Zeit $\mathcal{O}(mr + c)$.

Insgesamt haben wir also Laufzeit $\mathcal{O}(|A| \cdot |V| \cdot U + mr + c) = \mathcal{O}(mr(m+r)U + c) = \mathcal{O}(mr(m+r)U)$, da $c \leq m \cdot U$.

ETH Zürich
Institute of Theoretical Computer Science
Prof. Angelika Steger, Prof. Emo Welzl
Dr. Johannes Lengler

FS 2017

# Algorithmen und Wahrscheinlichkeit
# Prüfung

**Schlagen Sie die Klausur erst auf Anweisung der Aufsicht auf!**

**Kandidat/in:**

Name: .....

Vorname: .....

Stud.-Nr.: .....

Ich bezeuge mit meiner Unterschrift, dass ich die Prüfung unter regulären Bedingungen ablegen kann und dass ich die unten stehenden allgemeinen Bemerkungen gelesen und verstanden habe.

Unterschrift: .....

**Allgemeine Bemerkungen und Hinweise:**

- Diese Prüfung besteht neben diesem doppelseitigen Deckblatt aus 6 beidseitig bedruckten Aufgabenblättern mit insgesamt 4 Aufgaben. Am Ende der Klausur finden Sie leere Blätter für Notizen. Ausserdem liegt eine Formelsammlung an Ihrem Platz.
- Die Prüfung dauert zwei Stunden. Vorzeitige Abgaben sind nicht möglich.
- Falls Sie während der Prüfung in irgendeiner Weise gestört oder beeinträchtigt werden, melden Sie dies sofort der Aufsichtsperson. Spätere Klagen werden nicht akzeptiert.
- Schreiben Sie nicht mit Bleistift und nicht mit roter oder grüner Farbe.
- Alle Mobiltelefone und sonstigen elektronischen Geräte müssen vollständig ausgeschaltet sein und im Gepäck verstaut werden.
- Pro Aufgabe ist höchstens eine gültige Version eines Lösungsversuchs zulässig. Streichen Sie ungültige Lösungsversuche klar durch.
- Abschreiben und sonstige Versuche des Betrugs führen zum sofortigem Ausschluss von der Prüfung und können rechtliche Folgen haben.
- Wenn Sie zusätzliches Papier (kein eigenes!) verwenden, dann versehen Sie jedes Blatt mit Ihrem Namen.
- **Vergessen Sie nicht, dieses Deckblatt zu unterschreiben.**

Viel Erfolg!

| Aufgabe | **1** (16-25 Punkte) | **2** (21 Punkte) | **3** (7 Punkte) | **4** (14 Punkte) | $\Sigma$ (58-67 Punkte) |
|---|---|---|---|---|---|
| Punkte | | | | | |
| 1. Korrektur | | | | | |
| 2. Korrektur | | | | | |

## Aufgabe 1 – *Multiple Choice*

$G = (V, E)$ bezeichnet immer einen Graphen mit $n$ Knoten und $m$ Kanten. $X$ und $Y$ bezeichnen immer eine Zufallsvariable. Wir nehmen an, dass alle vorkommenden Werte (Erwartungswerte, bedingte Wahrscheinlichkeiten, ...) wohldefiniert sind.

Welche der folgenden Aussagen sind immer wahr, welche nicht? *Sie brauchen Ihre Antworten nicht zu begründen.*

1. Man kann in Zeit $O(m + n)$ entscheiden, ob ein Graph $G$ 2-zusammenhängend ist, falls $G$ als Adjazenzliste gespeichert ist. *(1 Punkte)*

   WAHR ☐ FALSCH ☐

2. Beschreiben Sie mit jeweils einem Schlagwort die drei wesentlichen Schritte des 2-Approximations Algorithmus für das metrische Travelling Salesman Problem (Problem des Handlungsreisenden). *(3 Punkte)*

   1. . . . . .

   2. . . . . .

   3. . . . . .

3. Sei $X$ eine Indikator-Zufallsvariable und $p := \Pr[X = 1]$. Geben Sie eine möglichst einfache Formel für $\mathbb{E}[X]$ an. *(1 Punkte)*

   . . . . . .

4. Sei $X$ eine Bernoulli-Zufallsvariable, und seien $X_1 := X, \ldots, X_n := X$. Dann ist die Summe der $X_i$ binomialverteilt. *(1 Punkte)*

   WAHR ☐ FALSCH ☐

5. Ein Las Vegas-Algorithmus terminiert immer. *(1 Punkte)*

   WAHR ☐ FALSCH ☐

6. Bestimmt man einen minimalen Schnitt mit zufälligen Kantenkontraktionen, so hat man die höchste Fehlerwahrscheinlichkeit im ersten Schritt. *(1 Punkte)*

   WAHR ☐ FALSCH ☐

7. Bis zu welcher Länge (in $\Theta$-Notation) lassen sich Pfade in einem Graphen mit dem Bunte-Pfade-Algorithmus in erwartet polynomieller Zeit finden? *(1 Punkte)*

   . . . . . .

Welche der folgenden Aussagen sind immer wahr, welche nicht? Falls nicht, geben Sie ein Gegenbeispiel an.
*(Falls die Aussage wahr ist, 1 Punkt für korrekte Antwort. Falls die Aussage falsch ist, 2 Punkte für korrekte Antwort mit korrektem Gegenbeispiel, 0 Punkte sonst.)*

8. Entfernt man eine Kante aus einem Baum $T$, so zerfällt $T$ in genau zwei Zusammenhangskomponenten.

   WAHR ☐ FALSCH ☐

   Gegenbeispiel:

9. Sei $e = \{u, v\}$ eine Brücke in einem Graph $G$. Dann ist $u$ ein Artikulationsknoten, falls $\deg(u) \geq 2$.

   WAHR ☐ FALSCH ☐

   Gegenbeispiel:

10. Der Greedy-Algorithmus aus der Vorlesung zur Färbung von Graphen färbt jeden Graphen $G$ mit höchstens $\chi(G) \cdot \log n$ Farben (für jede Knotenreihenfolge).

    WAHR ☐ FALSCH ☐

    Gegenbeispiel:

11. Ein Matching für das es keinen augmentierenden Pfad gibt ist kardinalitätsmaximal.

    WAHR ☐ FALSCH ☐

    Gegenbeispiel:

12. Für zwei Ereignisse $A, B$ gilt immer $\Pr[A \mid B] \leq \Pr[A]$.

    WAHR ☐ FALSCH ☐

    Gegenbeispiel:

13. Sind $X$ und $Y$ zwei unabhängige Zufallsvariabeln, so sind auch die Zufallsvariabeln $X^2$ und $Y^2$ unabhängig.

WAHR ☐ FALSCH ☐

Gegenbeispiel:

14. Für jede Zufallsvariable $X$ gilt $\mathbb{E}[X^2] = (\mathbb{E}[X])^2$.

WAHR ☐ FALSCH ☐

Gegenbeispiel:

15. Sei $X \geq 0$ eine nicht-negative Zufallsvariable mit Erwartungswert $\mathbb{E}[X] = 10$. Dann gilt $\Pr[X \leq 1] \leq 1/10$.

WAHR ☐ FALSCH ☐

Gegenbeispiel:

16. Sei $f$ ein maximaler Fluss auf einem Netzwerk $N = (V, A, c, s, t)$. Dann gibt es keine zwei Knoten $u, v$ in $V$, sodass sowohl die Kante $(u, v)$ als auch die Kante $(v, u)$ im Residualgraphen vorkommen.

WAHR ☐ FALSCH ☐

Gegenbeispiel:

## Aufgabe 2 – *Vermischtes*

Begründen Sie jeweils knapp Ihre Antwort.

(a) Geben Sie alle $m, n \geq 2$ an, für welche das $m \times n$-Gitter einen Hamiltonkreis enthält. *(5 Punkte)*

(b) Beschreiben Sie einen möglichst einfachen Algorithmus, der als Input einen Graphen $G = (V, E)$, welcher als Adjazenzliste gespeichert wurde, bekommt, und in Zeit $O(|V| + |E|)$ entscheidet, ob der Graph eine Eulertour besitzt. *Hinweis:* Ihr Algorithmus braucht keine Eulertour auszugeben. *(5 Punkte)*

(c) Skizzieren Sie das Argument, warum in jeder Phase des Quick-Select-Algorithmus mit Wahrscheinlichkeit mindestens $1/2$ die Länge des Arrays auf $3/4$ der ursprünglichen Länge oder weniger reduziert wird. *(3 Punkte)*

(d) Sei $f$ ein nicht-maximaler Fluss auf einem Netzwerk $N = (V, A, c, s, t)$. Zeigen Sie: Dann gibt es keinen $s$-$t$-Schnitt $(S, T)$ mit $\text{cap}(S, T) = \text{val}(f)$. *(4 Punkte)*

(e) Sei $n \in \mathbb{N}$, und sei $X$ eine Zufallsvariable mit $\mathbb{E}[X] = n \log n$ und $\mathrm{Var}[X] = n^2$. Zeigen Sie: Es gibt ein $C > 0$, sodass $\Pr[X \geq n \log n + C \cdot n] \leq 0.01$. *(4 Punkte)*

## Aufgabe 4 – *Permutation*

Betrachten Sie folgenden (ineffizienten) Algorithmus zur Ziehung einer zufälligen Permutation.

**Input:** Eine Zahl $n \in \mathbb{N}$.
**Output:** Eine gleichverteilte zufällige Permutation von $\{1, \ldots, n\}$.

1. $L :=$ leere Liste.
2. **WHILE** length$(L) < n$
3. Ziehe gleichverteilt zufällig ein $i \in \{1, \ldots, n\}$;
4. Falls $i$ nicht in $L$ vorkommt, hänge $i$ an $L$ an;
5. **END WHILE**
6. return $L$

Sei $T$ die Anzahl Schleifendurchläufe der Zeilen 2-5.

(a) Berechnen Sie (mit Begründung) $\mathbb{E}[T]$. *(6 Punkte)*

(b) Berechnen Sie (mit Begründung) $\mathrm{Var}[T]$ in $\Theta$-Notation. *(8 Punkte)*

(a):

(b):