MINIQUIZ

Wenn ein probabilistischer Algorithmus mit einer Wahrscheinlichkeit von mindestens $2/3$ eine korrekte JA/NEIN-Antwort liefert und wir ihn unabhängig n-mal ausführen, ist die Wahrscheinlichkeit, dass er mehr als $n/2$ Mal eine falsche Antwort gibt, kleiner als $\exp(-\Omega(n))$.

$E[X] = \frac{1}{3}$

Chernoff: $\Pr[X \geq (1+\delta)E[X]] \leq e^{-\frac{1}{3}\cdot\delta^2\cdot E[X]}$

$\Pr[X \geq \frac{n}{2}] = \Pr[X \geq (1+\frac{1}{2})E[X]] \leq e^{-\frac{1}{3}\cdot\frac{1}{4}\cdot\frac{n}{3}} = e^{-\frac{n}{24}}$

- [x] Wahr
- [ ] Falsch

Es gibt einen probabilistischen Algorithmus, der testet, ob n eine Primzahl ist, und zwar in einer Zeit, die polynomisch in $\log n$ ist.

Miller-Rabin

- [x] Wahr
- [ ] Falsch

Sei $\mathcal{A}$ ein probabilistischer Algorithmus, der eine Zahl im Bereich $[0, 1]$ ausgibt, s.t. $\mathbb{E}[\mathcal{A}] = s$. Dann nehmen Sie eine Durchschnitt von $O(s^{-1}\log(1/\delta)/\varepsilon^2)$ unabhängigen Läufen von $\mathcal{A}$ können wir eine Schätzung $\hat{s}$ finden, die mit einer Wahrscheinlichkeit von mindestens $1-\delta$ erfüllt, dass $\hat{s} \in [(1-\varepsilon)s, (1+\varepsilon)s]$.

- [x] Wahr
- [ ] Falsch

Es gibt einen probabilistischen Algorithmus, der prüft, ob ein Graph in der Zeit $2^{O(k)}\mathrm{poly}(n)$ einen einfachen Pfad der Länge von mindestens k hat. Darüber hinaus gibt der Algorithmus immer „NEIN" aus, wenn der Graph keinen solchen Pfad hat.

- [x] Wahr
- [ ] Falsch

Wir haben eine Menge von k Objekten und färben sie zufällig, jedes unabhängig in eine von k Farben (mit gleicher Wahrscheinlichkeit). Die Wahrscheinlichkeit, dass alle Objekte unterschiedliche Farben haben, beträgt mindestens 2^{-Ck} für eine Konstante C (die nicht von k abhängt).

$\frac{k!}{k^k} \leq e^{-k}$ $\quad e^k = \sum_{i=0}^{\infty}\frac{k^i}{i!} \geq \frac{k^k}{k!}$

- [x] Wahr
- [ ] Falsch

Für ein Netzwerk (V, A, c, s, t) gilt: Wenn $f : A \to \mathbb{R}_0^+$ ein nicht-maximaler Fluss ist, gibt es einen Pfad P von s nach t in (V, A), sodass auf jeder Kante e von P gilt: $f(e) < c(e)$.

- ○ Wahr
- ⊗ Falsch

Für ein Netzwerk (V, A, c, s, t), wenn $f : A \to \mathbb{R}_0^+$ ein beliebiger Fluss ist und (S, T) ein beliebiger $s - t$-Schnitt ist, dann $\mathbf{val}(f) \leq \mathbf{cap}(S, T)$.

- ⊗ Wahr
- ○ Falsch

For a network (V, A, c, s, t), and any valid flow $f : A \to \mathbb{R}_0^+$, there is an $s - t$-cut with $\mathbf{val}(f) = \mathbf{cap}(S, T)$.

- ○ Wahr
- ⊗ Falsch

Wenn ein Graph mit mindestens zwei verschiedenen Knoten k-fach kantenzusammenhängend ist, dann gibt es für jedes Paar verschiedener Knoten k-fach kantendisjunkte Pfade zwischen ihnen.

- ⊗ Wahr
- ○ Falsch

Der Ford-Fulkerson-Algorithmus findet einen maximalen Fluss in einem Graphen in der Zeit $O(n + m)$.

- ○ Wahr
- ⊗ Falsch

COMMON MISTAKES: T4

1. Chebyshev auf $Pr[|\hat{X} - F^*| > \varepsilon F^*]$ direkt angewendet

   → macht folgenden Zwischenschritt und wendet dann Chebyshev an

   $$Pr[|\hat{X} - F^*| > \varepsilon F^*] \leq Pr[|\hat{X} - F^*| \geq \varepsilon F^*] \leq \dots$$

2. Fallunterscheidung in A, B und C, obwohl $A \cap B \cap C \neq \emptyset$

   $\varepsilon_i \in \{-1, 1\}$

   $$\mathbb{E}[X^4] = \sum_i \sum_j \sum_k \sum_\ell \mathbb{E}[\varepsilon_i \varepsilon_j \varepsilon_k \varepsilon_\ell] F_i F_j F_k F_\ell$$

   Wann ist $\mathbb{E}[\varepsilon_i \varepsilon_j \varepsilon_k \varepsilon_\ell] \neq 0$?

   ① $i = j$, $k = \ell$ $\longrightarrow N^2$

   ② $i = k$, $j = \ell$, $i \neq j$ $\longrightarrow N(N-1)$

   ③ $i = \ell$, $j = k$, $i \neq j$ $\longrightarrow N(N-1)$

FLÜSSE

Netzwerk $N = (V, A, c, s, t)$

- Knoten
- gerichtete Kanten
- s∈V, Quelle
- $c: A \to \mathbb{R}_0^+$ Kapazitätsfunktion
- t∈V, Senke

Definition 3.5. Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$. Ein Fluss in N ist eine Funktion $f : A \to \mathbb{R}$ mit den Bedingungen

$0 \le f(e) \le c(e)$ für alle $e \in A$, die *Zulässigkeit*, und

für alle $v \in V \setminus \{s, t\}$ gilt

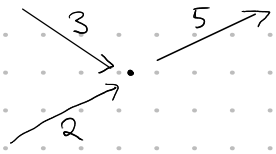
$$\sum_{u \in V:\,(u,v) \in A} f(u,v) = \sum_{u \in V:\,(v,u) \in A} f(v,u)$$

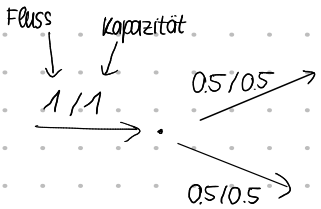
die *Flusserhaltung*.

Der *Wert* (engl.: value) eines Flusses f ist durch

$$\mathrm{val}(f) := \mathrm{netoutflow}(s) := \sum_{u \in V:\,(s,u) \in A} f(s,u) - \sum_{u \in V:\,(u,s) \in A} f(u,s)$$

definiert. Wir nennen f *ganzzahlig*, wenn $f(e) \in \mathbb{Z}\ \forall e \in A$.





Lemma 3.6. Der Nettozufluss der Senke gleicht dem Wert des Flusses, d.h.

$$\mathrm{netinflow}(t) := \sum_{u \in V:\,(u,t) \in A} f(u,t) - \sum_{u \in V:\,(t,u) \in A} f(t,u) = \mathrm{val}(f)\ .$$

Beweis:

$$0 = \sum_{(u,v) \in A} f(u,v) - \sum_{(v,u) \in A} f(v,u)$$

$$= \sum_{v \in V} \Big(\sum_{\substack{u \in V \\ (u,v) \in A}} f(u,v) - \sum_{\substack{u \in V \\ (v,u) \in A}} f(v,u) \Big)$$

$= 0$ für $v \in V \setminus \{s,t\}$

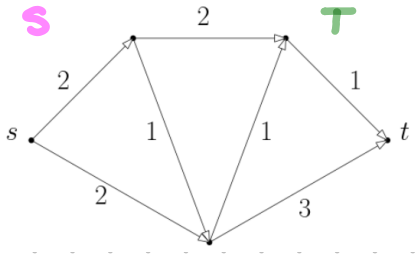
$$= \Big(\sum_{\substack{u \in V \\ (s,u) \in A}} f(s,u) - \sum_{\substack{u \in V \\ (u,s) \in A}} f(u,s) \Big) + \Big(\sum_{\substack{u \in V \\ (t,u) \in A}} f(t,u) - \sum_{\substack{u \in V \\ (u,t) \in A}} f(u,t) \Big)$$

$\mathrm{netoutflow}(s)$ $\quad$ $-\,\mathrm{netinflow}(t)$

maximaler Fluss und Schnitte

s-t-Schnitt ist Partition von V in (S,T) ($S \cup T = V$ und $S \cap T = \emptyset$), wobei $s \in S$ und $t \in T$

$$\mathrm{cap}(S,T) := \sum_{(u,w) \in (S \times T) \cap A} c(u,w)$$



Lemma: Ist f ein Fluss und (S,T) ein s-t-Schnitt in einem Netzwerk N, so gilt $val(f) \leq cap(S,T)$

Beweis:

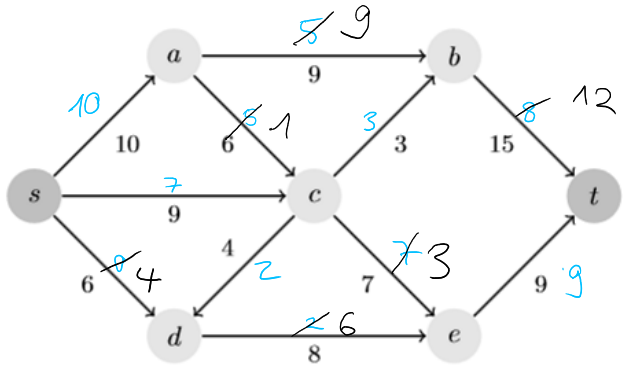
Satz 3.9 („Maxflow-Mincut Theorem"). Jedes Netzwerk $N = (V, A, c, s, t)$ erfüllt

$$\max_{\text{f Fluss in N}} val(f) = \min_{\text{(S,T) s-t-Schnitt in N}} cap(S,T)$$

Wie finden wir nun einen Maxflow?

Definition Restnetzwerk: (1) Falls $e \in A$ mit $f(e) < c(e)$, dann $e \in A_f$ mit $r_f(e) := c(e) - f(e)$
(2) Falls $e \in A$ mit $f(e) > 0$, dann $e^{opp} \in A_f$ mit $r_f(e^{opp}) = f(e)$

Die folgende Abbildung zeigt ein Netzwerk mit Quelle s und Senke t, wobei die Zahlen die Kapazitäten der Kanten angeben.



Auf einigen Netzwerkkanten ist eine nichtnegative Funktion f gegeben durch

(x,y)	(s,a)	(s,c)	(s,d)	(a,c)	(d,e)	(b,t)
$f(x,y)$	10	7	0	5	2	8

(a) Wie ist f auf alle übrigen Netzwerkkanten fortzusetzen, so dass f ein Fluss ist? Was ist der Wert dieses Flusses?

(b) Zeichnen Sie das Residualnetzwerk.

(c) Finden Sie einen augmentierenden Pfad von s nach t und erhöhen Sie den Fluss entlang dieses Pfades um den maximal möglichen Betrag. Falls nötig, dann iterieren Sie diesen Schritt, bis der so gefundene Fluss maximal ist.

Satz 3.11. Ein Fluss f in einem Netzwerk N ist ein maximaler Fluss gdw. es im Restnetzwerk N_f keinen gerichteten Pfad von der Quelle s zur Senke t gibt. Für jeden solchen maximalen Fluss gibt es einen s-t-Schnitt (S, T) mit $\mathrm{val}(f) = \mathrm{cap}(S, T)$.

FORD-FULKERSON(V, A, c, s, t)

1: $f \leftarrow \mathbf{0}$ ▷ Fluss konstant 0
2: **while** $\exists$ s-t-Pfad P in (V, A_f) **do** ▷ augmentierender Pfad
3: Erhöhe den Fluss entlang P ▷ wie in Beweis zu Satz 3.11
4: **return** f ▷ maximaler Fluss

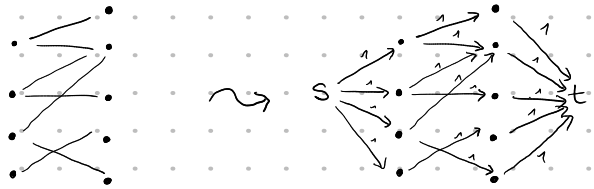
Satz 3.12. Sind in einem Netzwerk ohne entgegen gerichtete Kanten alle Kapazitäten ganzzahlig und höchstens U, so gibt es einen ganzzahligen maximalen Fluss, der in Zeit $O(mnU)$ berechnet werden kann (m ist die Anzahl Kanten, n die Anzahl Knoten im Netzwerk).

Capacity Scaling: Kapazitäten ganzzahlig und höchstens U $O(mn(1+\log n))$

Dynamic Trees: $O(mn\log n)$

ANWENDUNGEN VON FLÜSSEN

① Bipartites Matching

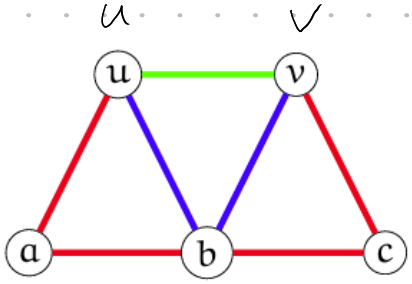


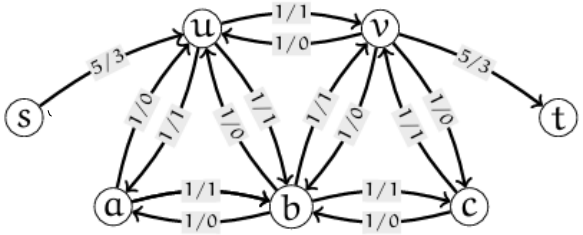
Lemma 3.15. Die maximale Grösse eines Matchings im bipartiten Graph G ist gleich dem Wert eines maximalen Flusses im Netzwerk N.

② Kanten- und knotendisjunkte Pfade

Wie viele intern kanten- (oder knoten-) disjunkte u-v-Pfade gibt es?

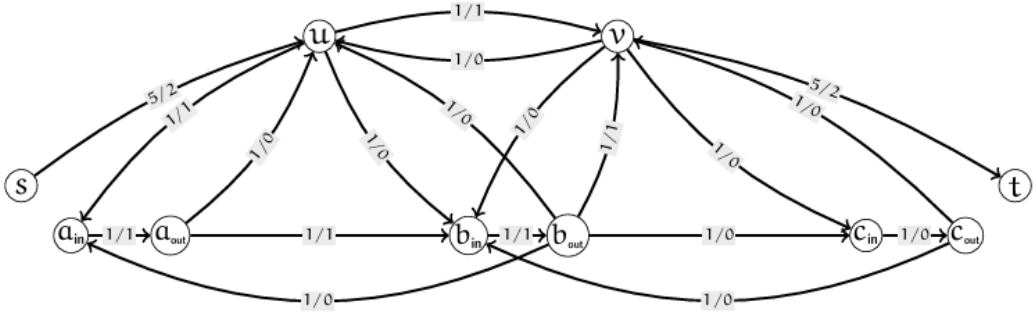
Netzwerkkonstruktion: für alle $\{u,v\} \in E$ füge (u,v) und (v,u) mit jeweils Kapazität 1 zu A hinzu
$(s,u), (v,t) \in A$, wobei Kapazität dieser Kanten $= |V|$





#kantendisjunkter Pfade = val(f)

für knotendisjunkte Pfade: $\forall x \in V \setminus \{u,v\}$ zwei neue Knoten x_{in} und x_{out}
alle Eingangskanten zu x_{in}, Ausgangskanten aus x_{out}
$(x_{in}, x_{out}) \in A$ mit $c(x_{in}, x_{out}) = 1$
→ wir können x nur einmal nutzen



③ Bildsegmentierung

Ziel: Unterteilung eines Bildes in Vordergrund (A) und Hintergrund (B)

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

(Vordergrund, Hintergrund, geteilte Teile)

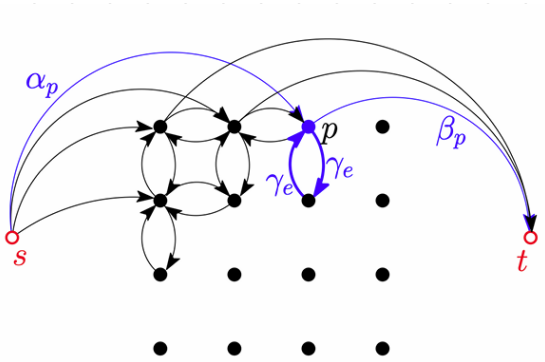
wenn wir eine Partition finden wollen, die $q(A,B)$ maximiert ist das equivalent zur Minimierung von

$$q'(A,B) = \sum_{p\in A} \beta_p + \sum_{p\in B} \alpha_p + \sum_{\substack{e\in E \\ |e\cap A|=1}} \gamma_e = cap(S,T)$$

$A := S\setminus\{s\}$
$B := T\setminus\{t\}$

(s,p), $p\in B$, α_p
(p,t), $p\in A$, β_p
(p,p'), $p\in A, p'\in B$, $\gamma_{(p,p')}$

Netzwerkkonstruktion:

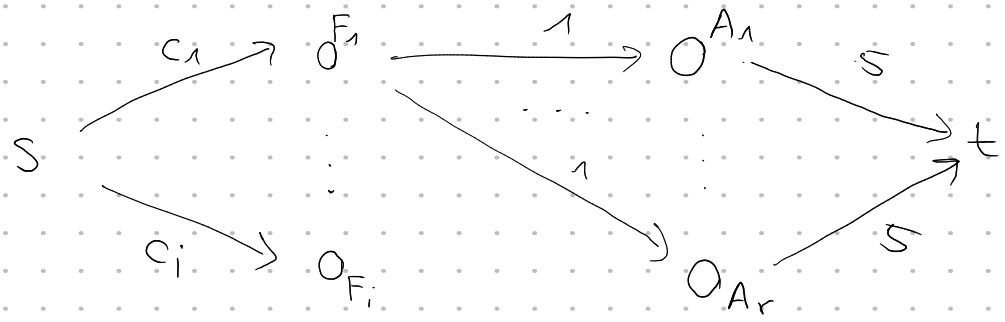


Aufgabe

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 verschiedenen 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 gibt eine mögliche Aufteilung wie oben genau dann wenn maxflow(N) hat eine gewisse (welche?) Eigenschaft.



$$val(f) = \sum_i c_i$$

MinCut

Problem: Gegeben einen Multigraphen G, bestimme die Kardinalität des minimalen Kantenschnitts $\mu(G)$

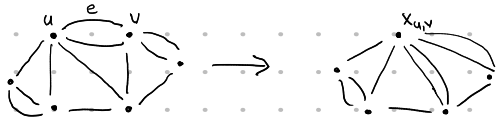
↑ Menge von Kanten C, so dass $(V, E \setminus C)$ nicht zusammenhängend ist

$\mu(G) \leq \min_{v \in V} \deg(v)$

mit Flow-Algorithmus: $O(n^2 m \log n)$

Kantenkontraktion

$e = \{u,v\}$ → Kontraktion von $e \hat{=}$ Verschmelzen von u und v zu $x_{u,v}$



$\deg_{G/e} x_{u,v} = \deg_G(u) + \deg_G(v) - 2 \cdot \#\text{Kanten zwischen } u \text{ und } v$

Lemma 3.20. Sei G ein Graph und e eine Kante in G. Dann gilt $\mu(G/e) \geq \mu(G)$ und falls es in G einen minimalen Schnitt C mit $e \notin C$ gibt, dann gilt $\mu(G/e) = \mu(G)$.

Zufällige Kantenkontraktionen

CUT(G)	G zusammenhängender Multigraph
1: **while** $\lvert V(G)\rvert > 2$ **do**	
2: $e \leftarrow$ gleichverteilt zufällige Kante in G	$O(n)$
3: $G \leftarrow G/e$	
4: **return** Grösse des eindeutigen Schnitts in G	

$O(n^2)$

Lemma 3.21. Sei $G = (V, E)$ ein Multigraph mit n Knoten. Falls e gleichverteilt zufällig unter den Kanten in G gewählt wird, dann gilt

$$\Pr[\mu(G) = \mu(G/e)] \geq 1 - \frac{2}{n}.$$

Beweis: $k := |C| = \mu(G)$

$$\Pr[\mu(G) = \mu(G/e)] \geq \Pr[e \notin C] = 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{\frac{kn}{2}} = 1 - \frac{2}{n}$$

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

$\hat{p}(G)$:= W'keit, dass CUT(G) den Wert $\mu(G)$ ausgibt

$\hat{p}(n) := \inf_{\substack{G=(V,E)\\ |V|=n}} \hat{p}(G)$, $\hat{p}(2) = 1$

Lemma 3.22. Es gilt für alle $n > 3$

$$\hat{p}(n) > (1 - 2/n) \cdot \hat{p}(n-1).$$

Beweis: $E_1 := (\mu(G) = \mu(G/e))$ $E_2 :=$ Ausgabe von CUT(G/e) ist $\mu(G/e)$

$$\hat{p}(G) = \Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr[E_2 | E_1] \geq (1 - \tfrac{2}{n}) \cdot \hat{p}(n-1)$$

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

$$= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$$

Satz 3.24. Für den Algorithmus der $\lambda\binom{n}{2}$-maligen Wiederholung von CUT(G) gilt:
↖ $O(n^3)$

(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 $\mu(G)$.

$$\text{Fehlerw'keit} = (1 - \hat{p}(n))^{\lambda\binom{n}{2}} \underset{1-x \leq e^{x}}{\leq} e^{-\hat{p}(n) \cdot \lambda\binom{n}{2}} \leq e^{-\lambda}$$

Für $\lambda = \ln(n)$: Fehlerw'keit $\leq \frac{1}{n}$
Laufzeit in $O(n^4 \ln n)$

Für $\lambda = 1$: Fehlerw'keit $\leq 1 - e^{-1}$
Laufzeit in $O(n^4)$

Den Rest schauen wir uns nächste Woche an!

Bootstrapping

Idee: Wiederhole den Teil, wo die Fehlerw'keit gross ist

Abbruch des Algorithmus nach t Knoten,
dann Wiederholen des Algorithmus für t Knoten in $O(t^4)$ und Erfolgsw'keit $\geq \frac{e-1}{e}$

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

$\lambda \cdot \frac{1}{\overline{p}_t(n)}$ Wiederholungen des ganzen Algorithmus:

Fehlerw'keit $\leq e^{-\lambda}$

Laufzeit: $O(\underbrace{\lambda \underbrace{\frac{n(n-1)e}{t(t-1)(e-1)}}_{\overline{p}_t(n)}}_{\#\text{ wdh}} \cdot \underbrace{(n(n-t) + t^4)}_{\text{kontrahieren bis } t \text{ Knoten übrig}}) = O(\lambda(\frac{n^4}{t^2} + n^2t^2))$

mögeich bis zu $O(n^2 \text{polylog}(n))$ - Algorithmus

Aufgaben

1) $\hat{p}(K_3) = \frac{2}{3}$

2) Falls G einen minimalen Schnitt C mit $e \in C$ hat, dann gilt $\mu(G \setminus e) = \mu(G) + 1$.

3) Falls G einen minimalen Schnitt C mit $e \notin C$ hat, dann gilt $\mu(G \setminus e) = \mu(G)$.

4) Wenn der minimale Schnitt von G k ist, dann ist G k-kanten-zusammenhängend.