Algorithms and Probability

Week 13

Georg Hasebe — 23/05/2024

Exercise Sheet 6

Minitest

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.

Nur, wenn alle Kantenkapazitäten ganzzahlige Werte aufweisen.

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.

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.

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

Welche der folgenden Punkte treffen zu?

Es gibt einen s-t-Schnitt (S, T) und einen Fluss f, so dass

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

Jedes Netzwerk erfüllt

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

Maxflow-Mincut Theorem.

Jedes Netzwerk erfüllt

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

Es gibt einen s-t-Schnitt (S, T) und einen Fluss f, so dass

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

Maxflow-Mincut Theorem.

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?

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

Nur wenn f nicht maximal ist.

$t \in X$.

Nur wenn f maximal ist.

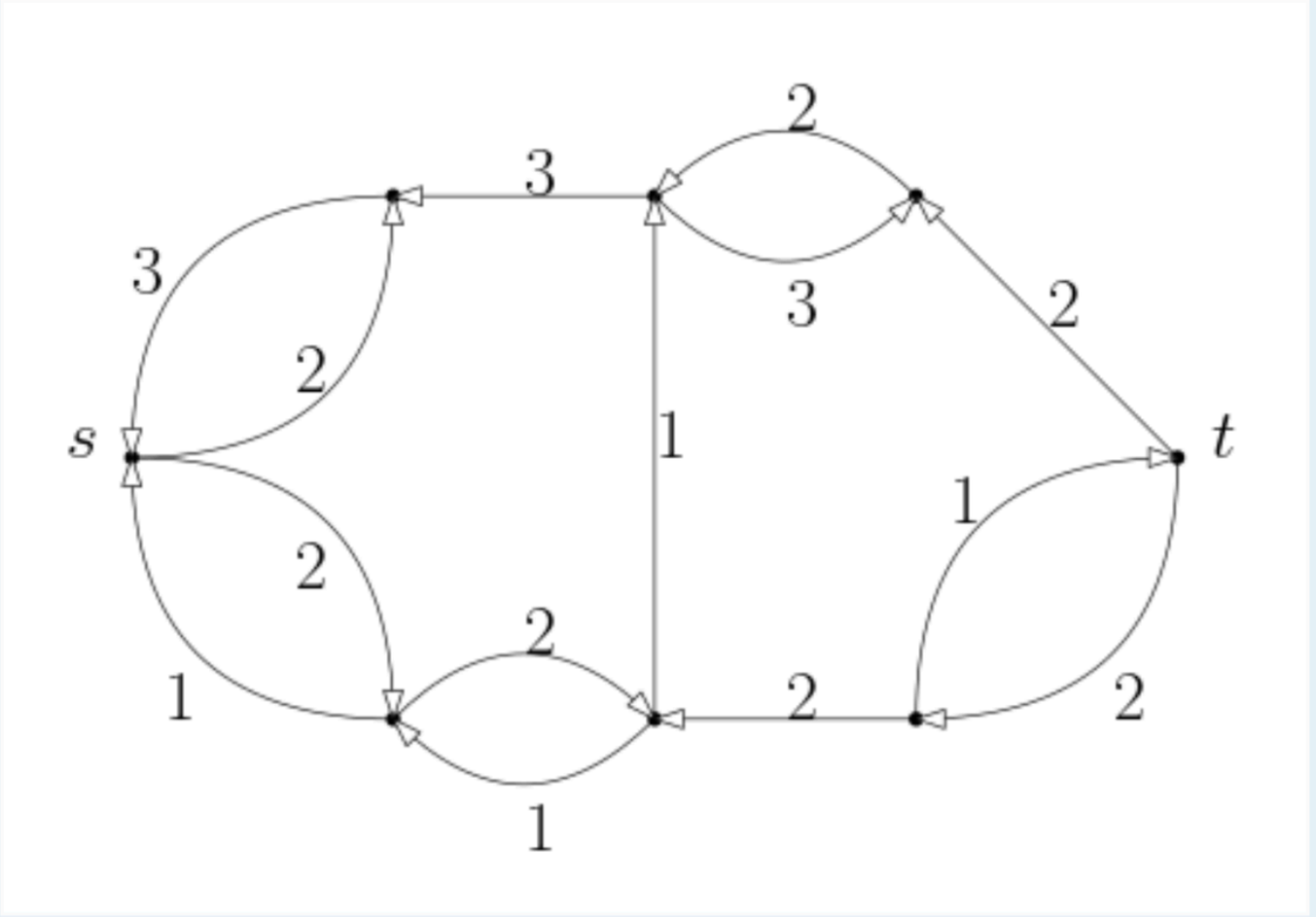
$(X, V \setminus X)$ ist ein s-t Schnitt.

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)

- $\leq$ ✓
- $=$ ✗
- $\geq$ ✗

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:

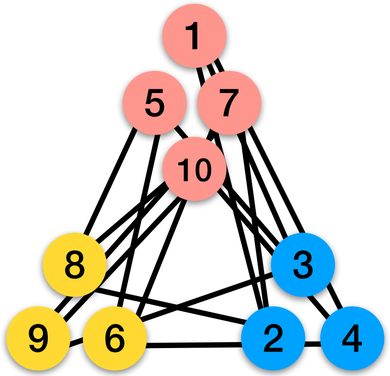


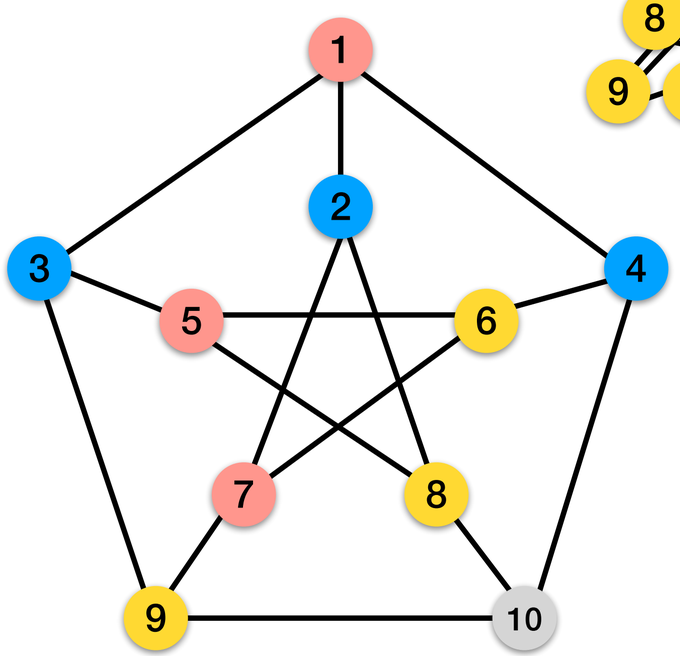
Ist der Fluss maximal?

Es gibt keine s-t Pfad in N_f.

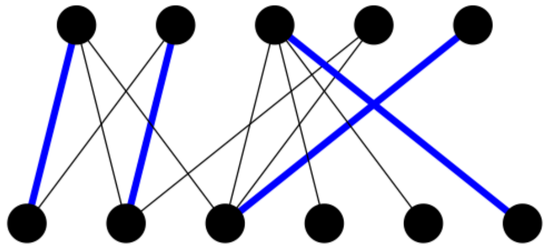
Sei $N = (V, A, c, s, t)$ ein Netzwerk. Wenn c nur ganzzahlige Kantengewichte hat, so ist jeder maximale Fluss in N ganzzahlig.

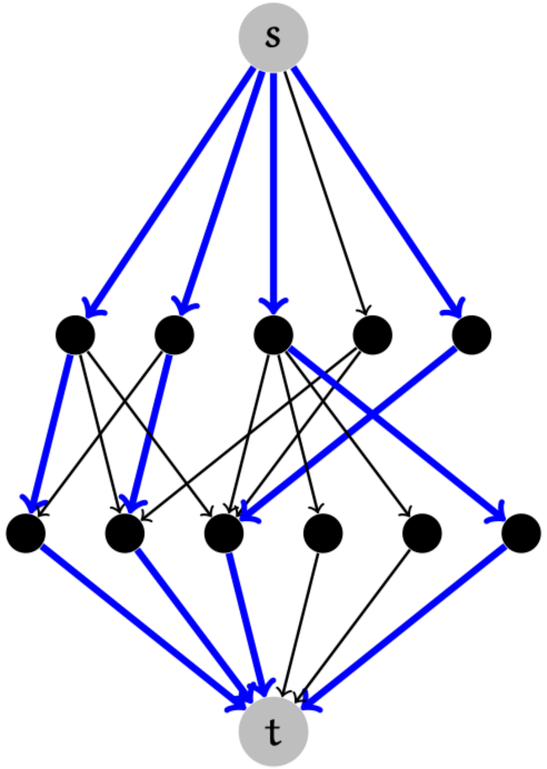
Für jeden 2-färbbaren Graph G können wir mithilfe eines Flussproblems herausfinden ob G ein perfektes Matching hat.





Graphen mit chromatischer Zahl $\chi(G) = k$ nennt man auch k-partit. 2 -färbbare Graphen sind also bipartit.



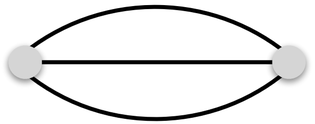


Theory Recap

Definitions

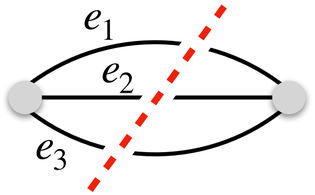
MIN-CUT Problem. Gegeben ein Multigraph G, bestimme die Kardinalität eines *minimalen Kantenschnitts*.

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.)



Kantenschnitt in einem Multigraph $G = (V, E)$: Menge von Kanten C, so dass $(V, E \setminus C)$ unzusammenhängend ist.

(Kantenmenge $C \subseteq E$ vs. Partition (S, T) von V)



$C = \{e_1, e_2, e_3\}$

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

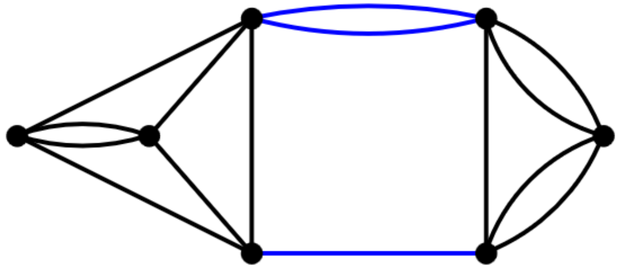
$$\mu(G) := \min_{C \subseteq E, (V, E \setminus C) \text{ unzusammenhängend}} |C|$$

Beispiele

MIN-CUT Problem. Gegeben ein Multigraph G, bestimme $\mu(G)$.

Für G nicht zusammenhängend gilt $\mu(G) = 0$.

$\mu = 4$

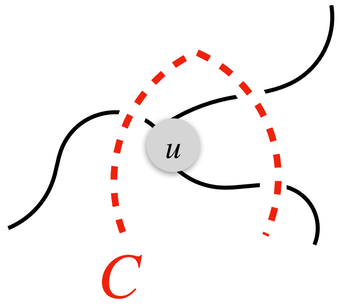


$\mu = 3$

Bemerkungen

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

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



Assuming u has the smallest degree in G, we can just remove all edges incident with u.

Kantenkontraktion

Gegeben $G = (V, E)$, $e = \{u, v\} \in E$.

Kontraktion von e: Verschmilzt u und v zu einem neuen Knoten $x_{u,v}$, der nun zu allen Kanten inzident ist, zu denen u oder v inzident war. Die Kanten zwischen u und v verschwinden.

Den entstehenden Graph bezeichnen wir mit G/e.

Zufällige Kantenkontraktion

CUT(G) — G zusammenhängender Multigraph

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

Zufällige Kantenkontraktion

```
CUT(G)                                   G zusammenhängender Multigraph
1: while |V(G)| > 2 do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G
```

Wir wollen den kleinstmöglichen Kantenschnitt $\mu(G)$.

Wie verändert sich dieser? Mit anderen Worten, in welcher Relation stehen $\mu(G)$ und $\mu(G/e)$?

Es stellt sich heraus: $\mu(G/e) \geq \mu(G)$.

Beweis an der Tafel.

Zufällige Kantenkontraktion

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 an der Tafel.

Erfolgswahrscheinlichkeit

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

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

Lemma 3.22. Es gilt für alle $n \geq 3$

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

Beweis im Skript.

Erfolgswahrscheinlichkeit

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 3.23. Es gilt $\hat{p}(n) \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$ für alle $n \geq 2$.

Laufzeit

CUT(G)	G zusammenhängender Multigraph

```
1: while |V(G)| > 2 do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G
```

Sei n Anzahl der Konten in G. Für die Implementierung setzen wir folgendes voraus:

- Eine Kantenkontraktion kann in $O(n)$ Zeit durchgeführt werden.
- Eine gleichverteilt zufällige Kante in G kann in $O(n)$ gewählt werden.

(Das ist nicht ganz offensichtlich, und erfordert u.a., dass Mehrfachkanten mittels Kantengewichten dargestellt werden.) Mit dieser Voraussetzung können wir CUT(G) mit einer Laufzeit von $O(n^2)$ implementieren.

Resultat

Satz 3.24. Für den Algorithmus der $\lambda\binom{n}{2}$-maligen Wiederholung von $\textsc{Cut}(G)$ gilt:

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

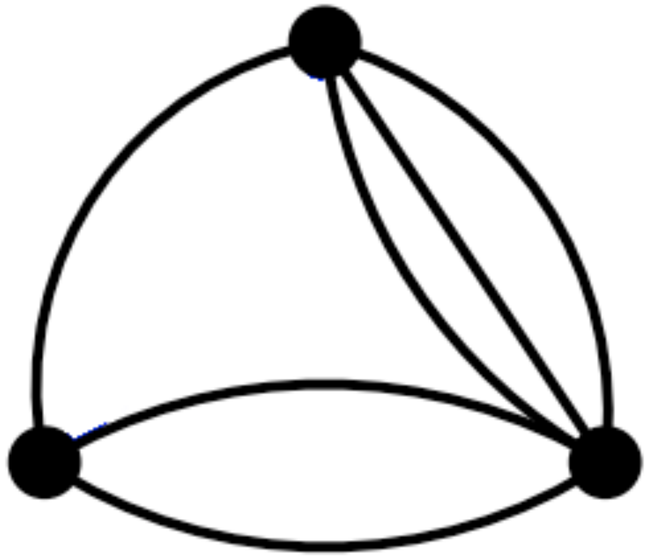
(2) Der kleinste angetroffene Wert ist mit einer Wahrscheinlichkeit von mindestens $1 - e^{-\lambda}$ gleich $\mu(G)$.

Zu (1): eine Ausführung von Cut(G) kann in $O(n^2)$ ausgeführt werden.
Zu (2):

$$(1-\hat{p}(n))^{\lambda\binom{n}{2}} \leq \left(1 - 1/\binom{n}{2}\right)^{\lambda\binom{n}{2}} \leq e^{-\lambda}$$

wobei $\forall x \in \mathbb{R} : 1 + x \leq e^x$ verwendet wurde.

Betrachten Sie den folgenden Multigraphen G. Was ist die Wahrscheinlichkeit, dass Cut(G) die richtige Antwort zurück gibt (nach einem Aufruf)? Bitte geben sie ihre Antwort als Dezimalzahl ein (x.xx).

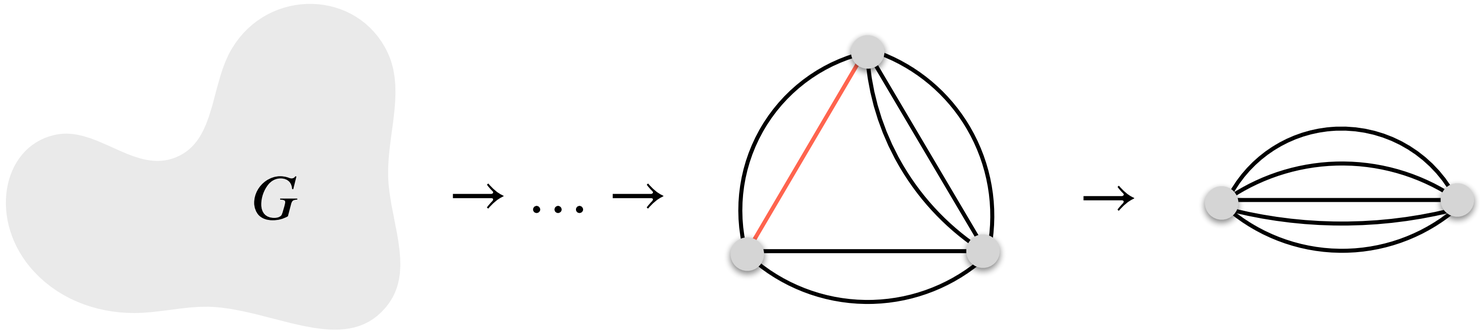


0.50

Anzahl Kanten ist 6, 3 davon liefern richtiges Ergebnis.
Zur Erinnerung: $\hat{p}(3) \geq 1 - 2/3 \approx 0.33$, das ist aber nur eine untere Schranke, in diesem Fall können wir das Ergebnis explizit berechnen.

Bootstrapping

Wir erinnern uns daran, dass, wenn C ein minimaler Schnitt ist, dann gilt
$e \notin C \Longrightarrow \mu(G/e) = \mu(G)$.



Ein Multigraph mit vielen Knoten und Kanten. Die Wahrscheinlichkeit, hier zufällig eine Kante zu wählen, die Teil des minimalen Schnittes ist, ist sehr gering.

Falls wir den minimalen Schnitt bis hier hin erhalten konnten, ist die Wahrscheinlichkeit, hier zufällig eine Kante zu wählen, die Teil des minimalen Schnittes ist, ist sehr hoch.

Bootstrapping

Wir erinnern uns daran, dass $\hat{p}(n)$ die Erfolgswahrscheinlichkeit ist. Wir wollen also, dass $\hat{p}(n)$ möglichst nahe an 1 ist.

$$\hat{p}(n) \geq \underbrace{\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2}}_{\text{Für sehr grosse } n\text{, sehr nahe an } 1.} \cdots \underbrace{\frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}}_{0.6,\, 0.5,\, 0.33\ldots} \cdot \underbrace{\hat{p}(2)}_{=1} = \frac{2}{n(n-1)}$$

Bei 3 Knoten liefert eine zufällige Kantenkontraktion nur mehr eine Erfolgswahrscheinlichkeit von $1/3$. Wieso dieses Risiko eingehen, um eventuell unser Resultat zu verschlechtern? Bei 3 Knoten können wir das Resultat schnell bestimmen. Auch bei $4,5,\ldots,t$.

Bootstrapping

Idee: Wir brechen Cut(G) ab, wenn wir bei einem Multigraphen mit t Knoten angelangt sind.

Danach lösen wir den Rest mit einem anderen Algorithmus in Laufzeit $z(t)$.

```
CUT1(G)                                          G zusammenhängender Multigraph
1: while |V(G)| > t do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G          ▷ in Zeit O(z(t))
```

```
CUT1(G)                                   G zusammenhängender Multigraph
1: while |V(G)| > t do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G        ▷ in Zeit O(z(t))
```

Bootstrapping

Welchen Algorithmus wählen wir im letzten Schritt?

(1) Deterministisch mit Flüssen in Laufzeit $O(t^4 \log t)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1$.

(2) Eine Wiederholung ($\lambda = 1$) von Cut(G) in Laufzeit $O(t^4)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1 - e^{-1}$.

CUT1(G) — G zusammenhängender Multigraph

1: **while** $|V(G)| > t$ **do**
2: $e \leftarrow$ gleichverteilt zufällige Kante in G
3: $G \leftarrow G/e$
4: **return** Grösse des eindeutigen Schnitts in G ▷ in Zeit $O(z(t))$

Bootstrapping

(2) Eine Wiederholung ($\lambda = 1$) von Cut(G) in Laufzeit $O(t^4)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1 - e^{-1}$.

Die Laufzeit von Cut1(G) ist $O(\underbrace{n \cdot (n-t)}_{\text{Nimm wieder an, dass Kantenkontraktion in } O(n).} + t^4)$, die Erfolgswahrscheinlichkeit ergibt sich durch

$$\hat{p}(n) \geq \underbrace{\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{t+1}{t+3} \cdot \frac{t}{t+2} \cdot \frac{t-1}{t+1}}_{\text{Führe Cut(G) aus bis wir einen Multigraphen mit } t \text{ Knoten haben.}} \cdot \underbrace{p^*(t)}_{\text{Führe (2) aus.}} = \frac{t(t-1)}{n(n-1)} \cdot p^*(t) \,.$$

```
CUT1(G)                                   G zusammenhängender Multigraph
1: while |V(G)| > t do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G        ▷ in Zeit O(z(t))
```

Bootstrapping

(2) Eine Wiederholung ($\lambda = 1$) von Cut(G) in Laufzeit $O(t^4)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1 - e^{-1}$.

Cut1(G) mit Laufzeit $O(n \cdot (n-t) + t^4)$ und Erfolgswskt. $\hat{p}(n) \geq \frac{t(t-1)}{n(n-1)} \cdot p^*(t)$.

Bei $\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1}$ Wiederholungen von Cut1(G), erreichen wir eine Erfolgswskt. von $1 - e^{-\lambda}$ und eine Laufzeit von $O\left(\lambda\left(\frac{n^4}{t^2} + n^2 t^2\right)\right)$.

```
CUT1(G)                                   G zusammenhängender Multigraph
1: while |V(G)| > t do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G        ▷ in Zeit O(z(t))
```

Bootstrapping

(2) Eine Wiederholung ($\lambda = 1$) von Cut(G) in Laufzeit $O(t^4)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1 - e^{-1}$.

Cut1(G) bei $\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1}$ Wiederholungen liefert Erfolgswskt. von $1 - e^{-\lambda}$

und eine Laufzeit von $O\left(\lambda\left(\frac{n^4}{t^2} + n^2 t^2\right)\right)$.

Für $t = n^{1/2}$ erhalten wir so $O(\lambda n^3)$.

```
CUT1(G)                                   G zusammenhängender Multigraph
1: while |V(G)| > t do
2:     e ← gleichverteilt zufällige Kante in G
3:     G ← G/e
4: return Grösse des eindeutigen Schnitts in G        ▷ in Zeit O(z(t))
```

Bootstrapping

Wir haben damit die Laufzeit von $O(\lambda n^4)$ auf $O(\lambda n^3)$ verbessern können.

ABER: Nun haben wir für den letzten Schritt in Zeile 4 eine neue Möglichkeit hinzugewonnen:

(3) Eine Wiederholung ($\lambda = 1$) von Cut1(G) in Laufzeit $O(t^3)$ und Erfolgswahrscheinlichkeit $p^*(t) = 1 - e^{-1}$.

Dieses in sich selbst einsetzen eines Algorithmus, um diesen immer schneller zu machen, nennt man **Bootstrapping**.

Welche der folgenden Aussagen ist korrekt bezüglich des in der Vorlesung vorgestellten Algorithmus zum berechnen eines minimalen Schnittes?

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.

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.

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