Algorithms and Probability

Week 11

G08 - mkilic

15.V.2025

Overview

1. **Minimale Schnitte**

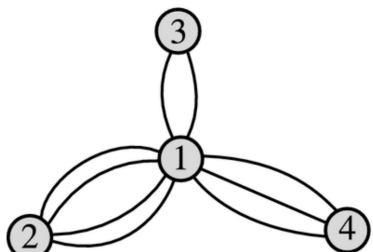
2. **Kleinster umschliessender Kreis**

Roadmap

1. Graphentheorie
   - Zusammenhang
   - Kreise
   - Matchings
   - Färbungen
2. W'keitstheorie
   - Bedingte W'keit
   - Unabhängigkeit
   - (mehrere) Zufallsvariablen
   - Diskrete Verteilungen
   - Abschätzen von W'keiten
   - Randomisierte Algorithmen
3. Algorithmen
   - Lange-Bunte Pfade
   - MaxFlow
   - MinCut
   - Kleinster umschliessender Kreis
   - Konvexe Hülle

Minimale Schnitte - Einführung

`MIN-CUT Problem`: Gegeben ein Multigraph G, bestimme $\mu(G)$: Die Kardinalität eines minimalen Schnitts.



Wir handlen ungerichteten ungewichteten Multigraphen $G = (V, E)$ ohne Schleifen.

Minimale Schnitte - Einführung

Kantenschnitt

In einem Multigraph $G = (V, E)$, nennen wir eine Menge $C \subseteq E$ für die $(V, E \setminus C)$ unzusammenhängend ist, ein Kantenschnitt.

Mit $\mu(G)$ bezeichnen wie die Kardinalität eines kleinstmöglichen Kantenschnitts in G

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

Erste Lösung

Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$

Erste Lösung

Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$
1. Statt Mehrfachkanten können wir ganzzählige Kantengewichte verwenden.

Erste Lösung

Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$
1. Statt Mehrfachkanten können wir ganzzählige Kantengewichte verwenden.
2. Ungerichtete Kanten $\{u, v\}$ können wir duch Paare gerichtete Kanten $(u, v), (v, u)$ ersetzen. Jetzt haben wir einen gerichteten Graph (V, A).

Erste Lösung

Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$
1. Statt Mehrfachkanten können wir ganzzählige Kantengewichte verwenden.
2. Ungerichtete Kanten $\{u, v\}$ können wir duch Paare gerichtete Kanten $(u, v), (v, u)$ ersetzen. Jetzt haben wir einen gerichteten Graph (V, A).
3. Wir fixieren einen Knoten $s \in V$ und berechnen minimale s-t-Schnitte, für alle $t \in V \setminus \{s\}$ im Netzwerk (V, A, w, s, t).

Erste Lösung

Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$
1. Statt Mehrfachkanten können wir ganzzählige Kantengewichte verwenden.
2. Ungerichtete Kanten $\{u, v\}$ können wir duch Paare gerichtete Kanten $(u, v), (v, u)$ ersetzen. Jetzt haben wir einen gerichteten Graph (V, A).
3. Wir fixieren einen Knoten $s \in V$ und berechnen minimale s-t-Schnitte, für alle $t \in V \setminus \{s\}$ im Netzwerk (V, A, w, s, t).
4. Der Kleinste dieser s-t-Schnitte zeigt uns den minimalen Schnitt in unserem originalen Graph G.

Erste Lösung

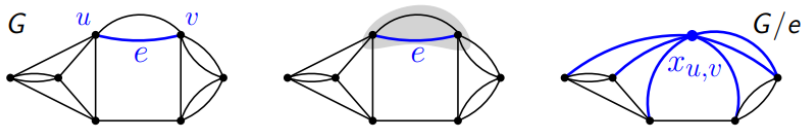
Wir können unsere Strategie aus maxFlow-minCut benutzen.

0. Wir haben einen Multigraph $G = (V, E)$
1. Statt Mehrfachkanten können wir ganzzählige Kantengewichte verwenden.
2. Ungerichtete Kanten $\{u, v\}$ können wir duch Paare gerichtete Kanten $(u, v), (v, u)$ ersetzen. Jetzt haben wir einen gerichteten Graph (V, A).
3. Wir fixieren einen Knoten $s \in V$ und berechnen minimale s-t-Schnitte, für alle $t \in V \setminus \{s\}$ im Netzwerk (V, A, w, s, t).
4. Der Kleinste dieser s-t-Schnitte zeigt uns den minimalen Schnitt in unserem originalen Graph G.

Laufzeit: $(n-1)$ mal $O(mn(1 + \log U))$ oder $O(mn^2 \log n) = O(n^4 \log n)$
Wir nehmen $m = O(n^2)$ weil im schlimmsten Fall hat jeder Knoten eine gerichtete Kante zu allen anderen Knoten.

Kantenkontraktionen

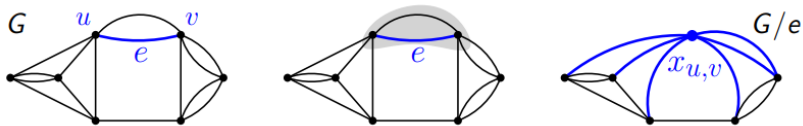
Nochmal sei $G = (V, E)$ ein Multigraph und $\{u, v\} \in E$. Die *Kontraktion* von e macht aus die beiden Knoten u und v einen neuen Knoten $x_{u,v}$, der nun zu allen Kanten inzident ist, zu denen u oder v inzident war, ausser den Kanten zwischen u und v – diese Kanten verschwinden.



[0]Achtung! Es ist möglich, dass $deg(v) \neq |N(v)|$

Kantenkontraktionen

Nochmal sei $G = (V, E)$ ein Multigraph und $\{u, v\} \in E$. Die *Kontraktion* von e macht aus die beiden Knoten u und v einen neuen Knoten $x_{u,v}$, der nun zu allen Kanten inzident ist, zu denen u oder v inzident war, ausser den Kanten zwischen u und v – diese Kanten verschwinden.



Notation: G/e. Ist k die Anzahl der Kanten zwischen u und v, dann gilt

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

und wir haben $|E(G/e)| = |E(G)| - k$.

[0]Achtung! Es ist möglich, dass $deg(v) \neq |N(v)|$

Minimaler Schnitte in G und G/e

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

- μ kann durch Kontraktion einer Kante e **nie fallen**.
- μ bleibt gleich, falls es einen minimalen Schnitt ohne e gibt.

Man spricht von einer natürlichen Bijektion. Entweder bleibt eine Kante in G bei der Kontraktion gleich, oder $\{w, u\}$ oder $\{w, v\}$ wird zu einer Kante $\{w, x_{u,v}\}$.

Erster randomisierter Algorithmus

$\textsc{Cut}(G)$ *G zusammenhängender Multigraph*

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

Sei $|V| = n$, dann machen wir zwei Annahmen:

- Eine Katnenkontraktion kann in $O(n)$ Zeit durchgeführt werden.
- Eine gleichverteilte zufällige Kante in G kann in $O(n)$ gewählt werden.[1]

Insgesamt können wir $\texttt{Cut}(G)$ mit Laufzeit $O(n^2)$ implementieren.

[1] Das erfordert Darstellung der Mehrfachkanten durch Kantengewichte.

Erhalt von μ

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

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

- 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.

Beweis: Lemma 3.21

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

□

Beweis: Lemma 3.21

- Sei C ein minimaler Schnitt in G und $k := |C| = \mu(G)$
- Sicherlich ist der Grad jedes Knotens in G mindestens k, da zu einem Knoten inzidenten Kanten immer einen Schnitt bilden: $\forall v \in V : \deg_G(v) \geq k$

□

Beweis: Lemma 3.21

- Sei C ein minimaler Schnitt in G und $k := |C| = \mu(G)$
- Sicherlich ist der Grad jedes Knotens in G mindestens k, da zu einem Knoten inzidenten Kanten immer einen Schnitt bilden: $\forall v \in V : \deg_G(v) \geq k$
- Es gilt daher: $|E| := \frac{1}{2} \sum_{v \in V} \deg(v) \geq \frac{kn}{2}$

□

Beweis: Lemma 3.21

- Sei C ein minimaler Schnitt in G und $k := |C| = \mu(G)$
- Sicherlich ist der Grad jedes Knotens in G mindestens k, da zu einem Knoten inzidenten Kanten immer einen Schnitt bilden: $\forall v \in V : \deg_G(v) \geq k$
- Es gilt daher: $|E| := \frac{1}{2} \sum_{v \in V} \deg(v) \geq \frac{kn}{2}$
- Wir wissen $e \notin C \Rightarrow \mu(G/e) = \mu(G)$ und somit

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

□

Erfolgswarscheinlichkeit von `Cut`

Wir interessieren uns für die Erfolgswahrscheinlichkeit

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

Wir definieren diese W'keit für unsere Analyse über Anzahl Knoten in G

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

[1] z.Bsp. $\hat{p}(2) = 1$

Erfolgswarscheinlichkeit von `Cut`

Wir interessieren uns für die Erfolgswahrscheinlichkeit

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

Wir definieren diese W'keit für unsere Analyse über Anzahl Knoten in G

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

Lemma 3.22.

Es gilt für alle $n \geq 3$

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

[1] z.Bsp. $\hat{p}(2) = 1$

Erfolgswharscheinlichkeit von`Cut`

Wenn wir jetzt dieses Lemma benutzen können wir $\hat{p}(n)$ für beliebiges n berechnen:

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

Erfolgswharscheinlichkeit von`Cut`

Wenn wir jetzt dieses Lemma benutzen können wir $\hat{p}(n)$ für beliebiges n berechnen:

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

Lemma 3.23.

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

Wir wiederholen unseren Algorithmus um die Fehlerw'keit zu reduzieren.

Analyse von `Cut`

Satz 3.24

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

Das ist für $\lambda = \ln n$ gleiche Laufzeit wie der deterministischer Algorithmus. Geht es besser?

Bootstrapping

Wir mahcen die meisten Fehler in den letzten Schritten. Der Schritt von 3 auf 2 Knoten ist mit W'keit 66% ein Fehler. Deswegen brechen wir unseren Algorithmus für eine fixe Grösse des Graphen $|V| = t$ ab und benutzen einne anderen Algorithmus $z(t)$ für den Graphen G' der jetzt t Knoten hat. Konkret heisst das:

CUT1(G) — G zusammenhängender Multigraph

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

Analyse von `Cut1`

`Cut1` läuft in Zeit $O(n(n-t)+z(t))$. Wenn wir annehmen dass $z(t)$ eine Erfolgsw;keit von $p^*(t)$ hat dann hat `Cut1` insgesamt eine Erfolgsw'keit von

$$p_{\texttt{Cut1}} \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{t+1}{t+3} \cdot \frac{t}{t+2} \cdot \frac{t-1}{t+1} \cdot p^*(t) = \frac{t(t-1)}{n(n-1)} \cdot p^*(t)$$

Wenn wir den zweiten Algorithmus $z(t)$ als unseren Algorithmus aus Satz 3.24 wählen und ihn nur 1-mal wiederholen, dann haben wir $z(t) = O(t^4)$ und $p^*(t) = 1 - e^{-1}$. Also haben wir

$$p_{\texttt{Cut1}} = \frac{t(t-1)}{n(n-1)} \cdot \frac{e-1}{e}$$

Analyse von `Cut1`

Jetzt wiederholen wir unseren Algorithmus $\lambda \cdot \frac{1}{1-p_{\texttt{Cut1}}}$-mal um die Erfolgsw'keit auf $1 - e^{-\lambda}$ zu erhöhen. Insgesamt haben wir also eine Laufzeit von

$$\lambda \frac{n(n-1)}{t(t-1)} \frac{e}{e-1} \cdot O(n(n-t) + t^4) = O\left(\lambda\left(\frac{n^4}{t^2} + n^2 t^2\right)\right)$$

Wir können $t = \sqrt{n}$ wählen um eine Laufzeit von $O(\lambda n^3)$ zu erhalten.

Jetzt können wir aber diesen schnelleren Algorithmus als eine Subroutine benutzen. Dann können wir den gleichen Prozess immer wieder durchgehen. Das nennt man *Bootstrapping*.

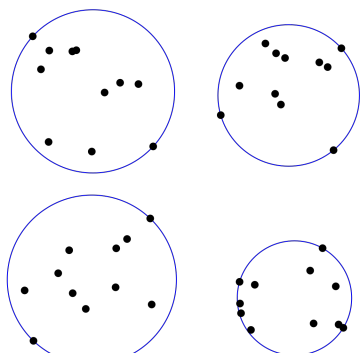
Es konvergiert zu einem Verfahren mit einer Laufzeit von $O(n^2 poly(\log n))$

Kleinster umschliessender Kreis

Problem: Zu einer gegebenen Menge P von n Punkten in der Ebene möchten wir einen Kreis $C(P)$ bestimmen, sodass $C(P)$ alle Punkte aus P umschliesst und der *Radius* von $C(P)$ so klein wie möglich ist.

Notation: $C^\bullet$ bezeichnet die geschlossene von C berandete Kreisscheibe.

Wir sagen C umschliesst P, falls $P \subseteq C^\bullet$

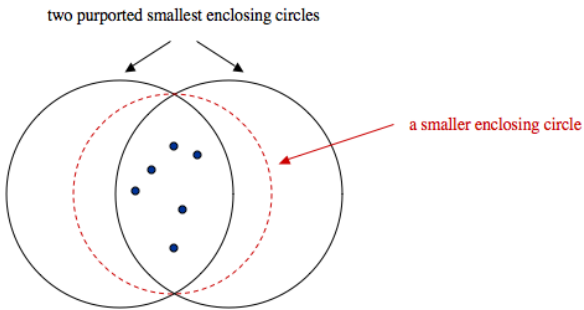


Existenz und Eindeutigkeit

Lemma 3.25

Für jede (endliche) Punktemenge P im $\mathbb{R}^2$ gibt es einen eindeutigen kleinsten umschliessenden Kreis $C(P)$.

Beweisskizze:



Lemma 3.26

Lemma 3.26

Für jede (endliche) Punktemenge P im $\mathbb{R}^2$ mit $|P| \geq 3$ gibt es eine Teilmenge $Q \subseteq P$, so dass $|Q| = 3$ und $C(Q) = C(P)$.

Beweisidee: Ist g eine beliebige Gerade durch den Mittelpunkt von C, so kann es nicht sein, dass alle Punkte von $B := \{$ Punkte von P die auf dem Rand von C liegen$\}$ strikt auf einer der beiden Seiten von g liegen.

Erster Algorithmus

CompleteEnumeration(P)

0: **for all** $Q \subseteq P$ mit $|Q| = 3$ **do**
0: bestimme $C(Q)$
0: **if** $P \subseteq C^\bullet(Q)$ **then**
0: **return** $C(Q)$

Die Laufzeit ist $O(\binom{n}{3} \cdot n) = O(n^4)$.

- $\binom{n}{3}$ viele Mengen Q
- Für jede solche Menge können wir $C(Q)$ in konstanter Zeit bestimmen.
- Wir müssen für jeden Punkt testen ob er in $C^\bullet(Q)$ liegt in $O(n)$

Erster randomisierter Algorithmus

RADNOMISED_PRIMITIVEVERSION(P)

0: **repeat forever**
0: wähle $Q \subseteq P$ mit $|Q| = 3$ zufällig und gleichverteilt
0: bestimme $C(Q)$
0: **if** $P \subseteq C^{\bullet}(Q)$ **then**
0: **return** $C(Q)$

Dieser Algorihmus hat aber im Erwartungswert die gleiche Laufzeit wie der deterministischer Algorithmus weil der Erwartungswert der geometrischverteilten Anzahl iterationen $\binom{n}{3}$ ist und wir immer noch für alle Punkte testen müssen, ob er in $C^{\bullet}(Q)$ liegt.

Der bessere randomisierte Algorithmus

RADNOMISED_CLEVERVERSION(P)

0: **repeat forever**
0: wähle $Q \subseteq P$ mit $|Q| = 11$ zufällig und gleichverteilt
0: bestimme $C(Q)$
0: **if** $P \subseteq C^{\bullet}(Q)$ **then**
0: **return** $C(Q)$
0: verdopple alle Punkten von P ausserhalb von $C(Q)$

[1]Wir können die gewünschte Teilmenge Q aus P in Zeit $O(n)$ zufällig und gleichverteilt wählen. siehe Skript s.201

Analyse vom Algorithmus

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

$$3 \cdot \frac{n-r}{r+1} \leq 3 \cdot \frac{n}{r+1}.$$

Beweis von Lemma 3.28

Für den Beweis definieren wir uns zwei Hilfsfunktionen, für $p \in P$, $R, Q \subseteq P$:

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

$$\text{essential}(p, Q) := \begin{cases} 1 & \text{falls } C(Q \setminus \{p\}) \neq C(Q) \\ 0 & \text{sonst} \end{cases}$$

Die beiden Funktionen stehen in folgender Relation:

$$\text{out}(p, R) = 1 \quad \Longleftrightarrow \quad \text{essential}(p, R \cup \{p\}) = 1.$$

Beweis von Lemma 3.28

Sei X die Anzahl Punkte ausserhalb von $C(R)$. Dann haben wir:

$$\begin{aligned}
\mathbb{E}[X] &= \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{s \in P \setminus R} \text{out}(s, R) \\
&= \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{s \in P \setminus R} \text{essential}(s, R \cup \{s\}) \\
&= \frac{1}{\binom{n}{r}} \sum_{Q \in \binom{P}{r+1}} \sum_{p \in Q} \text{essential}(p, Q) \\
&\leq \frac{1}{\binom{n}{r}} \sum_{Q \in \binom{P}{r+1}} 3 = 3 \cdot \frac{\binom{n}{r+1}}{\binom{n}{r}} = 3 \cdot \frac{n-r}{r+1}.
\end{aligned}$$

□

[1] $\textit{essential}(p, Q) \leq 3$

Analyse von Algorithmus

Satz 3.29

Algorithmus `Randomized_CleverVersion` berechnet den kleinsten umschliessenden Kreis von P in erwarteter Zeit $O(n \log n)$.

Laufzeitanalyse

Sei T die Anzahl Iterationen und X_k die Anzahl Punkte nach k Iterationen. $X_0 = n$. Den Erwartungswert von X_k können wir mit Hilfe von Lemma 3.28 leicht abschätzen. Es gilt:

$$\begin{aligned}
\mathbb{E}[X_k] &= \sum_{t=0}^{\infty} \mathbb{E}[X_k \mid X_{k-1} = t] \cdot \Pr[X_{k-1} = t] \\
&\overset{\text{Lemma 3.28}}{\leq} \sum_{t=0}^{\infty} \left(1 + \frac{3}{r+1}\right) t \cdot \Pr[X_{k-1} = t] \\
&= \left(1 + \frac{3}{r+1}\right) \cdot \sum_{t=0}^{\infty} t \cdot \Pr[X_{k-1} = t] \\
&= \left(1 + \frac{3}{r+1}\right) \cdot \mathbb{E}[X_{k-1}].
\end{aligned}$$

Per Induktion folgt somit (da $X_0 = n$), dass $\mathbb{E}[X_k] \leq \left(1 + \frac{3}{r+1}\right)^k \cdot n$.

Laufzeitanalyse

wenn der Algorithmus nach k Runden noch nicht terminiert hat, so gibt es von mindestens einem der Punkte in Q_0 mindestens $2^{k/3}$ viele Kopien. Insbesondere gilt daher:

$$\begin{aligned}\mathbb{E}[X_k] &= \underbrace{\mathbb{E}[X_k \mid T \geq k] \cdot \Pr[T \geq k]}_{\geq 2^{k/3}} + \underbrace{\mathbb{E}[X_k \mid T < k] \cdot \Pr[T < k]}_{\geq 0} \\ &\geq 2^{k/3} \cdot \Pr[T \geq k].\end{aligned}$$

Vergleichen wir beide Ungleichungen, die wir für $\mathbb{E}[X_k]$ erhalten haben, so folgt

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

Laufzeitanalyse

Für $r = 11$ gilt $\Pr[T \geq k] \leq \min\left\{1, \left(\frac{1+3/12}{2^{1/3}}\right)^k n\right\} \leq \min\{1, 0{,}995^k n\}$ und somit für $k_0 = -\log_{0{,}995} n$:

$$\begin{aligned}
\mathbb{E}[T] &= \sum_{k \geq 1} \Pr[T \geq k] \\
&\leq \sum_{k=1}^{k_0} 1 + \sum_{k > k_0} 0.995^k n \\
&= \underbrace{\sum_{k=1}^{k_0} 1}_{=k_0} + \sum_{k'=1}^{\infty} 0.995^{k'} \cdot \underbrace{0.995^{k_0} n}_{=1} \\
&= k_0 + \mathcal{O}(1) \leq 200 \ln n + \mathcal{O}(1).
\end{aligned}$$

Das ist die Anzahl runden. Für die Laufzeit müssen wir noch mit n multiplizieren. □

The End

♠