AnW Exam Summary

Tom Offermann FS23

Lemmas/ Merksachen:

- Min Cut $=$ Max Flow
- $val(f) = netinflow(f) = netoutflow(f)$
- Flusserhaltung an einem Knoten v
- $\sum_{v \in V} deg(v) = 2|E|$
- $1 + x \leq e^x$
- $(1+x)^n \geq 1 + nx, x > -1, n \in \mathbb{Z}$
- $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$

Wahrscheinlichkeitstheorie

Wahrscheinlichkeitsraum:

- Definition der Elementarereignisse und deren Wahrscheinlichkeiten
- $\Omega = \{w_1, w_2, ..., w_n\}$
- $\sum_{i=0}^{n} Pr[w_i] = 1$
- Ereignis $E \subseteq \Omega$, $Pr[E] = \sum_{w \in E} Pr[w]$

Erwartungswert

- $\mathbb{E}[\alpha \cdot X + Y] = \alpha \cdot \mathbb{E}[X] + \mathbb{E}[Y]$
- $\mathbb{E}[X] = \sum_{x \in W_X} x \cdot Pr[X = x]$
- $\mathbb{E}[X] = \sum_{w \in \Omega} X(w) \cdot Pr[w]$
- $\mathbb{E}[X^k] = \sum_{x \in W_X} x^k \cdot Pr[X = x]$
- $\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X|A_i] \cdot Pr[A_i]$, A_i **disjunkt**
- $\mathbb{E}[X|A] = \frac{\mathbb{E}[1_A \cdot X]}{Pr[A]}$
- Für **unabhängige** X, Y:
  - $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$
- Für $W_X \subseteq \mathbb{N}_0$:
  - $\mathbb{E}[X] = \sum_{i=0}^{\infty} i \cdot Pr[X = i] = \sum_{i=0}^{\infty} \sum_{j=1}^{i} Pr[X = i] = \sum_{j=1}^{\infty} \sum_{i=j}^{\infty} Pr[X = i] = \sum_{j=1}^{\infty} Pr[X \geq j]$

Varianz

- $\mathbb{V}ar[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
- $\mathbb{V}ar[\alpha X + b] = \alpha^2 \mathbb{V}ar[X]$
- Für **unabhängige** X, Y:
  - $\mathbb{V}ar[X+Y] = \mathbb{V}ar[X] + \mathbb{V}ar[Y] = \mathbb{V}ar[X-Y]$

Zufallsvariablen

- Zufallsvariablen X, Y sind unabhängig wenn gilt:
  - $Pr[X = x, Y = y] = Pr[X = x] Pr[Y = y], \forall (x, y) \in W_X \times W_Y$
  - für mehr als 2 Variablen muss das selbe gelten: $\forall (x_1, x_2, ..., x_n) \in W_{X_1} \times W_{X_2} \times ... \times W_{X_N}$
- Zwei Indikatorvariablen I_A, I_B sind unabhängig, wenn:
  $Pr[I_A = 1, I_B = 1] = Pr[I_A = 1] Pr[I_B = 1]$

Negative Binomialverteilung

Wir warten auf den n-ten Erfolg innerhalb von z Zügen.

Der letzte Zug ist immer schon vor bestimmt, heißt wir können aus $z - 1$ Zügen $n - 1$-mal ziehen:
$f_X(x) = p\binom{z-1}{n-1} p^{n-1} (1-p)^{n-z+1} = \binom{z-1}{n-1} p^n (1-p)^{n-z+1}$

Kombinatorik

	ungeordnet	geordnet
mit zurücklegen	$\binom{n+k-1}{k}$	n^k
ohne zurücklegen	$\binom{n}{k}$	$\frac{n!}{(n-k)!}$

Graphentheorie

- Ein Weg ist eine Aufzählung von benachbarten Knoten, z.B.: $W = (v_0, v_1, v_2, v_0, v_3)$
- Ein Pfad ist ein Weg, in dem jeder Knoten höchstens einmal vorkommt, z.B.: $P = (v_0, v_1, v_3, v_2)$
- Ein Zyklus ist ein Weg der Länge k, in dem $v_0 = v_k$ gilt, z.B.: $Z = (v_0, v_1, v_2, v_1, v_0)$
- Ein Kreis ist ein Pfad der Länge $k \geq 3$, für den gilt, dass v_0 und v_{k-1} benachbart sind, z.B.: $K = (v_0, v_1, v_2), \{v_0, v_2\} \in E$
- G ist Baum $\iff$ G ist zusamm. UND G ist kreisfrei $\iff$ G ist zusamm. UND $|E| = |V| - 1$
- G ist k-zusammenhängend $\iff \forall X \subseteq V, |X| < k, G$ aus einer ZHK besteht
- G ist k-kanten-zusammenhängend $\iff \forall Y \subseteq E, |Y| < k, G$ aus einer ZHK besteht
- Wenn G k-zusammenhängend ist, ist er mindestens k-kanten-zusammenhängend
- Wenn G k-kanten-zusammenhängend ist, ist er maximal k-zusammenhängend
- **Satz von Menger:** Der Zusammenhang lässt sich auch mit der Anzahl von intern knoten-disjunkten, bzw. kantendisjunkten Pfaden definieren (Alle u-v-Pfade)
- Blockgraph definiert eine Relation auf Kantenpaare, so dass $e \sim f \iff \exists$ Kreis durch e und f ODER $e = f$
  - Blätter sind Blöcke

    - e Brücke $\iff$ Block besteht aus e
    - Artikulationsknoten sind auch Blöcke
- G ist eulersch (besitzt Eulertour) $\iff$ G ist zusamm. UND alle Knotengerade in G sind gerade
- **Hamiltonkreise**
    - Kreis auf dem alle Knoten liegen
    - NP-complete
    - In n-dim Hyperwürfel ⇒ Gray codes
    - DP ⇒ Exp. Speicher & Exp. Laufzeit
    - In/ Ex. (Siebformel) ⇒ Poly Speicher & Exp. Laufzeit
- **Satz von Dirac**
    - Knotengrad ≥ n/2 ⇒ Hamiltonkreis
    - Beweis Ideen:
        - Kreis der Länge $k < n$, kann immer zu Pfad der Länge $k + 1$ erweitert werden **(zsm!)**
        - Ein Pfad der Länge k, der an den Endpunkten nicht erweitert werden kann, kann immer zu einem Kreis der Länge k transformiert werden, da beide Endpunkte immer mindestens n/2 Kanten in den Pfad hinein haben, und somit ein benarchbartes Paar $(v_{i-1}, v_i), v_{i-1} \in N(v_k), v_i \in N(v_0)$ existieren muss
        - Dies gibt ein Verfahren für das finden eines Hamilonkreises ($O(n^2)$)
        - Die Schranke n/2 ist bestmöglich!, Bipartiter Graph mit Minimalgrad (n-1)/2 , n ungerade ist ein Gegenbeispiel
- **TSP**
    - NP Complete
    - Keine Approximationen, Existenz ⇒ P = NP
- **Metric-TSP**
    - **2**-Approx:
        - In $O(n^2)$ MST berechnen
        - Kanten verdoppeln und in $O(m)$ Eulertour berechnen
        - Eulertour ablaufen und schon gesehene Knoten überspringen
    - **3/2**-Approx:
        - In $O(n^2)$ MST berechnen
        - Knoten mit ungeradem Grad finden (gerade viele)
        - Minimales perfektes Matching zwischen diesen Knoten in $O(n^3)$ finden
        - Matching zum MST hinzufügen
        - Eulertour in $O(m)$ berechnen
        - Abkürzen

    - Das Matching ist höchstens 1/2 so groß wie der optimale minimale Hamiltonkreis

- **Matchings**
  - Kardinalitätsmaximal $\implies$ Inklusionsmaximal
  - Greedy Algo findet inklusions-maximales Matching (**2**-Approximation)
  - **Satz von Berge:** Matching M nicht kard. maximal $\iff$ $\exists$ augmentierender Pfad
  - $\not\exists$ augmentierender Pfad $\implies$ M kard. maximal
  - **Der Satz von Hall:**
    - Für einen bipartiten Graph $G = (A \cup B, E)$ gibt es genau dann ein Matching der Größe $|M| = |A|$, wenn gilt:
      $|N(X)| \geq |X|$, für alle nichtleeren Schnittmengen X von A
    - Beweis Ideen:
      - ($\implies$) Offensichtlich
      - ($\impliedby$) Induktion über a = |A|
        - BC: a = 1 ✓
        - Entweder gilt $|X| < N(X), \emptyset \neq X \subset A$ oder es gibt X_0 mit $|X_0| = N(X_0), \emptyset \neq X_0 \subset A$, in diesem Fall schauen wir uns die Teilgraphen $G' = (X_0, N(X_0))$ und $G''(A \backslash X_0, B \backslash N(X_0))$ seperat an.
  - Folgerung aus **Hall:**
    - **Frobenius:** Bipartite k-reguläre Graphen haben ein perfektes Matching, welches in $O(m)$ gefunden werden kann.
    - Für 2^k reguläre Graphen: Der Algo berechnet eine Eulertour in $O(m)$ pro ZHK, dann löscht er jede zweite Kante und wiederholt dies, bis der Graph $2^0 = 1$ regulär ist, also ein perfektes Matching besitzt, der Algo hat insgesamt eine Laufzeit von $O(m)$, da man in jeder Iteration die Kanten halbiert ⇒ $O(2m) = O(m)$

- **Färbungen**
  - chromatische Zahl $\chi(G)$ ist die minimale Anzahl an Farben, die notwendig sind den Graphen zu Färben
  - G mit chromatischer Zahl k wird k-partit genannt
  - G ist bipartit $\iff$ G hat keinen Kreis ungerader Länge
  - Jeder planare Graph kann mit 4 Farben gefärbt werden
  - Die Greedy Färbung berechnet immer eine Färbung mit k Farben, wobei $k = \Delta(G) + 1$
  - $\chi(G) \leq \Delta(G) + 1$ (Greedy)
  - Hat G (zsm.) einen Knoten mit kleinerem Grad, kann mit DFS oder BFS eine Reihenfolge festgelegt werden, für die insgesamt maximal $\Delta(G)$ Farben gebraucht werden
  - G ist ein k-regulärer Graph (zsm.) der einen Artikulationsknoten besitzt, die Partitionen die durch den Art. Knoten entstehen können je (mit Art. Knoten selber) nach oberem Verfahren mit k Farben gefärbt werden, dann können die Farben in jeder Partition so gecycelt werden, dass unser Art. Knoten in jeder Partition die selbe Farbe hat ⇒ k-Färbung des ganzen Graphen

- **Satz von Brooks:**
    - Sei G ein zsm. Graph, der nicht vollständig und kein ungerader Kreis ist, dann gilt: $\chi(G) \leq \Delta(G)$, und es existiert ein Algorithmus $\mathcal{A}(G)$ der G in $O(m)$ mit $\Delta(G)$ Farben färbt
- Reduktion des **LONGEST-PATH** Problem auf das Hamiltonkreisproblem:
    - Wir wählen einen Knoten v aus unserem Graphen G aus, entfernen ihn, fügen aber für jeden seiner Nachbarn einen neuen Knoten hinzu, welcher weiterhin mit diesem verbunden ist. Für den neuen Knoten gilt $deg(w) = 1$. Dieser neue Graph hat genau dann einen Pfad der Länge n wenn G einen Hamiltonkreis besitzt

Graphenalgorithmen

- **Low-Wert** DFS
    - Die low-Werte werden mittels DFS berechnet (umgekehrte Reihenfolge + DP)
    - low[v] := Die minimale DFS Nummer erreichbar von v über alle DFS-Baum-Kanten und eine Restkante
    - v ist Artikulationsknoten $\iff$ v ist root und hat mindestens 2 Nachbarn im DFS-Baum ODER v hat mindestens einen Nachbarn w mit dfs[v] ≤ low[w]
    - (v, w) ist Brücke $\iff$ dfs[v] < low[w]
- Schneller, langsamer Läufer Algo entscheidet ob ein Graph eulersch ist
- Algorithmus von **Hopcraft & Karp**
    - Findet ein kardinalitätsmaximales Matching in einem bipartiten Graphen
    - Layerstruktur mit BFS und findet kürzeste disjunkte augmentierende pfade von der ersten Layer aus
    - Diese können dann ALLE gleichzeitig (disjunkt) augmentiert werden
    - Das ganze wird so lange wiederholt bis es keine augmentierenden Pfade mehr gibt $O(\sqrt{V})$
- **Min-Cut**
    - Der minimale s-t-Schnitt lässt sich als Flussproblem in $O(nm \log n)$ berechnen. Wir müssen jedoch alle $n-1$ s-t-Schnitte (von s aus) durchprobieren um den minimalen Schnitt des gesamten Graphen zu finden. Das ergibt insgesamt $O(n^2 m \log n) = O(n^4 \log n)$
    - Wir bauen nun einen randomisierten Algorithmus, welcher eine zufällige Kante wählt und eine Kontraktion der beiden Knoten ausführt.
    - Dieser Algorithmus CUT(G) läuft in $O(n^2)$
    - Bei jeder Kantenkontraktion liegt der Algorithmus richtig zu $1 - 2/n$. Daraus folgt, das der gesamte Algorithmus mit Wahrscheinlichkeit $\frac{2}{n(n-1)}$ das gewollte Ergebnis liefert.
    - Für $\lambda\binom{n}{2}$ Wiederholungen liefert der Algorithmus das korrekte Ergebnis mit mindestens $1 - e^{-\lambda}$. Die Laufzeit ist $O(\lambda n^4)$
    - Bootstrapping:

      Wir brechen den Algorithmus nach t Iterationen ab und rechnen z.B. deterministisch mit Flüssen weiter, da die Wahrscheinlichkeit einen Fehler zu machen am Ende immer am größten ist. Diesen Algorithmus nennen wir $CUT_1(G)$. Für eine geeignetes t, z.B. $\sqrt{n}$ erreichen wir so einen

Algorithmus mit $O(\lambda n^3)$. Wir definieren nun einen neuen Algorithmus $CUT_2(G)$ dieser bricht wieder nach t Schritten ab, führt dann aber nicht mit der deterministischen Variante weiter, sondern nutzt unseren schnelleren $CUT_1(G)$ Algorithmus. Dieses in sich selber einsetzen wiederholen wir nun bis ins unendliche, wo wir dann im Grenzwert auf einen $O(n^2 poly(\log n))$ Algorithmus konvergieren.

- **LONGEST-PATH**
    - Bunte Pfade: Wir ermitteln in Polynomialzeit (in logn), (gegeben eine Färbung) die Existenz eines bunten Pfades in G (deterministisch)
    - Wir färben wir unseren Graphen zufällig und lassen den Bunte Pfade Algorithmus darauf laufen. Wenn dieser JA aus gibt, gibt es einen langen "kurzen" ($O(logn)$) Pfad, gibt er NEIN aus, obwohl es einen pfad gab, ist diese Ausgabe nur mit Wahrscheinlichkeit $1/e^k$ aufzufinden. Mittles **Monte-Carlo** Formel findet sich ein Ergebnis mit Erfolgswahrscheinlichkeit $1 - e^{-\lambda}$ in Poly Zeit. In $O(\lambda(2e)^k km)$
- **Max Flow** & **Min Cut**
    - **Ford-Fulkerson** findet einen augmentierenden Pfad im Restnetzwerk und augmentiert diesen jeweils um die minimale Restkapazität c auf diesem Pfad.
        - Wenn e in R_f UND $N_f \implies f(e) = f(e) + c$
        - Wenn e in R_f nicht $N_f \implies f(e) = f(e) - c$
    - **Ford-Fulkerson** (aus der Vorlesung) erlaubt keine entgegengesetzten Kanten, und läuft nur auf ganzzahligen oder rationalen Kapazitäten
    - Ganzzahlige Kapazitäten $\implies$ Ganzzahliger maximaler Fluss
    - Ganzzahliger maximaler Fluss $\overset{!nicht!}{\implies}$ Ganzzahlige Kapazitäten und $f(e)$ nicht unbedingt ganzzahlig $e \in E$

Primzahltest

- **Euklid:**
    - Wähle $a \in [n-1]$ zufällig
    - Wenn $ggT(a, n) = 1$ **return** *"Primzahl"*
    - Sonst **return** *"Keine Primzahl"*
    - *"Keine Primzahl"* ist immer richtig
    - *"Primzahl"* ist mit $\frac{|\mathbb{Z}_n^*|}{n-1}$ falsch
- $\varphi(n) \overset{def.}{=}$ Anzahl Teilerfremde Zahlen kleiner n
- Für n prim gilt: $\varphi(n) = n - 1$
- Es gilt für jedes Element in $a \in \mathbb{Z}_n^*,\ ord(a)\ \ teilt\ \ |\mathbb{Z}_n^*|$
- $a^{\varphi(n)} = 1 \mod n,\ a \in \mathbb{Z}_n^*$, für n prim: $a^{n-1} = 1 \mod n,\ a \in [n-1]$
- **Fermat:**
    - Wähle $a \in [n-1]$ zufällig
    - Wenn $ggT(a, n)$ ODER $a^{n-1} \neq 1 \mod n$ **return** *"Keine Primzahl"*

- Sonst **return** *"Primzahl"*
- *"Keine Primzahl"* ist immer richtig
- *"Primzahl"* ist falsch zu $P < 1/2$, außer n ist eine Carmichael-Zahl
- Wir machen einen Fehler, wenn n eine Pseudoprimzahlbasis ist: $ggT(a,n) = 1$ UND $a^{n-1} = 1 \mod n$

- **Carmichael-Zahlen:**
  - n ist Carmichael Zahl, wenn für alle $a \in \mathbb{Z}_n^*$, $ggT(a,n) = 1$ UND $a^{n-1} = 1 \mod n$ gilt
- **Miller-Rabin:**
  - Wenn n prim, ist $<\mathbb{Z}_n^*, +, \times>$ ein Körper, folglich hat $x^2 = 1$ zwei Lösungen: $x = \pm 1$
  - Für n prim gilt also: $a^{n-1} = 1 \mod n$ UND: $a^{\frac{n-1}{2}} \in \{1, n-1\} \mod n$
  - Zertifikat:
    - $n - 1 = 2^k d$
    - Wähle $a \in [n-1]$ zufällig
    - Wenn $a^d \neq 1 \mod n$ UND $\nexists i < k, a^{2^i d} = n - 1 \mod n$ **return** *"Keine Primzahl"*
    - Sonst **return** *"Primzahl"*
    - *"Keine Primzahl"* ist immer richtig
    - *"Primzahl"* ist falsch mit $P \leq 1/4$

Geometrische Algorithmen

- **Kleinster umschließender Kreis**
  - Es gibt immer eine Menge $Q, |Q| = 3$, so dass $C(Q) = C(P)$ gilt
  - Wenn wir alle Punktetripel durchprobieren ergibt dies einen trivialen $O(n^4)$ Algorithmus
  - Idee 1: Random Tripel ziehen $\implies O(n^4)$ erwartete Laufzeit, da wir genau eine der $\binom{n}{3}$ Tripel ziehen müssen.
  - Idee 2: Mehr als 3 Punkte ziehen $\implies O(n^4)$ erwartete Laufzeit, da wir immer noch genau eines der $\binom{n}{3}$ Tripel ziehen müssen, und das ziehen von z.B. 13 Punkten nur ein konstante Verbesserung mit sich bringt
  - Idee 3: Punkte außerhalb des berechneten Kreises verdoppeln
    - $out(p, P) = 1 \iff p \notin C(P)$
    - $ess(p, P) = 1 \iff C(P \backslash \{p\}) \neq C(P)$
    - Pro Runde ist die erwartete Anzahl an Punkten die verdoppelt werden gleich $3\frac{n-r}{r+1}$
    - In der k-ten Iteration gibt es höchstens $(1 + \frac{3}{r+1})^k n$ Punkte
    - In der k-ten Iteration gibt es mindestens $2^{k/3} Pr[T \geq k]$ Punkte
    - Umstellen nach $Pr[T \geq k]$ führt mit $r = 11$ zu einer gewünschten erwarteten Laufzeit von $O(n \log n)$
- **Jarvis-Wrap**

- Packt die Punktemenge ein
- Sucht immer den “rechtesten” Punkt
- Findet eine Randkanten in $O(n)$
- Braucht h Iterationen, bis alle Randkanten gefunden wurden
- $a \prec_b c \iff a$ rechts von bc

Lemma

Seien $p = (p_x, p_y)$, $q = (q_x, q_y)$, und $r = (r_x, r_y)$ Punkte in $\mathbb{R}^2$. Es gilt $q \neq r$ und p liegt links von qr genau dann wenn

$$\det(p,q,r) := \begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{vmatrix} = \begin{vmatrix} q_x - p_x & q_y - p_y \\ r_x - p_x & r_y - p_y \end{vmatrix} > 0$$

$$\Leftrightarrow \quad (q_x - p_x)(r_y - p_y) > (q_y - p_y)(r_x - p_x)$$

- **Local-Repair**
  - Sortiert die Punktemenge nach x-Koordinate
  - Berechnet den oberen und unteren Rand der konvexen Hülle
  - **Local-Repair** speichert immer den bisherig berechneten konvexe Rand
  - Wenn ein Punkt an den Rand “angeheftet” werden soll, wird rückwärts getestet, welcher Punkt der erste “passende” ist
  - Es gibt immer $2n - 2 - h = O(n)$ lokale Verbesserungen, h ist die Anzahl der Ecken des resultierenden konvexen Polygons

Extra Stuff

- Für Target-Shooting immer $Pr[|X - \mathbb{E}[X]| < \varepsilon \cdot \mathbb{E}[X]] \leq \delta$ zeigen
- **Monte Carlo** Algorithmus ist manchmal falsch
  - Einseitiger Fehler wird durch mehrfaches wiederholen “beseitigt”
  - Zweiseitiger Fehler kann nur “beseitigt” werden, wenn die Erfolgswahrscheinlichkeit bei mindestens $\frac{1}{2} + \varepsilon$ liegt
- **Las Vegas** Algorithmus ist nie falsch, läuft aber evtl. unendlich lange
  - Kann nach $t \cdot \mathbb{E}[T]$ Runden abgebrochen werden um eine Erfolgswahrscheinlichkeit von $1 - Pr[T \geq t \cdot \mathbb{E}[T]] \geq 1 - \frac{1}{t}$ zu garantieren

Algorithmen & Laufzeiten

Algorithmus	**Laufzeit**
Matrix Multiplikation	$O(n^3)$ oder besser $O(n^{2.81})$

DFS, BFS	$O(m+n)$ bzw. $O(m)$
Dijkstra	$O(n+m\log m)$
MST berechnen	$O(m\log m)$
Low-Wert, Brücken und Artik. Knoten	$O(m)$
Eulertour Schneller/langsamer Läufer	$O(m)$
Hamiltonkreis finden DP	$O(\exp(n))$
Hamiltonkreis finden Siebformel	$O(exp(n))$, poly Speicher
Hamiltonkreis	$O(\log n * n^{2.81} * 2^n)$ oder $O(n^2 * 2^n)$
Hamiltonkreis Dirac	$O(n^2)$
2-Approx Matching (Greedy)	$O(m)$
Hopcraft & Karp Kard. Max. Matching in bipartiten Graphen, (auch für alle Graphen, Blossom-Algo)	$O(\sqrt{n}(n+m))$
n gerade, mit $l : E \to \mathbb{N}_0$, findet minimales perfektes matching in K_n	$O(n^3)$
2-Approx Metric TSP	$O(m\log m)$
3/2-Approx Metric TSP	$O(n^3)$
Perfektes Matching in k-regulären bipartiten Graphen, insbesondere 2^k regulär	$O(m)$
Greedy Färbung	$O(m)$
Brooks: G zsm. nicht vollständig und kein ungerader Kreis: $\Delta(G)$-Färbung	$O(m)$
Dreifärbbarer Graph G kann mit $O(\sqrt{n})$ Farben färben	$O(m+n)$
Ford Fulkerson	$O(mnU)$
Capacity Scaling	$O(mn(1+\log U))$
Dynamic Trees	$O(mn\log n)$
Jarvis-Wrap	$O(hn)$, h = Ecken
LocalRepair	$O(n\log n)$
Kleinster Umschließender Kreis	$O(n\log n)$ (erwartet) Las-Vegas
Min Cut, Fluss	$O(n^4\log n)$
CUT(G), Ohne Bootstrapping	$O(\lambda n^4)$, für Erfolg mit mind. $1-e^{-\lambda}$
Min Cut, Bootstrapping	$O(n^2 poly(\log n))$
Bunte Pfade	$O(\lambda(2e)^k km)$, für Erfolg mit mind. $1-e^{-\lambda}$