Randomisierte Algorithmen

REDUKTION VON FEHLERWAHRSCHEINLICHKEITEN

Las-Vegas-Algorithmus: immer korrekt, Laufzeit Zufallsvariable, z.B. Quicksort

Monte-Carlo-Algorithmus: nicht immer korrekt, Laufzeit konstant, z.B. Testen einer Münze

Satz: Sei $\mathcal{A}$ ein randomisierter Algorithmus, der nie eine falsche Antwort gibt, aber zuweilen "???" ausgibt, wobei $Pr[\mathcal{A}(I) \text{ korrekt}] \geq \varepsilon$.
Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathcal{A}$ solange aufruft, bis entweder ein Wert verschieden von "???" ausgegeben wird oder bis $N = \varepsilon^{-1} \ln(\delta^{-1})$-mal "???" ausgegeben wurde, so gilt für $\mathcal{A}_\delta$, dass
$Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta$

Beweis: $Pr[\mathcal{A}_\delta(I) \text{ gibt bei } N \text{ Aufrufen immer ??? aus}] \leq (1-\varepsilon)^N$
Da $1 - x \leq e^{-x}$ $\leq e^{-\varepsilon N} = e^{\ln(\delta)} = \delta$

Satz (Monte-Carlo mit einseitigem Fehler):
Sei $\mathcal{A}$ ein randomisierter Algorithmus, der immer eine der beiden Antworten JA oder NEIN ausgibt, wobei
$Pr[\mathcal{A}(I) = \text{JA}] = 1$ falls I JA-Instanz
$Pr[\mathcal{A}(I) = \text{NEIN}] \geq \varepsilon$ falls I NEIN-Instanz
Dann gilt für alle δ, mit $\mathcal{A}_\delta$, der $\mathcal{A}$ solange aufruft bis NEIN oder $N = \varepsilon^{-1} \ln \delta^{-1}$-mal JA
$Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta$

Satz (Monte-Carlo mit zweiseitigem Fehler)
Sei $\mathcal{A}$ MC-Algo und $\varepsilon > 0$ mit
$Pr[\mathcal{A}(I) \text{ korrekt}] \geq \frac{1}{2} + \varepsilon$.
Dann gilt für alle $\delta > 0$, $\mathcal{A}_\delta$ macht $N = 4\varepsilon^{-2} \ln \delta^{-1}$ unabhängige Aufrufe und gibt Mehrheit der Aufrufe aus
$Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta$

Beweis: $\varepsilon \leq \frac{1}{2}$, $X :=$ Anzahl korrekter Antworten unter N Aufrufen
$P[\mathcal{A}_\delta(I) \text{ korrekt}] \geq Pr[X > \frac{N}{2}] = 1 - Pr[X \leq \frac{N}{2}]$
$E[X] = pN \geq \frac{N}{2} + \varepsilon N$, somit $\frac{N}{2} \leq (1-\varepsilon)(\frac{N}{2} + \varepsilon N) \leq (1-\varepsilon)E[X]$
$\Rightarrow$ Chernoff: $Pr[X \leq \frac{N}{2}] \leq Pr[X \leq (1-\varepsilon)E[X]] \leq e^{-\frac{1}{2}\varepsilon^2 E[X]} \leq \delta$
$\uparrow$ $E[X] \geq \frac{N}{2} = 2\varepsilon^{-2} \ln \delta^{-1}$

Satz (Optimierungsproblem)
Sei $\varepsilon > 0$ und $\mathcal{A}$ ein random. Algo für ein Maximierungsproblem, wobei gelte
$Pr[\mathcal{A}(I) \geq f(I)] \geq \varepsilon$
Dann gilt für alle $\delta > 0$: $\mathcal{A}_\delta$ mit $N = \varepsilon^{-1} \ln \delta^{-1}$ unabhängige Aufrufe und bester Antwort als Ausgabe
$Pr[\mathcal{A}_\delta \geq f(I)] \geq 1 - \delta$

[analog für Minimierungsprobleme]

QUICKSORT/QUICKSELECT

Quicksort ist Lasvegas-Algorithmus

$T_{\ell,r}$:= zufällige Anzahl an Vergleichen, die bei ~~einem~~ Quicksort(A,ℓ,r) ausgeführt werden

für $\ell \geq r$ ist $T_{\ell,r} = 0$

ansonsten:

$$E[T_{\ell,r}] = \sum_{i=\ell}^{r} Pr[t=i] \cdot (r-\ell + E[T_{\ell,i-1}] + E[T_{i+1,r}])$$

$$= \frac{1}{r+\ell+1} \cdot \sum_{i=0}^{r-\ell} (r-\ell + E[T_{\ell,\ell+i-1}] + E[T_{\ell+i+1,r}])$$

$$t_n = \begin{cases} 0 & \text{falls } n \leq 1 \\ \frac{1}{2}\sum_{i=0}^{n-1} (n-1 + t_i + t_{n-i-1}) & \text{falls } n \geq 2 \end{cases}$$

es gilt: $E[T_{\ell,r}] = t_{r-\ell+1}$

für alle $n \geq 3$ gilt:

$$n \cdot t_n = \sum_{i=0}^{n-1} (n-1+t_i+t_{n-i-1}) \quad (1) \quad \text{und} \quad (n-1)\cdot t_{n-1} = \sum_{i=0}^{n-2} (n-2+t_i+t_{n-i-2}) \quad (2)$$

$(2)-(1) \Rightarrow \quad n t_n - (n-1)t_{n-1} = 2(n-1) + 2t_{n-1}$

$$\Rightarrow \quad t_n = \frac{n+1}{n} \cdot t_{n-1} + \frac{2(n-1)}{n} \leq \frac{n+1}{n} \cdot t_{n-1} + 2$$

$$t_n \leq 2\sum_{i=3}^{n+1} \frac{n+1}{i} \qquad \text{(Beweis durch Induktion über } n\text{)}$$

$$\sum_{i=1}^{n} \frac{1}{i} = H_n = \ln(n) + O(1) \Rightarrow E[T_{1,n}] = t_n \leq 2(n+1)\ln(n) + O(n)$$

QUICKSELECT

$T = \sum_{i=1}^{N} (r_i - \ell_i)$, weil jeder Aufruf von PARTITION(A,ℓ_i,r_i,p) $r_i - \ell_i$ Vergleiche braucht

N_j := #Aufrufe von Quickselect für die $\left(\frac{3}{4}\right)^j n \leq r_i - \ell_i + 1 \leq \left(\frac{3}{4}\right)^{j-1} n$

$$T \leq \sum_{j=1}^{\infty} N_j \cdot \left(\frac{3}{4}\right)^{j-1} n \quad \text{und} \quad E[T] \leq n \cdot \sum_{j=1}^{\infty} \underbrace{E[N_j]}_{\leq 2 \ \forall j \geq 1} \cdot \left(\frac{3}{4}\right)^{j-1}$$

$$E[T] \leq 2n \sum_{j=1}^{\infty} \left(\frac{3}{4}\right)^{j-1} = 8n$$

Randomisierte Algorithmen

PRIMZAHLTEST

Euklid: $a \in \{2,\dots,\sqrt{n}\} \wedge ggT(a,n) > 1$

geringe Erfolgswahrscheinlichkeit, falls z.B. $n = \overset{\text{prim}}{p \cdot q}$, dann $Pr = \frac{1}{\sqrt{n}}$

bei genügender Fehlerwahrscheinlichkeitsreduktion $O(\sqrt{n})$ Wiederholungen nötig → nicht besser als Brute-Force

Kleiner fermatscher Satz:
Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$ $a^{n-1} \equiv_n 1$.

Carmichael-Zahlen: $a^{n-1} \equiv_n 1$ $\forall n$ mit $ggT(a,n) = 1$
Kleinste Carmichael-Zahl: $561 = 3 \cdot 11 \cdot 17$

$x^2 \equiv_n 1$ hat Lösungen $x = 1$ und $x = n-1$
$n - 1 = 2^k \cdot d$
Falls n prim, muss $a^{n-1} = (a^d)^{2^k} \equiv_n 1$ für alle $a \in \{1,\dots,n-1\}$
$\Rightarrow$ entweder $(a^d)^{2^{k-1}} \equiv_n 1$ oder $(a^d)^{2^{k-1}} \equiv_n n-1$

n nicht prim, falls nicht für alle $0 \le i \le k$ $(a^d)^{2^i} \equiv_n 1$
und nicht für ein $0 \le i \le k-1$ $(a^d)^{2^i} \equiv_n n-1$

MILLER-RABIN-PRIMZAHLTEST(n)
  if n=2 then return "Primzahl"
  else if n gerade oder n=1 then return "keine Primzahl"
  wähle $a \in \{2,\dots,n-1\}$ zufällig und berechne $k,d \in \mathbb{Z}$ mit $n-1 = d \cdot 2^k$ und d ungerade
  $x \leftarrow a^d \pmod n$
  if x=1 or x=n-1 then return "Primzahl"
  repeat k-1 mal
    $x \leftarrow x^2 \pmod n$
    if x=1 then return "keine Primzahl"
    if x=n-1 then return "Primzahl"
  return "keine Primzahl"

Laufzeit: $O(\ln(n))$

$Pr[\text{Algorithmus korrekt} \mid n \text{ nicht prim}] \ge \frac{3}{4}$

TARGET-SHOOTING

Problem: gegeben eine Menge U und eine Untermenge $S \subseteq U$ unbekannter Grösse, wie gross ist $\frac{|S|}{|U|}$?

TARGET-SHOOTING$_\varepsilon$()
wähle $u_1, \dots, u_N \in U$ zufällig, gleichverteilt und unabhängig
return $N^{-1} \sum_{i=1}^{N} I_S(u_i)$

Annahmen:
(1) I_S ist effizient berechenbar
(2) effiziente Prozedur, die uns ein uniform zufälliges Element aus U gibt

genauere Analyse: $Y_i := I_S(u_i)$, $i \in [n]$

$\Pr[Y_i = 1] = \frac{|S|}{|U|}$ $\qquad Y = \frac{1}{N} \sum_{i=1}^{N} Y_i = \frac{1}{N} \sum_{i=1}^{N} I_S(u_i)$

$E[Y] = \frac{|S|}{|U|}$ (unabhängig von N!) $\qquad \mathrm{Var}[Y] = \frac{1}{N} \left(\frac{|S|}{|U|} - \left(\frac{|S|}{|U|} \right)^2 \right)$

Sei $\varepsilon > 0$, wie gross muss N sein, damit der Algorithmus mit W'keit mind. $\frac{1}{2}$ eine Antwort im Intervall $[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}]$?

$\Rightarrow$ Satz: Seien $\delta, \varepsilon > 0$. Falls $N \geq 3 \cdot \frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln\left(\frac{2}{\delta}\right)$, so ist die Ausgabe des Algorithmus Target-Shooting mit W'keit mind. $\frac{1}{2}$.

Beweis: $Z := \sum_{i=1}^{N} Y_i = NY$

$\Pr[|Z - E[X]| \geq \varepsilon \cdot E[X]] \leq \delta$

Chernoff: $\Pr[|Z - E[X]| \geq \varepsilon \cdot E[X]] \leq 2 \cdot e^{-\varepsilon^2 E[Z] \cdot \frac{1}{3}} = 2 \cdot e^{-\varepsilon^2 N \cdot \frac{|S|}{3|U|}}$

mit $N = 3\frac{|U|}{|S|} \cdot \frac{1}{\varepsilon^2} \cdot \ln\left(\frac{2}{\delta}\right) \leq \delta$

FINDEN VON STABILEN MENGEN

gesucht: möglichst grosse Teilmenge $S \subseteq V$, sodass $G[S]$ keine Kanten enthält

Algorithmus für P_v:
Setze $S_v \leftarrow 1$ mit W'keit p, $S_v \leftarrow 0$ sonst;
for all $u \in V$ mit $\{u,v\} \in E$ do tausche Nachricht mit P_u aus:
if $S_u = S_v = 1$ then
setze S_u oder S_v auf 0
return S_v

$X_v :=$ Wert der für S_v in Zeile 1 gewählt wird
$X := \sum_{v \in V} X_v$
$Y_{e=\{u,v\}} := \begin{cases} 1 & \text{falls } X_u = X_v = 1 \\ 0 & \text{sonst} \end{cases}$
$Y := \sum_{e \in E} Y_e$

$S \geq X - Y \xrightarrow{\text{Linearität von } E} E[S] \geq E[X] - E[Y] = np - mp^2$

$E[X] = \sum_{v \in V} E[X_v] = n \cdot p$
$E[Y] = \sum_{e \in E} E[Y_e] = m \cdot p^2$

Erwartungswert maximieren: wenn $2mp = n \Leftrightarrow p = \frac{n}{2m}$ (erkennbar durch Ableitung
Annahme: G d-regulär: $2m = dn$ und $p = \frac{1}{d}$
$E[S] \geq \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d}$

Bunte Pfade

Satz: Falls wir das LONG-PATH-Problem für Graphen mit n Knoten in $t(n)$ Zeit entscheiden können, dann können wir in $t(2n-2)+O(n^2)$ Zeit entscheiden, ob ein Graph einen Hamiltonkreis hat.

Bunte Pfade mit DP finden

$\Pr[\text{Pfad der Länge } k \text{ bunt}] = \frac{k!}{k^k} \geq e^{-k}$ $\quad \left[e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \geq_{i=x} \frac{x^x}{x!}\right]$

#Wiederholungen bis Pfad bunt $\approx e^k = e^{\log(n)} = \text{poly}(n)$ mit $k = \log(n)$

$P_i(v) = \{S \in \binom{[k]}{i+1} \mid \exists$ in v endender genau mit S gefärbter ~~Pfad~~ bunter Pfad$\}$

$\exists$ bunter Pfad der Länge $k-1 \Leftrightarrow \bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$

Base Case: $P_0(v) = \{\{c(v)\}\}$ $\quad i-1 \to i$: $\forall v \in V$
$P_i(v) \leftarrow \emptyset$
$\forall w \in N(v)$
$\forall S \in P_{i-1}(w)$:
if $c(v) \notin S$: $P_i(v) \leftarrow P_i(v) \cup \{S \cup \{c(v)\}\}$

Laufzeit $i-1 \to i$: $\boxed{\sum_{v \in V} \deg(v)} \binom{k}{i} k$ ($2m$)

$$\sum_{i=1}^{k} m \binom{k}{i} k = m \cdot k \cdot \sum_{i=1}^{k} \binom{k}{i} \leq mk \cdot 2^k \overset{k=\log_2 n}{=} mn \log_2(n)$$

Monte-Carlo-Algo mit Pfaden der Länge B

1. Zufälliges Färben von G mit B+1 Farben $O(n)$
2. Suche bunten Pfad, wenn gefunden, gib „es gibt einen" ansonsten „es gibt keinen" aus → nur einseitiger Fehler möglich

$\Pr[\text{Erfolg}] = \frac{k!}{k^k} \geq e^{-k}$ $\qquad$ Erfolg $\sim$ Geom(e^{-k})

$\Pr[\text{immer Fehler bei } \lambda e^k \text{ Wiederholungen}] \leq (1-e^{-k})^{\lambda e^k}$

$$\leq (e^{e^{-k}})^{\lambda e^k} = e^{-e^k \cdot e^k \cdot \lambda} = e^{-\lambda}$$

gesamte Runtime: $O(\lambda (2e)^k mk)$ mit $k = \log_2(n)$ $\quad O(\lambda n^2 \log(n) m)$

Convex Hull

Liniensegment: $v_0, v_1 \in \mathbb{R}^m$, $\overline{v_0 v_1} := \{(1-\lambda)v_0 + \lambda v_1 \mid \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1\}$

Eine Menge ~~heisst~~ $C \subseteq \mathbb{R}^m$ heisst **konvex**, falls für alle $v_0, v_1 \in C$ das ganze Liniensegment $\overline{v_0 v_1}$ in C enthalten ist.

konvexe Hülle: $\text{conv}(S) := \bigcap_{\substack{S \subseteq C \subseteq \mathbb{R}^d, \\ C \text{ konvex}}} C$

ConvexHull-Problem:

Annahme: allgemeine Lage, d.h. keine 3 Punkte auf gemeinsamer Gerade,
keine 2 Punkte selbe x-Koordinate

finde konvexe Hülle von P

Lemma: $(q_0, q_1, \dots, q_{h-1})$ ist die Eckenfolge des von $\text{conv}(P)$ umschliessenden Polygons gegen den Uhrzeigersinn genau dann, wenn alle Paare (q_{i-1}, q_i), $i = 1, 2, \dots, h$, Randkanten von P sind.

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

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

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

naiver Algorithmus: gehe durch jedes der $n(n-1)$ qr-Paare und prüfe auf Randkante, indem man für alle $n-2$ Punkte testet, ob sie links von qr liegen
$\rightarrow O(n^3)$

Lemma: Ist q eine Ecke der konvexen Hülle von P, so ist die Relation $\prec_q$ eine totale Ordnung auf $P \setminus \{q\}$. Für das Minimum p_{min} dieser Ordnung gilt, dass qp_{min} eine Randkante ist ($p_1 \prec_q p_2 \overset{\text{def}}{\Leftrightarrow}$ p_1 rechts von p_2).

```
JARVISWRAP(P)
  h ← 0
  p_now ← Punkt ∈ P mit kleinster x-Koor.
  repeat
    q_h ← p_now
    p_now ← FINDNEXT(q_h)
    h ← h+1
  until p_now = q_0
  return (q_0, ..., q_{h-1})
```

```
FINDNEXT(q)
  wähle p_0 ∈ P\{q} beliebig
  q_next ← p_0
  for all p ∈ P\{q, p_0} do
    if p rechts von q q_next then q_next ← p
  return q_next
```

Laufzeit: $O(nh)$

LOCAL REPAIR

planar := kann eingebettet werden

eben := ist schon eingebettet

Graph $G=(P,E)$ eben $\Leftrightarrow$ Segmente $pq := \text{conv}(\{p,q\})$ der Kanten $\{p,q\} \in E$ schneiden sich höchstens in ihren Endpunkten

werden diese Segmente entfernt, enthält man Gebiete von G,
falls beschränkt → innere Gebiete, ansonsten äussere Gebiete

$T=(P,E)$ ist Triangulierung von P $\Leftrightarrow$ T ist eben und mit dieser Eigenschaft maximal
→ inneren Gebiete sind immer Dreiecke

Lemma: Der lokale Verbesserungsprozess macht genau $2n-2-h$ Verbesserungsschritte und erzeugt eine Triangulierung mit $2n-2-h$ inneren Dreiecken und $3n-3-h$ Kanten.

Beweis: zu Beginn $2n-2$ gerichtete Kanten, 0 Dreiecke
1 Verbesserungsschritt $\hat{=}$ Umrichtung zweier gerichteter Kanten und Einfügen einer neuen gerichteten Kante (+1 Dreieck)

am Ende: h gerichtete Kanten
$\Rightarrow$ #Kanten $= n-1+$ #Schritte $= 3n-3+h$ (≥ 3)

Lemma (Euler Relation): Sei $G=(P,E)$ ein ebener Graph auf P mit $v:=|P|$ Knoten, $e:=|E|$ Kanten, f Gebieten (inkl. äusseren Gebiets), und c ZHKs. Dann gilt:
$v-e+f=1+c$

Korollar: P menge mit $n \geq 3$ Punkten in allgemeiner Lage, h Anzahl Kanten von $\text{conv}(P)$
(i) Jede Triangulierung hat genau $3n-3-h$ Kanten und $2n-2-h$ innere Gebiete
(ii) Jeder ebene Graph auf P hat höchstens $3n-3-h \leq 3n-6$ Kanten und höchstens $2n-2-h \leq 2n-5$ innere Gebiete

```
LOCALREPAIR(p_1,...,p_n)        (setzt Sortierung nach x-Koordinate voraus)
  q_0 ← p_1
  h ← 0
  for i ← 2 to n do                          // untere konvexe Hülle (links → rechts)
    while h>0 und q_h links von q_{h-1}p_i do
      h ← h-1
    h ← h+1
    q_h ← p_i
  h' ← h
  for i ← n-1 downto 1 do                    // obere konvexe Hülle (rechts → links)
    while h>h' und q_h links von q_{h-1}p_i do
      h ← h-1
    h ← h+1
    q_h ← p_i
  return (q_0,...,q_{h-1})
```

Laufzeit: $O(n)$

konvexe Hülle kann nicht schneller als Sortieren gelöst werden
→ kann man konvexe Hülle in $t(n)$ berechnen, so kann man in Zeit $t(n)+O(n)$ sortieren

Geometrische Algorithmen

KLEINSTER UMSCHLIESSENDER KREIS

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

Beweis: Annahme: es gibt zwei verschiedene kleinste umschliessende Kreise $C_1, C_2 \rightarrow C_1 \neq C_2$ $P \subseteq C_1 \cap C_2$

→ Existenz eines kleineren umschliessenden Kreises mit $z = \frac{1}{2}(z_1 + z_2)$ und $\hat{r} = \sqrt{r^2 - (\frac{1}{2}|z_1 z_2|)^2}$

Eigenschaften

(1) Der Rand von C(P) enthält mind. zwei Punkte
→ ansonsten Verschieben kleinern bis zweiter Punkt draufliegt

(2) Wenn der Rand von C(P) nur zwei Punkte p und q enthält, $C(P) = C(\{p,q\})$

(3) Enthält der Rand von C(P) mind. 3 Punkte p,q,r, dann $C(P) = C(\{p,q,r\})$

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

naiver Algorithmus: $O(n^4)$

CLEVER

```
RANDOMISED_PRIMITIVE_VERSION(P)
  repeat forever
    wähle Q⊆P mit |Q|=3 (11) zufällig und gleichverteilt
    bestimme C(Q)
    if P⊆ C(Q) then return C(Q)
    else verdopple alle Punkte außerhalb von C(Q)
```

Problem: WC: geom. verteilt mit $\frac{1}{\binom{n}{3}} \Rightarrow O(n^4)$
→ Idee: wir ziehen mehr Punkte (konstante Anzahl) → O(1)
→ Verdoppeln aller Punkte, die ausserhalb von C(Q) liegen

Lemma: Seien $n_1, \ldots, n_t$ natürliche Zahlen und $N := \sum_{i=1}^{t} n_i$.
Wir erzeugen $X \in \{1, \ldots, t\}$ zufällig wie folgt:

```
k ← UNIFORMINT(1,N)
x ← 1
while Σ_{i=1}^{x} n_i < k do
    x ← x+1
return x
```

Dann gilt $Pr[X=i] = \frac{n_i}{N}$ für alle $i = 1, \ldots, t$.

Laufzeitanalyse:

$X_k :=$ Anzahl der Punkte nach k Iterationen
$T :=$ Anzahl Iterationen

$Q_0 \subseteq P$ mit $|Q_0| = 3$ und $CC(Q_0) = CC(P)$

untere Schranke: $E[X_k] = E[X_k | T > k] \cdot Pr[T > k] + \underbrace{E[X_k | T \leq k]}_{=0} \cdot Pr[T \leq k]$

$$\geq 2^{\frac{k}{3}} \cdot Pr[T > k]$$

obere Schranke:

Lemma: Für P und $|P| = n$ und $R \subseteq P$ zufällig mit $|R| = r$, dann gilt
$E[\underbrace{\#\text{Punkte in } P \setminus C^*(R)}_{X}] \leq 3 \cdot \frac{n}{r+1}$

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

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

$out(p,R) = 1 \iff essential(p, R \cup \{p\}) = 1$

$$E[X] = \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{s \in P \setminus R} out(s,R)$$

$$= \frac{1}{\binom{n}{r}} \sum_{R \in \binom{P}{r}} \sum_{s \in P \setminus R} essential(s, R \cup \{s\})$$

$$= \frac{1}{\binom{n}{r}} \sum_{Q \in \binom{P}{r+1}} \underbrace{\sum_{p \in Q} essential(p,Q)}_{\leq 3}$$

(warum pr)

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

Korollar: $E[X_k] \leq (1 + 3 \cdot \frac{1}{r+1})^k \cdot n$

Beweis: $E[X_k] = \sum_{\hat{n}=1}^{\infty} E[X_k | X_{k-1} = \hat{n}] Pr[X_{k-1} = \hat{n}] \leq (1 + \frac{3}{r+1}) \underbrace{\sum_{\hat{n}=1}^{\infty} \hat{n} \cdot Pr[X_{k-1} = \hat{n}]}_{E[X_{k-1}]}$

$$\leq (1 + \frac{3}{r+1})^k \cdot E[X_0] = (1 + \frac{3}{r+1})^k \cdot n$$

$$2^{\frac{k}{3}} \cdot Pr[T > k] \leq E[X_k] \leq (1 + \frac{3}{r+1})^k \cdot n$$

$E[T] = \sum_{k=1}^{\infty} Pr[T > k]$

$$\leq \sum_{k=0}^{k_0} 1 + n \cdot \sum_{k > k_0}^{\infty} 0{,}995^k \leq$$

$$= \underbrace{\sum_{k=0}^{k_0} 1}_{O(\log n)} + \underbrace{\sum_{k > k_0}^{\infty} 0{,}995^{k-k_0}}_{O(1)} \cdot \underbrace{0{,}995^{k_0} n}_{\leq 1}$$

$Pr[T > k] \geq \left(\frac{1 + \frac{3}{n^2}}{2^{\frac{1}{3}}} \right)^k \cdot n \quad (r = n-1)$

Definition k_0: $\left(\frac{1+\frac{1}{n}}{2^{\frac{1}{3}}}\right)^{k_0} = \frac{1}{n} \Rightarrow k_0 \leq \log_{0{,}995}(n)$

MinCut

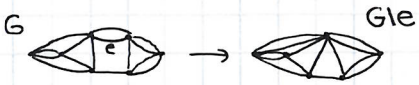
Problem: gegeben einen Multigraphen G, bestimme die Kardinalität des minimalen Kantenschnitts $[\mu(G)]$

Kantenschnitt in G: Menge von Kanten C, sodass $(V, E \setminus C)$ nicht zusammenhängend ist

$\mu(G) \leq$ minimaler Grad $\deg(v)$ für alle $v \in V$

mit Flows: minimale Schnitte in $O(nm \log n) = O(n^3 \log n)$ berechnen
→ $O(n^4 \log n)$ für MINCUT ($n-1$ s-t-Schnitte)

KANTENKONTRAKTION



für $k :=$ #kanten zwischen u und v
und $e = \{u,v\} \in E$ gilt
$\deg_{G \setminus e}(x_{u,v}) = \deg_G(u) + \deg_G(v) - 2k$
$|E(G/e)| = |E(G)| - k$

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

bei einem einzelnen Punkt wird die Kardinalität des minimalen Schnittes als unendlich definiert

```
CUT(G)
  G' ← G
  while |V(G')| > 2 do
    e ← gleichverteilt zufällige Kante in G
    G' ← G/e
  return Grösse des Schnitts in G
```

- kantenkontraktion in $O(n)$
- kantenauswahl in $O(n)$

→ $O(n^2)$ für Cut(G)

Lemma: Falls e ~~oder~~ $\in E$ gleichverteilt zufällig unter den Kanten in G gewählt wird, dann gilt $\Pr[\mu(G) = \mu(G/e)] \geq 1 - \frac{2}{n}$.

Beweis: $k := |C| = |\mu(G)|$, $\deg_G(v) \geq k \Rightarrow |E| = \frac{1}{2} \sum_{v \in V} \deg(v) \geq \frac{kn}{2}$
wegen „$e \notin C \Rightarrow \mu(G \setminus e) = \mu(G)$" gilt:
$\Pr[\mu(G) = \mu(G/e)] \geq \Pr[e \notin C] = 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{\frac{kn}{2}} = 1 - \frac{2}{n}$

$\hat{p}(G) :=$ w'keit, dass CUT(G) den Wert $\mu(G)$ ausgibt $\quad \hat{p}(n) := \inf_{\substack{G=(V,E) \\ |V|=n}} \hat{p}(G)$

Lemma: für $n \geq 3$ gilt $\hat{p}(n) \geq (1 - \frac{2}{n}) \cdot \hat{p}(n-1)$

Beweis: $E_1 :=$ Ereignis $\mu(G) = \mu(G/e)$
$E_2 :=$ Ereignis, dass CUT(G/e) den Wert $\mu(G/e)$ ausgibt

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

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

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

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

(1) Laufzeit: $O(\lambda n^4)$

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

Beweis (2): $(1-\hat{p}(n))^{\lambda\binom{n}{2}} \leq (1-\frac{1}{\binom{n}{2}})^{\lambda\binom{n}{2}} \leq \left(e^{-\frac{1}{\binom{n}{2}}}\right)^{\lambda\binom{n}{2}} = e^{-\lambda}$

→ mit $\lambda = \ln(n)$ Laufzeit von $O(n^4 \log n)$ [nicht besser als deterministischer Fluss-Algorithmus]

BOOTSTRAPPING

Idee: Wir brechen bei G' mit t Knoten ab und verwenden den randomisierten $O(t^4)$ Algorithmus mit Erfolgsw'keit $\geq 1-e^{-1}$

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

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

$$\underbrace{\lambda \frac{n(n-1)e}{t(t-1)(e-1)}}_{\#\text{Wdh}} \cdot O(n(n-t)+t^4) = O\left(\lambda\left(\frac{n^4}{t^2} + n^2 t^2\right)\right) \underset{t=\sqrt{n}}{=} O(\lambda n^3)$$

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

Flows

gerichteter Graph ohne Schleifen

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

Fluss $f :=$ Funktion $f: A \to \mathbb{R}$ mit folgenden Bedingungen

- Zulässigkeit: $0 \leq f(e) \leq c(e) \quad \forall e \in A$
- Flusserhaltung: $\forall v \in V \setminus \{s,t\}$ gilt $\sum_{\substack{u \in V \\ (u,v) \in A}} f(u,v) = \sum_{\substack{u \in V \\ (v,u) \in A}} f(v,u)$
- Wert des Flusses $val(f) := netoutflow(s)$
$:= \sum_{\substack{v \in V \\ (s,v) \in A}} f(s,v) - \sum_{\substack{u \in V \\ (u,s) \in A}} f(u,s)$

f ganzzahlig $\Leftrightarrow f(e) \in \mathbb{Z} \quad \forall e \in A$

Lemma: Nettozufluss der Senke = Wert des Flusses, also

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

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

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

$$= \underbrace{\left(\sum_{\substack{u \in V \\ (s,u) \in A}} f(s,u) - \sum_{\substack{u \in V: \\ (u,s) \in A}} f(u,s) \right)}_{val(f)} + \underbrace{\left(\sum_{\substack{u \in V \\ (t,u) \in A}} f(t,u) - \sum_{\substack{u \in V \\ (u,t) \in A}} f(u,t) \right)}_{-netinflow(t)}$$

s-t-Schnitt: Partition (S,T) von V mit $s \in S$ und $t \in T$ und Kapazität

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

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

Beweis: ~~$F(v,w) = \sum$~~ $F(U,W) = \sum_{(u,v) \in (U \times W) \cap A} f(u,v)$

$$val(f) = F(S,T) - F(T,S)$$

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

$$= \sum_{v \in S} \underbrace{\left(\sum_{(v,u) \in A} f(v,u) - \sum_{(u,v) \in A} f(u,v) \right)}_{=0 \text{ für } v \neq s}$$

→ Case $x \in S, y \in S$: kommt zweimal in Summe vor
Case $x \in S, y \notin S$: "$v = x$" $+ f(x,y)$
Case $x \notin, y \in S$: "$v = y$" $- f(x,y)$
Case $x \notin S, y \notin S$: kommt nicht in Summe vor

$$val(f) = F(S,T) - F(T,S)$$
$$\leq F(S,T) \qquad \text{[Nicht-negativität des Flusses]}$$
$$\leq cap(S,T) \qquad [f(u,w) \leq c(u,w)]$$

Maxflow-Mincut-Theorem

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

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

Definition des Restnetzwerks

$N_f = (V, A_f, r_f, s, t)$

(1) Ist $e \in A$ mit $f(e) < c(e)$, dann ist $e \in A_f$ mit $r_f(e) = c(e) - f(e)$

(2) Ist $e \in A$ mit $f(e) > 0$, dann $e_{opp} \in A$ mit $r_f(e_{opp}) = f(e)$

Satz: Fluss maximal $\Leftrightarrow$ keinen gerichteten s-t-Pfad in N_f

Beweis: "$\Rightarrow$" Annahme: ~~Fluss nicht maximal~~, es gibt gerichteten s-t-Pfad, f kann augmentiert werden $\Rightarrow$ f nicht maximal

"$\Leftarrow$" es gibt keinen gerichteten s-t-Pfad
$\Rightarrow$ es gibt s-t-Schnitt (S,T) mit $cap(S,T) = val(f) \Rightarrow$ f maximal

```
FORD-FULKERSON(V, A, sc, s, t)
  f ← 0
  while ∃ s-t-Pfad P in N_f do
    augmentiere den Fluss entlang P
  return f
```

Satz: Sind alle Kapazitäten ganzzahlig und höchstens U, so gibt es einen ganzzahligen maximalen Fluss, der in $O(mnU)$ berechnet werden kann.

Capacity-Scaling: $O(mn(1+\log U))$
Dynamic Trees: $O(mn \log n)$

Anwendung:

- matching in bipartiten Graphen → kardinalitätsmaximales $m = val(f)$
- kanten- und knotendisjunkte Pfade:
  Konstruktion: 1 ungerichtete Kante → 2 gerichtete Kanten
  $val(f) = outdeg_f(u) - indeg_f(u) = indeg_f(v) - outdeg_f(v)$
  wiederhole $val(f)$-mal: laufe beginnend bei u alle ungebrauchten Kanten mit Fluss 1 entlang und markiere als gebraucht
  → $val(f)$ kantendisjunkte Pfade

Bildsegmentierung

$\beta: P \to \mathbb{R}$ $\quad \beta_p$ Hintergrund
$\alpha: P \to \mathbb{R}$ $\quad \alpha_p$ größer $\Rightarrow$ eher im Vordergrund
$\gamma: E \to \mathbb{R}$ $\quad \gamma_e$ größer $\Rightarrow$ eher im gleichen Teil

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

Problem: finde Partition (A,B), die $q(A,B)$ maximiert
$\Leftrightarrow$ minimierung von $q'(A,B) := \sum_{p \in A} \beta_p + \sum_{p \in B} \alpha_p + \sum_{\substack{e \in E \\ |e \cap A| = 1}} \gamma_e = cap(S,T)$

Netzwerkkonstruktion: