Randomisierte Algorithmen

Markus Lohrey (Univ. Stuttgart / Univ. Halle)

Zimmer 211

lohrey@informatik.uni-stuttgart.de

Vorlesung:

Di 14.15 - 15.45 } Raum 1.29
Di 16.15 - 17.00 }

Übungen:

Di 17.00 - 17.45 Raum 1.29

Literatur:

Randomized Algorithms
R. Motwani, P. Raghavan
Cambridge University Press

Kapitel 1. Einführung

Randomisierter Algorithmus:
klassischer deterministischer Algorithmus erweitert um Zufallskomponente (Münzwurf)

Beispiel 1: Quicksort (QS)

gegeben: Menge S von n Zahlen
gesucht: S in sortierter (aufsteigender) Reihenfolge

klassischer QS:

(1) Wähle ein Element $y \in S$
(2) $S_1 := \{x \in S \mid x < y\}$
$S_2 := \{x \in S \mid y < x\}$
(3) Sortiere rekursiv $S_1 \rightsquigarrow$ Liste L_1
Sortiere rekursiv $S_2 \rightsquigarrow$ Liste L_2
(4) Output L_1, y, L_2

Problem: Wie wählt man y in Schritt (1)
optimale Wahl von y falls $|S_1|, |S_2| < \frac{n}{2}$
(Median von S)

Der Median von S kann in Zeit $O(n)$, d.h. Zeit $c \cdot n$ für Konstante c, gefunden werden (relativ kompliziert).

$\rightarrow T(n) \leq 2 \cdot T(\frac{n}{2}) + (c+1) \cdot n$

$\rightarrow T(n) \in O(n \cdot \log(n))$

Angenommen wir würden in Schritt (1) in Zeit $O(n)$ ein $y \in S$ mit $|S_1|, |S_2| \leq \frac{3}{4}n$ finden (davon gibt es $\frac{n}{2}$ viele).

$\rightarrow T(n) \leq 2 \cdot T(\frac{3}{4} \cdot n) + O(n)$

$\rightarrow T(n) \in O(n \cdot \log(n))$

randomisierte Wahl von y

↳ Algorithmus rand QS

(1) Wähle ein $y \in S$ zufällig und gleichverteilt aus S.

(2) $S_1 := \{x \in S \mid x < y\}$

$S_2 := \{x \in S \mid y < x\}$

(3) Sortiere rekursiv $S_1 \rightarrow$ Liste L_1

Sortiere rekursiv $S_2 \rightarrow$ Liste L_2

(4) Output $L_1 \, y \, L_2$

Analyse von rand QS

Für $1 \leq i \leq n$ sei

$S_{(i)}$ = "das Element von Rang i in S"

= "das eindeutige $s \in S$ mit $|\{y \in S \mid y \leq s\}| = i$"

Definiere Zufallsvariable X_{ij} durch:

$$X_{ij} = \begin{cases} 1 & \text{falls } S_{(i)} \text{ und } S_{(j)} \text{ in einem Ablauf von rand QS verglichen werden} \\ 0 & \text{sonst} \end{cases}$$

Wir wollen den Erwartungswert $E[V]$ für die Anzahl V der Vergleiche in einem Ablauf von rand QS berechnen.

$$E[V] = E\left[\sum_{i=1}^{n} \sum_{j>i} X_{ij}\right]$$

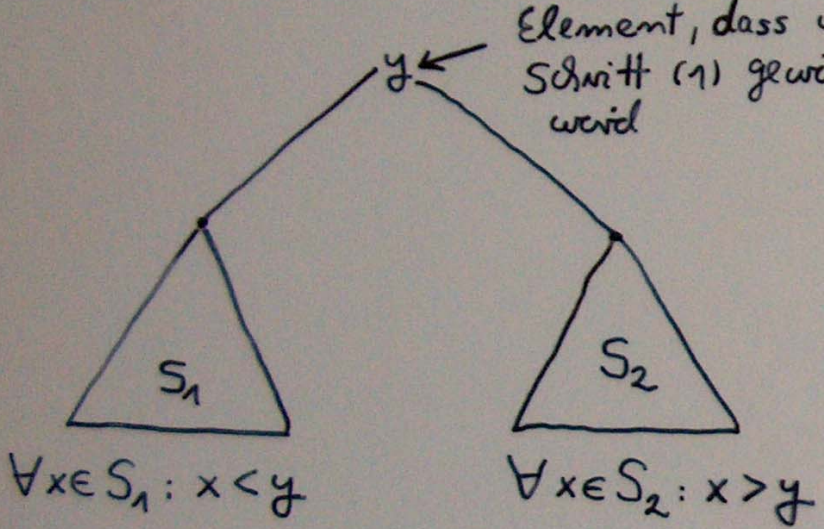
$$= \sum_{i=1}^{n} \sum_{j>i} E[X_{ij}]$$

Linearity of Expectations

Sei p_{ij} = Prob[$S_{(i)}$ und $S_{(j)}$ werden verglichen]

$\rightarrow E[X_{ij}] = 1 \cdot p_{ij} + 0 \cdot (1 - p_{ij}) = p_{ij}$

Ein Ablauf von randQS kann durch einen binären Suchbaum T beschrieben werden.



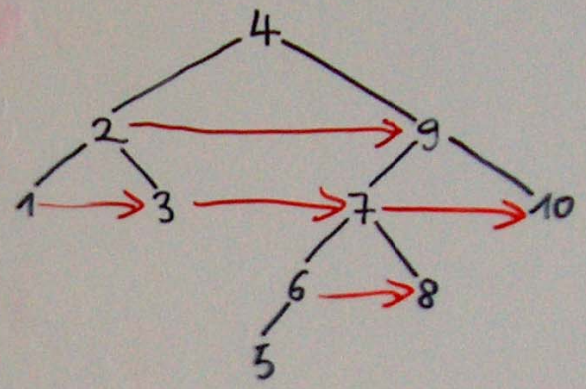
Beobachtung:

$a \in S$ wird mit $b \in S$ verglichen.

$\Leftrightarrow$

a ist Vorgänger von b in T oder
b ist Vorgänger von a in T.

Beispiel: $S = \{1, \ldots, 10\}$



Level-Order-Transversal $\pi(T)$ von T:

4, 2, 9, 1, 3, 7, 10, 6, 8, 5

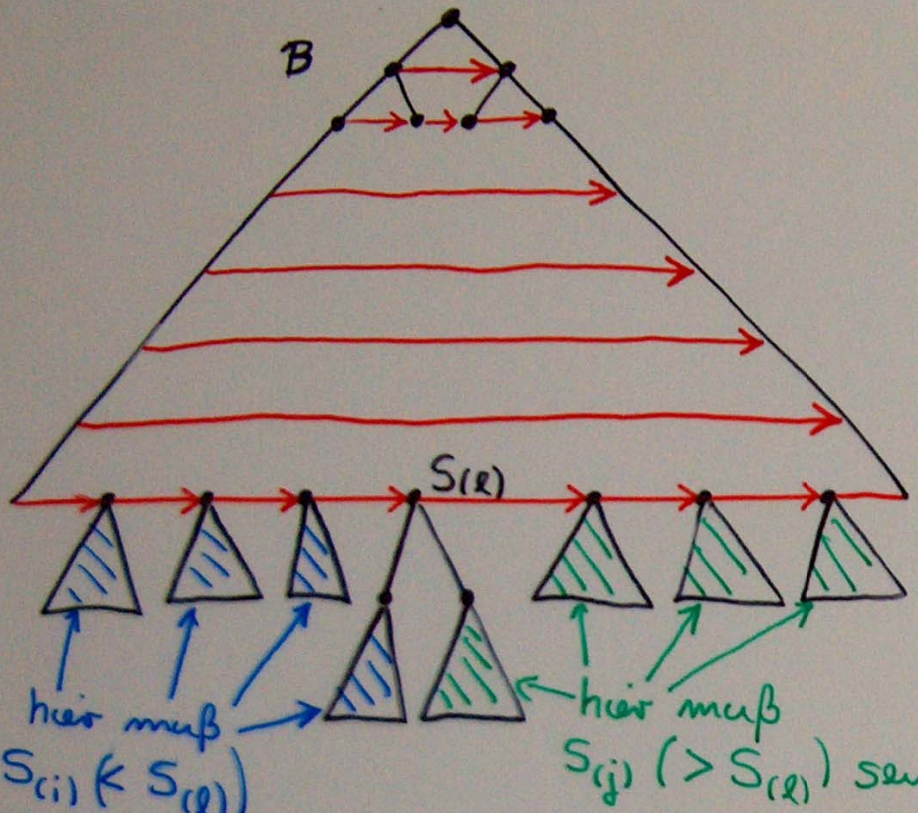
Lemma: $S_{(i)}$ wird mit $S_{(j)}$ verglichen $(i<j)$

$\Leftrightarrow$

Es gibt <u>kein</u> $S_{(l)}$ mit $i<l<j$, welches in $\pi(T)$ vor $S_{(i)}$ und $S_{(j)}$ steht.

Beweis:

$\Rightarrow$: Angenommen $S_{(l)}$ $(i<l<j)$ steht in $\pi(T)$ vor $S_{(i)}$ und $S_{(j)}$

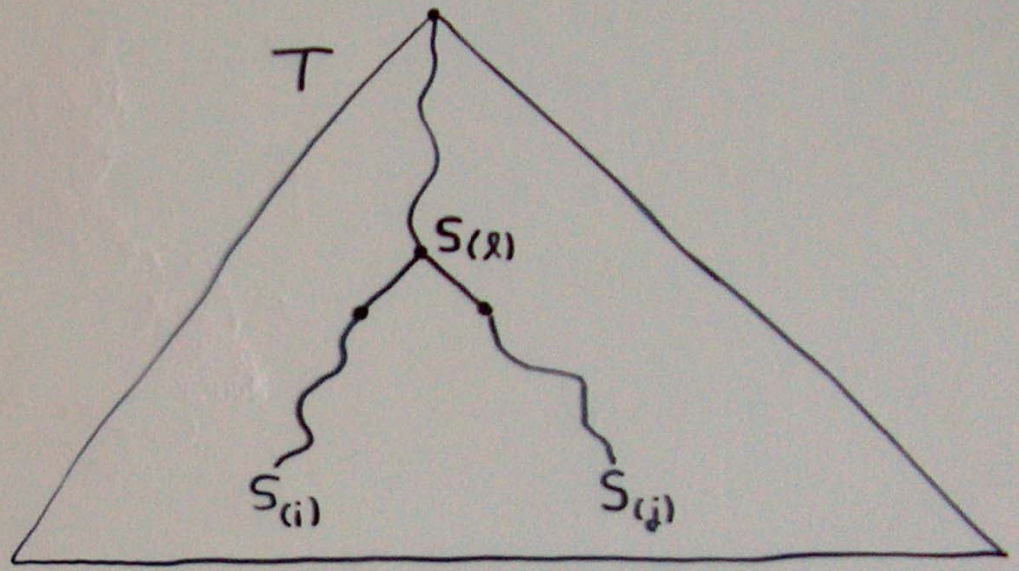


Weder $S_{(i)}$ kann Vorgänger von $S_{(j)}$ sein, noch $S_{(j)}$ kann Vorgänger von $S_{(i)}$ sein

$\Rightarrow$ $S_{(i)}$ und $S_{(j)}$ werden nicht verglichen.

$\Leftarrow$: Angenommen $S_{(i)}$ wird nicht mit $S_{(j)}$ verglichen.

$\rightarrow$ $S_{(i)}$ kein Vorgänger von $S_{(j)}$ und $S_{(j)}$ kein Vorgänger von $S_{(i)}$



$S_{(i)} < S_{(\ell)} < S_{(j)}$ und

$S_{(\ell)}$ vor $S_{(i)}$ und $S_{(j)}$ in $\pi(T)$ $\square$

Beobachtung: Jedes Element aus dem Intervall $I = \{S_{(i)}, S_{(i+1)}, \dots, S_{(j)}\} \subseteq S$ ist mit gleicher Wahrscheinlichkeit das erste Element aus I in der Liste $\pi(T)$.

Diese Wahrscheinlichkeit ist somit

$$\frac{1}{|I|} = \frac{1}{j-i+1}$$

$$\rightarrow p_{ij} = \frac{2}{j-i+1}$$

$$\rightarrow E[V] = \sum_{i=1}^{n} \sum_{j>i} p_{ij} = \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j-i+1}$$

$$= 2 \cdot \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{1}{k}$$

$$\leq 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k}$$

$$= 2 \cdot n \cdot \boxed{\sum_{k=1}^{n} \frac{1}{k}} = n\text{'te Harmonische Zahl } H_n$$

Es gilt: $H_n \in \Theta(\log(n))$

Satz 1: Der Erwartungswert $E[V]$ für die Anzahl von Vergleichen von randQS ist in $\mathcal{O}(n \cdot \log(n))$.

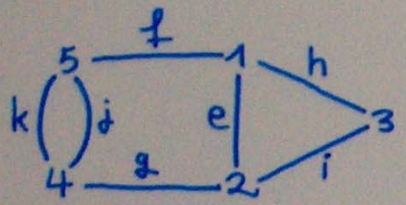
Bemerkungen:

- randQS ist ein Las Vegas Algorithmus. Er liefert stets das richtige Ergebnis, aber die Rechenzeit ist Zufallsvariable
- $E[V] \in \mathcal{O}(n \cdot \log(n))$ gilt für jeden Input. Keine Annahmen über Wahrscheinlichkeitsverteilung für Input S.
- Zufallsbits (= Münzwürfe) sollten als Rechenressource (wie Zeit und Speicher) angesehen werden.

Wieviele Zufallsbits sind nötig um ein Element y zufällig und gleichverteilt aus einer Menge S auszuwählen?

Beispiel 2: rand. Min-Cut Algorithmus

- Sei G zusammenhängender ungerichteter Multigraph (d.h. mehrere Kanten zwischen zwei Knoten, aber keine Schlingen ⟲)
- $V(G)$ = Menge der Knoten von G
  $E(G)$ = Menge der Kanten von G
  $|V(G)| = n$
- Beispiel:



- Ein **Cut** von G ist eine Menge $C \subseteq E(G)$, so dass $G \setminus C$ nicht mehr zusammenhängend ist.

Beispiel: $\{f, g\}$ ist Cut in obigen Graph.

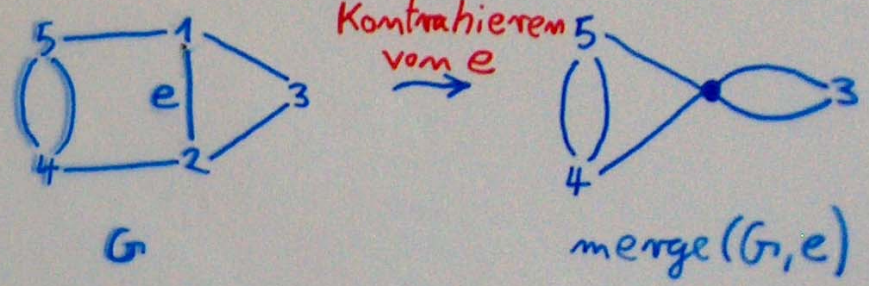
- Ein **Min-Cut** von G ist ein Cut C von G, so dass kein Cut C' von G mit $|C'| < |C|$ existiert.

Min-Cut-Size(G) = Größe eines (beliebigen) Min-Cut von G.

Bsp.: Min-Cut-Size$(G) = 2$ für obigen Graphen.

- Sei e eine Kante von G.
  merge(G, e) ist der Multigraph, der durch Verschmelzen der Endpunkte von e entsteht (entstehende Schleifen werden entfernt).

Beispiel:



G merge(G, e)

rand Min-Cut (o.B.d.A. $|V(G)| \geq 2$)

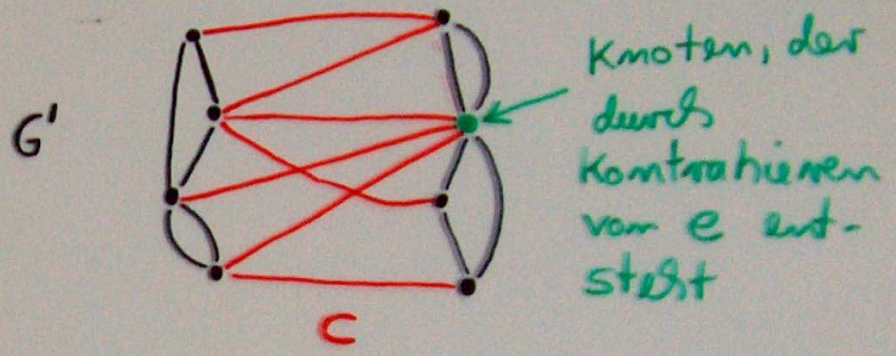
while "G hat noch mind. 3 Knoten" do
  wähle zufällig und gleichverteilt
  Kante $e \in E(G)$ aus.
  $G := merge(G, e)$
od.
% G hat nun noch genau zwei Knoten
output $E(G)$

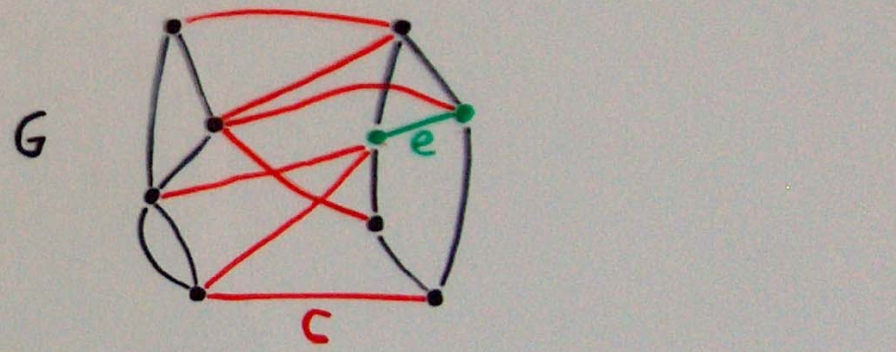
Bemerkungen:

- Zeitbedarf: stets $n-2$ ($n = |V(G)|$)
- Liefert rand Min-Cut stets einen Min-Cut von G zurück? Nein!

Lemma: Sei $e \in E(G)$.
Min-Cut-Size$(G) \leq$ Min-Cut-Size$(merge(G,e))$

Beweis: Sei C ein Min-Cut von $G' = merge(G, e)$





C ist auch ein Cut von G
$\Rightarrow$ Min-Cut-Size$(G) \leq |C| =$
$=$ Min-Cut-Size(G')

□

Analyse von rand Min-Cut

- Sei $v \in V(G)$.

  Grad von v: $d(v)$ = "Anzahl der Kanten, die v als Endpunkt haben"

Beobachtung: $|E(G)| = \frac{1}{2} \cdot \sum_{v \in V(G)} d(v)$

- Betrachte einen festen Min-Cut C von G.

  $k = |C|$ = Min-Cut-Size(G)

Angenommen, in einem Ablauf von rand Min-Cut wird nie eine Kante $e \in C$ kontrahiert.

→ rand Min-Cut liefert C als output und somit ein korrektes Ergebnis.

Wir schätzen die Wahrscheinlichkeit Prob[keine Kante aus C wird kontrahiert] ab.

- Sei $\mathcal{E}_i$ das Ereignis, dass im i-ten Durchlauf der while-Schleife keine Kante aus C kontrahiert wird

Prob[keine Kante aus C wird kontrahiert]

$= \text{Prob}\left[\bigwedge_{i=1}^{n-2} \mathcal{E}_i\right] =$

$= \text{Prob}[\mathcal{E}_1] \cdot \text{Prob}[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \text{Prob}[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2]$

$\cdot \ldots \cdot \text{Prob}\left[\mathcal{E}_{n-2} \mid \bigwedge_{j=1}^{n-3} \mathcal{E}_j\right]$

- Sei G_i der Graph nach dem $(i-1)$-ten Schleifendurchlauf (d.h. $G_1 = G$)

→ $|V(G_i)| = n - i + 1$

Min-Cut-Size$(G_i) \geq k$ (wegen Lemma) ← Min-Cut-Size(G)

Behauptung: $|E(G_i)| \geq \frac{k \cdot (n-i+1)}{2}$

Beweis: Sei $|E(G_i)| < \frac{k \cdot (n-i+1)}{2}$

$\rightarrow \frac{1}{2} \cdot \sum_{v \in V(G_i)} d(v) < \frac{1}{2} \cdot k \cdot |V(G_i)|$

$\rightarrow$ Es existiert ein Knoten $v \in V(G_i)$ mit $d(v) < k$

Da $\{e \in E(G_i) \mid v \text{ ist Endpunkt von } e\}$ ein Cut von G_i ist, folgt:

Min-Cut-Size$(G_i) < k$ ↯

- Angenommen, es gilt $\bigwedge_{j=1}^{i-1} \varepsilon_j$, d.h. in den Schleifendurchläufen $1, \dots, i-1$ wird keine Kante aus C kontrahiert.

$$\rightarrow \text{Prob}\left[\varepsilon_i \mid \bigwedge_{j=1}^{i-1} \varepsilon_j\right] = 1 - \frac{|C|}{|E(G_i)|}$$

$$\geq 1 - \frac{k}{\frac{k \cdot (n-i+1)}{2}} = 1 - \frac{2}{n-i+1}$$

$$= \frac{n-i-1}{n-i+1}$$

Also ergibt sich:

$$\text{Prob}[\text{keine Kante aus } C \text{ wird kontrahiert}]$$

$$= \prod_{i=1}^{n-2} \text{Prob}\left[\varepsilon_i \mid \bigwedge_{j=1}^{i-1} \varepsilon_j\right]$$

$$\geq \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} =$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \dots \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}$$

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

<u>Satz 2</u>: Ein Ablauf von rand Min-Cut liefert mit Wahrscheinlichkeit mindestens $\frac{2}{n^2}$ einen Min-Cut zurück. (n = Anzahl der Knoten des Eingabegraphen)

Wahrscheinlichkeitsverstärkung
(Probability amplification)

Wir wiederholen nun rand Min-Cut $\frac{n^2}{2}$ mal ($\rightarrow$ Gesamtlaufzeit: $\frac{n^2}{2}(n-2)$) und liefern den kleinsten Cut aus den einzelnen Wiederholungen zurück.

$$\text{Prob}[\text{keine der } \tfrac{n^2}{2} \text{ Wiederholungen liefert einen Min-Cut}]$$

$$\leq \left(1-\frac{2}{n^2}\right)^{\frac{n^2}{2}} < \frac{1}{e} \quad (e = \text{Eulersche Konstante})$$

Bei $10 \cdot n^2$ Wiederholungen ergibt sich eine Fehlerwahrscheinlichkeit von $< \left(\frac{1}{e}\right)^{20}$

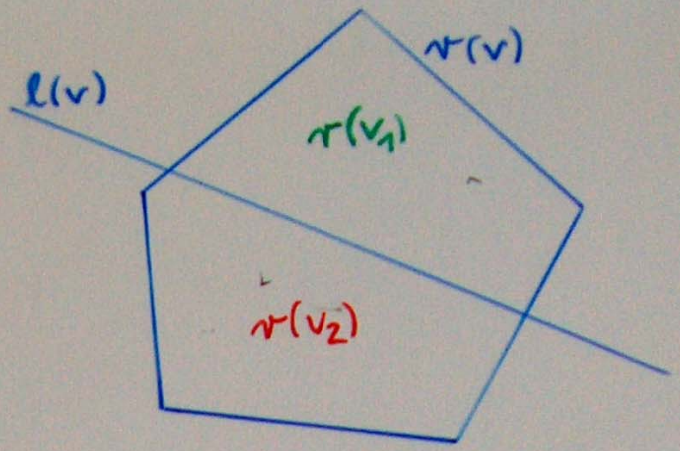
Bemerkungen:

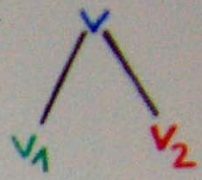
- Wahrscheinlichkeitsverstärkung ist unter Beibehaltung einer polynomiellen Laufzeit möglich, falls Fehlerwahrscheinlichkeit für einen Durchlauf $< 1 - \frac{1}{p(n)}$ für ein Polynom $p(n)$ gilt:
$$\left(1-\frac{1}{p(n)}\right)^{p(n)} < \frac{1}{e}$$
- rand Min-Cut ist ein Monte-Carlo-Algorithmus: Er liefert nur mit einer Wahrscheinlichkeit < 1 ein richtiges Ergebnis.

Beispiel 3: Binäre Planare Partitionen

Def: Eine <u>binäre planare Partition</u> ist ein binärer Baum P.

- Jedem Knoten v von P ist eine (evtl. unendliche) Region $r(v) \subseteq \mathbb{R}^2$ zugeordnet.
- Für die Wurzel u von P gilt $r(u) = \mathbb{R}^2$
- Jedem internen Knoten v von P (d.h. jedem Nicht-Blatt) ist weiterhin eine Gerade $\ell(v)$ zugeordnet, die $r(v)$ schneidet.
  - $\rightarrow$ $\ell(v)$ teilt $r(v)$ in zwei Regionen r_1 und r_2 auf. Diese Regionen sind den beiden Kindern von v zugeordnet.





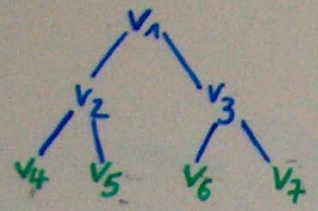
Sei nun $S = \{s_1, \dots, s_n\}$ eine Menge von Geradensegmenten (endlicher Länge) mit $s_i \cap s_j = \emptyset$ für $i \neq j$

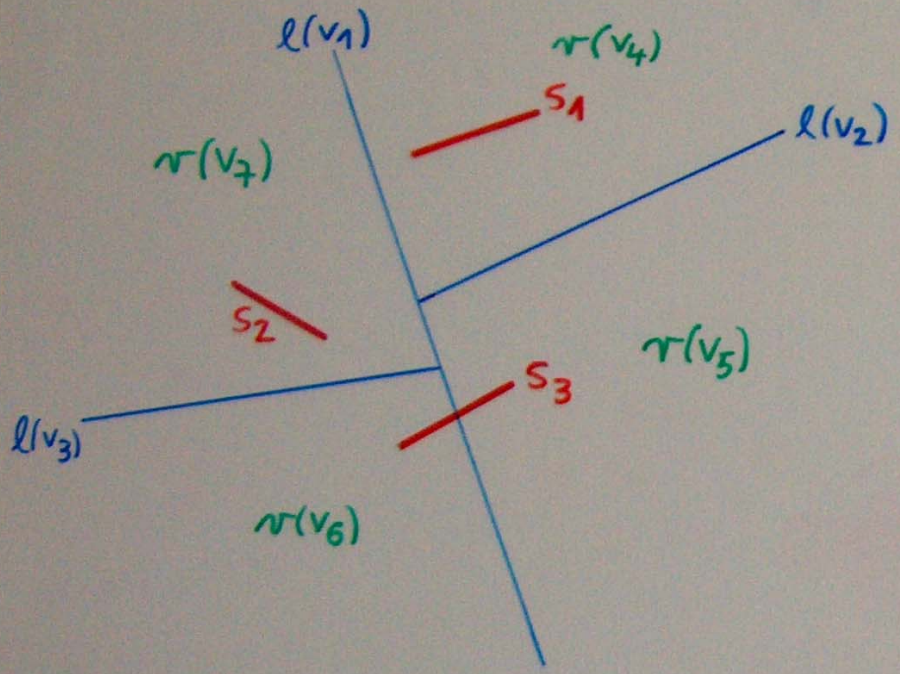
Eine <u>binäre planare Partition P von S</u> ist eine binäre planare Partition P, so dass für jedes Blatt v von P gilt:

Es gibt höchstens ein $1 \leq i \leq n$ mit $s_i \cap r(v) \neq \emptyset$.

Beispiel

P:





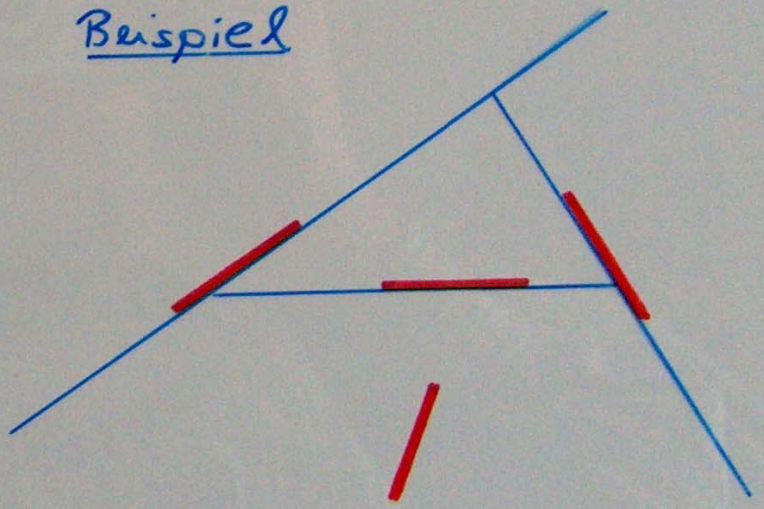
Ziel: Finde binäre planare Partition P von S mit möglichst wenig Baumknoten (bzw. wenig Regionen).

Für ein Geradensegment s_i sei $\ell(s_i)$ die Gerade mit $s_i \subseteq \ell(s_i)$.

Eine Autopartition von $S = \{s_1, \dots, s_n\}$ ist eine binäre planare Partition P von S, so dass für jeden internen Knoten v des Baums P ein i mit $\ell(v) = \ell(s_i)$ existiert.

Bemerkung Für eine Autopartition P von S kann natürlich ein $s_i \in S$ genau auf der Grenze von 2 Regionen r_1 und r_2 liegen. Wir ordnen dann s_i entweder r_1 oder r_2 zu.

Beispiel



Algorithmus: rand Auto

Input: Menge $S = \{s_1, \dots, s_n\}$ von Geradensegmenten, $s_i \cap s_j = \emptyset$ für $i \neq j$.

Output: Eine Autopartition P von S

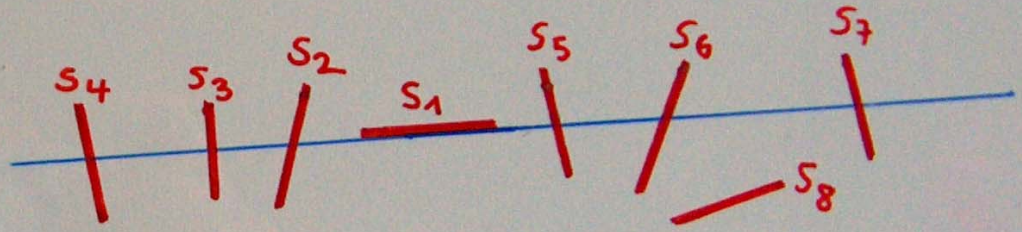
(1) Wähle zufällig und gleichverteilt eine Permutation π von $\{1, \dots, n\}$

(2) **while** "Es gibt Region r, die mit mind. zwei $s \in S$ einen nichtleeren Schnitt hat" **do**
Sei $k = \min\{i \mid r \cap s_{\pi(i)} \neq \emptyset\}$
Teile r mit $\ell(s_{\pi(k)})$ auf.
od

Analyse von rand Auto

- Für Geradensegmente $s_i, s_j \in S$ sei

$$index(s_i, s_j) = \begin{cases} \infty & \text{falls } \ell(s_i) \text{ nicht } s_j \text{ schneidet.} \\ k & \text{falls } \ell(s_i) \text{ genau } k-1 \text{ Geradensegmente schneidet, bevor } \ell(s_i) \text{ auf } s_j \text{ trifft.} \end{cases}$$



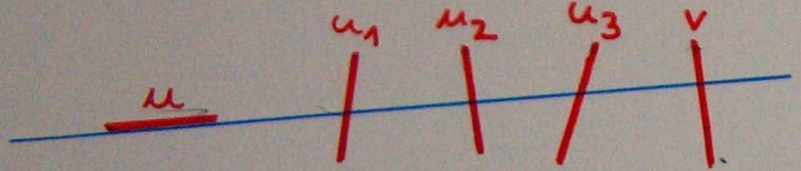
$index(s_1, s_4) = index(s_1, s_7) = 3$

$index(s_1, s_8) = \infty$

- $s_i \dashv s_j \Leftrightarrow \ell(s_i)$ schneidet s_j in der Autopartition, die rand-Auto erzeugt.

Beobachtung: Sei $index(u,v) = k$ für $u, v \in S$ und seien $u_1, \dots, u_{k-1} \in S$ die Geradensegmente, die $l(u)$ schneidet, bevor $l(u)$ auf v trifft. Dann gilt:

$u \dashv v \Rightarrow$ u steht in der Zufallspermutation π vor allen Elementen $u_1, \dots, u_{k-1}(, v)$



$Prob[u \text{ steht in } \pi \text{ vor } u_1, \dots, u_{k-1}(, v)]$

$= \frac{1}{k(+1)}$

$C_{u,v} := \begin{cases} 1 & \text{falls } u \dashv v \\ 0 & \text{sonst} \end{cases}$

(Indikatorvariable)

$$E[C_{u,v}] = Pr[u \dashv v] \leq \frac{1}{k(+1)} = \frac{1}{index(u,v)(+1)}$$

Sei P die Autopartition von S, die randAuto produziert.

Anzahl der Regionen von P =

$= n +$ "Anzahl der Schnitte von Geradensegmenten"

↑ Anzahl d. Geradensegmente

$\Rightarrow E[\text{Anzahl der Regionen}]$

$$= n + E\left[\sum_{u \in S} \sum_{\substack{v \in S \\ v \neq u}} C_{u,v}\right]$$

$$= n + \sum_{u \in S} \sum_{\substack{v \in S \\ v \neq u}} E[C_{u,v}]$$

$$\leq n + \sum_{u \in S} \sum_{\substack{v \in S \\ v \neq u}} \frac{1}{index(u,v)(+1)}$$

Beobachtung: Für ein $k \in \mathbb{N}$ und $u \in S$ gibt es höchstens zwei $v \in S$ mit $index(u,v) = k$

$$\Rightarrow \sum_{\substack{v \in S \\ v \neq u}} \frac{1}{index(u,v)(+1)} \leq \sum_{k=1}^{n-1} \frac{2}{k(+1)}$$

$$\Rightarrow E[\text{Anzahl der Regionen von } P]$$

$$\leq n + n \cdot \sum_{k=1}^{n-1} \frac{2}{k(+1)}$$

$$\leq n + 2 \cdot n \sum_{k=1}^{n} \frac{1}{k} =$$

$$= n + 2 \cdot n \cdot H_n \in \Theta(n \cdot \log(n))$$

Satz: Der Erwartungswert für die Größe der von rand Auto konstruierten Autopartition von $S = \{s_1, \ldots, s_n\}$ ist in $O(n \cdot \log(n))$.

Korollar: Für jede Menge $S = \{s_1, \ldots, s_n\}$ von Geradensegmenten mit $(i \neq j \Rightarrow s_i \cap s_j = \emptyset)$ existiert eine lineare planare Partition von S der Größe $O(n \cdot \log(n))$.

Begründung: Eine Zufallsvariable X muß mit Wahrscheinlichkeit > 0 einen Wert $\geq E[X]$ annehmen können.

Probabilistische Methode zum Beweis der Existenz eines bestimmten kombinatorischen Objekts (hier: einer linearen planaren Partition)

Randomisierte Komplexitätsklassen

- Eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse **RP** (randomisierte Polynomialzeit), falls ein randomisierter Algorithmus A existiert mit:
  - Es gibt ein Polynom $p(n)$, so dass A auf jeder Eingabe $x \in \Sigma^*$ höchstens $p(|x|)$ Schritte rechnet.
  - $x \in L \Rightarrow \text{Prob}[A \text{ akzeptiert } x] \geq \frac{1}{2}$
  - $x \notin L \Rightarrow \text{Prob}[A \text{ akzeptiert } x] = 0$

Bemerkung: Die Wahrscheinlichkeitsschranke $\frac{1}{2}$ kann durch Wahrscheinlichkeitsverstärkung erhöht werden.

- Eine Sprache L gehört zur Klasse **coRP** genau dann, wenn $\bar{L} \in RP$.

- Eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse **ZPP** (Zero-Error Probabilistic Polynomial Time), falls ein randomisierter (Las Vegas) Algorithmus existiert mit:
  - Es gibt ein Polynom $p(n)$, so dass für jede Eingabe $x \in \Sigma^*$ gilt:
  - $E[\text{Rechenzeit von } A \text{ bei Eingabe } x] \leq p(n)$
  - $x \in L \Rightarrow \text{Prob}[A \text{ akzeptiert } x] = 1$
  - $x \notin L \Rightarrow \text{Prob}[A \text{ akzeptiert } x] = 0$

Satz: $ZPP = RP \cap coRP$

Beweis: Übungen