Karlsruher Institut für Technologie

Algorithmen II
Vorlesung am 20.11.2012

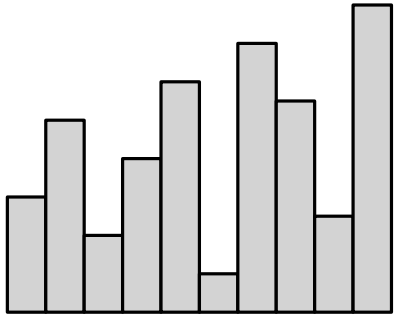
Randomisierte Algorithmen

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

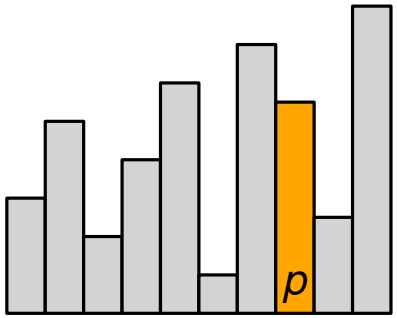
Definition

Definition 8.1: Ein Algorithmus, der im Laufe seiner Ausführung zufällige Entscheidungen trifft, heißt *randomisierter Algorithmus*.

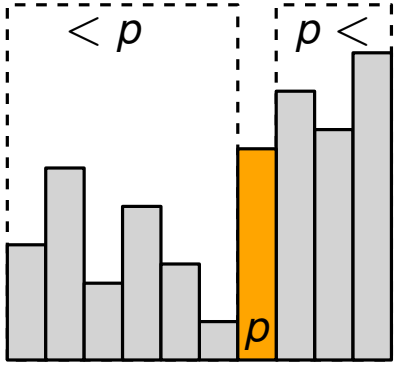
Bekanntes Beispiel: Quicksort



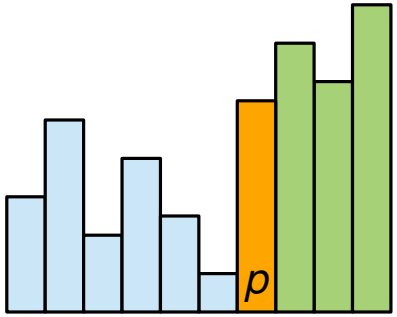
Eingabe:
Unsortierte Liste



1. Schritt
Wähle Element zufällig.



2. Schritt:
Umsortieren



3. Schritt:
Rekursives Vorgehen auf Teilmengen

Laufzeit:

- Schlimmster Fall: $\Theta(n^2)$.
- Zu erwartende Laufzeit: $\mathcal{O}(n \cdot \log n)$

Bemerkung: Algorithmus liefert immer korrektes Ergebnis.

Arten von randomisierten Algorithmen

Las Vegas Algorithmus

- Liefert immer korrektes Ergebnis.
- Laufzeit variiert.
- Beispiel: Quicksort

Eingabe: Unsortierte Liste

1. Schritt Wähle Element zufällig.

2. Schritt: Umsortieren

3. Schritt: Rekusives Vorgehen auf Teilmengen

Monte Carlo Algorithmus (Eselsbrücke: mostly correct)

- Kann auch falsches Ergebnis liefern.
- Betrachte Wahrscheinlichkeit für Fehler.
- Für Entscheidungsproblem, d.h. nur *JA*/*NEIN*-Antwort möglich, gibt es zwei Arten:
  - **beidseitig**: Für beide möglichen Antworten gibt es Wahrscheinlichkeit > 0, dass Antwort falsch ist.
  - **einseitig**: Für eine der beiden Antworten ist Wahrscheinlichkeit gleich Null, dass Antwort fehlerbehaftet ist.

    Beispiel: Die Antwort *JA* ist immer richtig, die Antwort *NEIN* kann auch falsch sein.

Wahrscheinlichkeitsklassen

Die Klasse $\mathcal{RP}$ (randomisiert polynomial) enthält alle Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I von Π gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \geq \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] = 0 \end{cases}$$

Die Klasse $\mathcal{PP}$ (probabilistic polynomial) enthält alle Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] > \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] < \frac{1}{2} \end{cases}$$

Die Klasse $\mathcal{BPP}$ (bounded error PP) ist die Klasse der Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \geq \frac{3}{4} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \leq \frac{1}{4} \end{cases}$$

- Y_Π ist die Menge der sogenannten „JA-Beispiele“ von Π.
- Dabei entspricht $\Pr[A(I) \text{ ist „JA“}]$ der Wahrscheinlichkeit, dass die Antwort, die A bei der Eingabe von I gibt, „JA“ ist.

Wahrscheinlichkeitsklassen

Die Klasse $\mathcal{RP}$ (randomisiert polynomial) enthält alle Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I von Π gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \geq \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] = 0 \end{cases}$$

einseitiger Monte Carlo Algorithmus

Die Klasse $\mathcal{PP}$ (probabilistic polynomial) enthält alle Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] > \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] < \frac{1}{2} \end{cases}$$

beidseitiger Monte Carlo Algorithmus

Die Klasse $\mathcal{BPP}$ (bounded error PP) ist die Klasse der Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \geq \frac{3}{4} \\ I \notin Y_\Pi \longrightarrow \Pr[A(I) \text{ ist „JA“}] \leq \frac{1}{4} \end{cases}$$

beidseitiger Monte Carlo Algorithmus

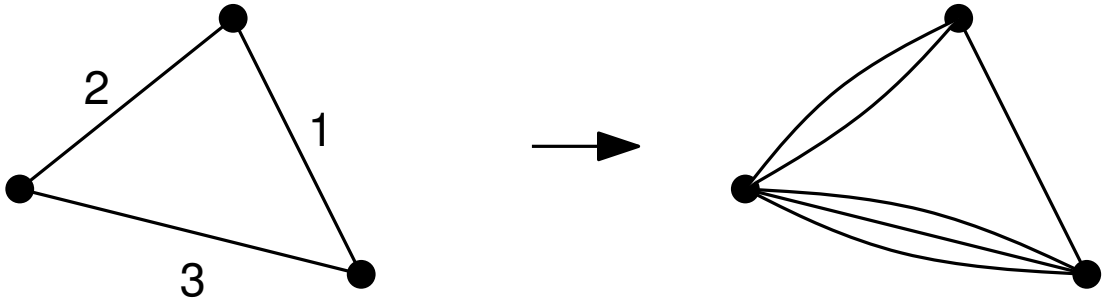
- Y_Π ist die Menge der sogenannten „JA-Beispiele“ von Π.
- Dabei entspricht $\Pr[A(I) \text{ ist „JA“}]$ der Wahrscheinlichkeit, dass die Antwort, die A bei der Eingabe von I gibt, „JA“ ist.

Monte Carlo Algorithmus für MinCut

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problemdefinition

Fasse $G = (V, E)$ mit Kantengewichtsfunktion $c : E \to \mathbb{N}$ als Multigraph auf, d.h. für $\{u, v\} \in E$ gibt es $c(\{u, v\})$ Kanten:



Problem MinCut: Sei $G = (V, E)$ mit $c : E \to \mathbb{N}$ ein solcher Multigraph. Gesucht ist eine Partition V_1 und V_2 von V, sodass

$$\text{cutsize}(V_1, V_2) := \left|\{\{u, v\} \in E \mid u \in V_1 \text{ und } v \in V_2\}\right|$$

minimal ist.

Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

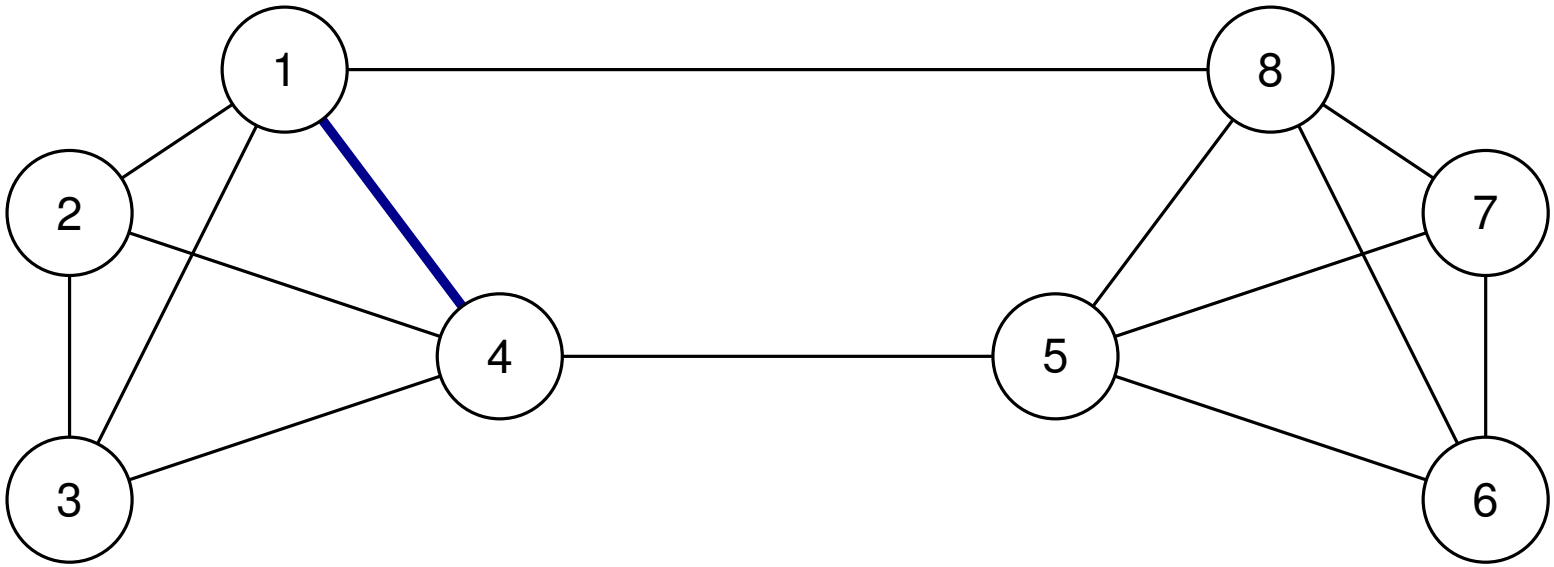
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

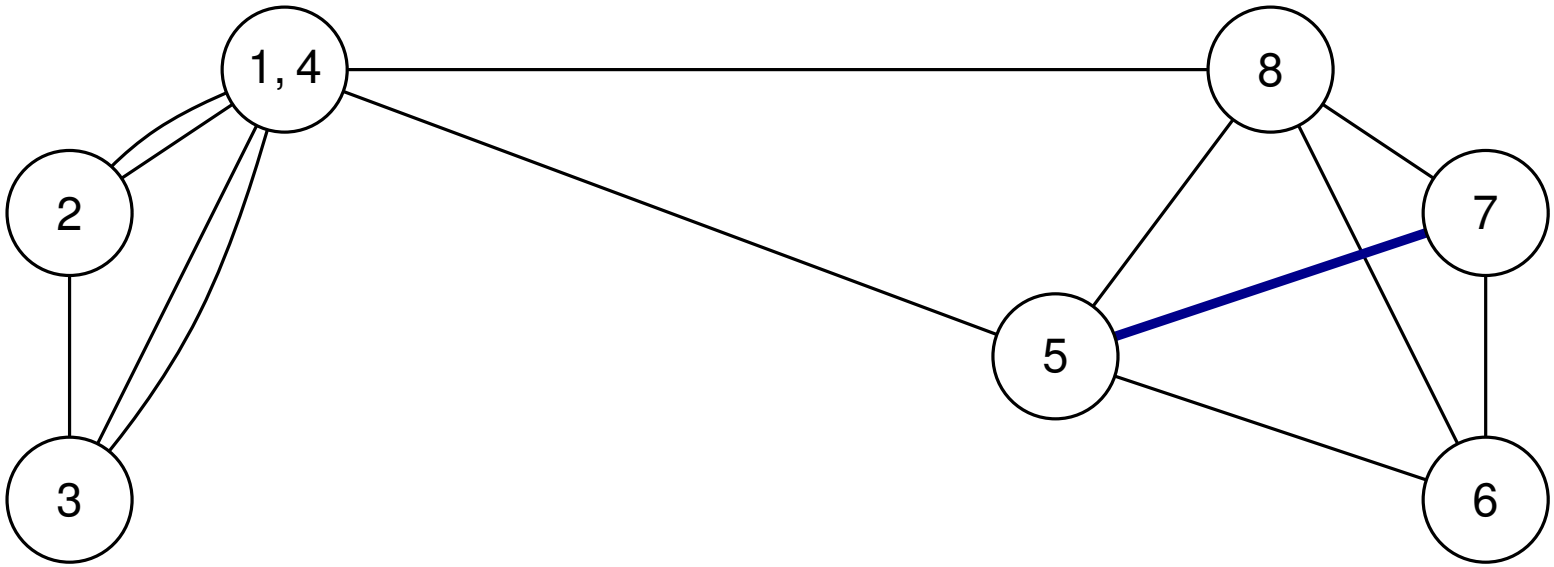
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

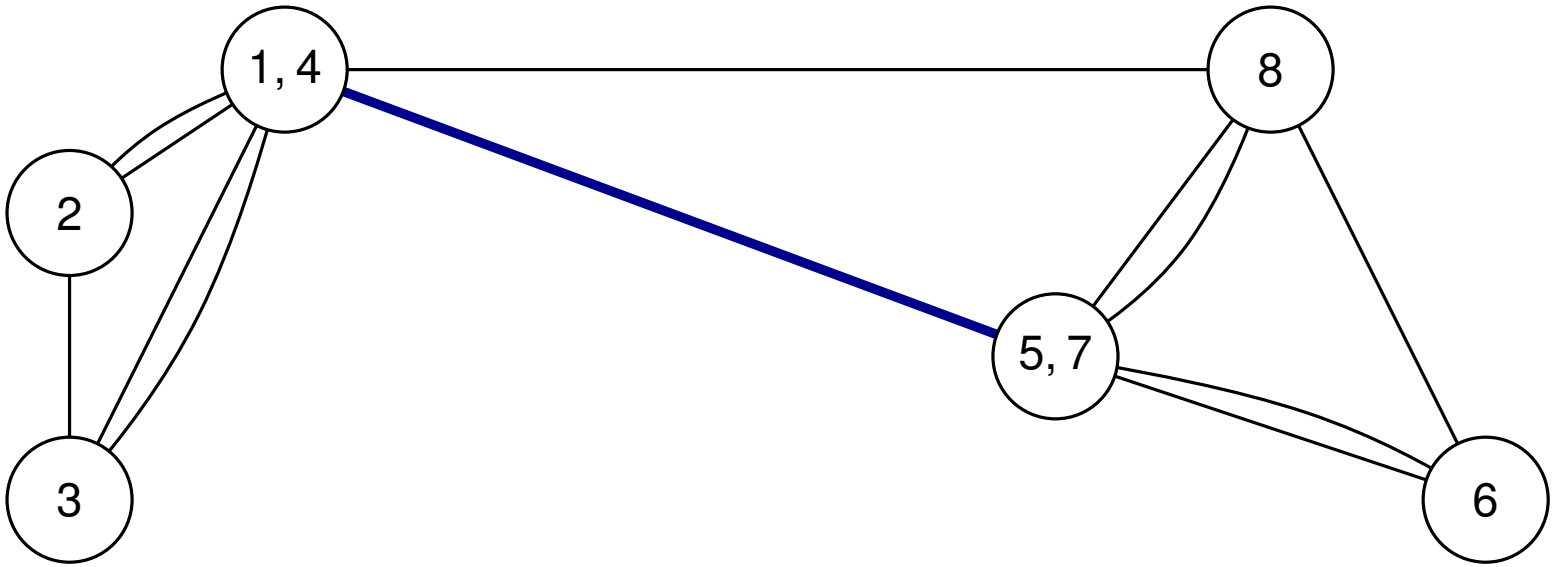
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

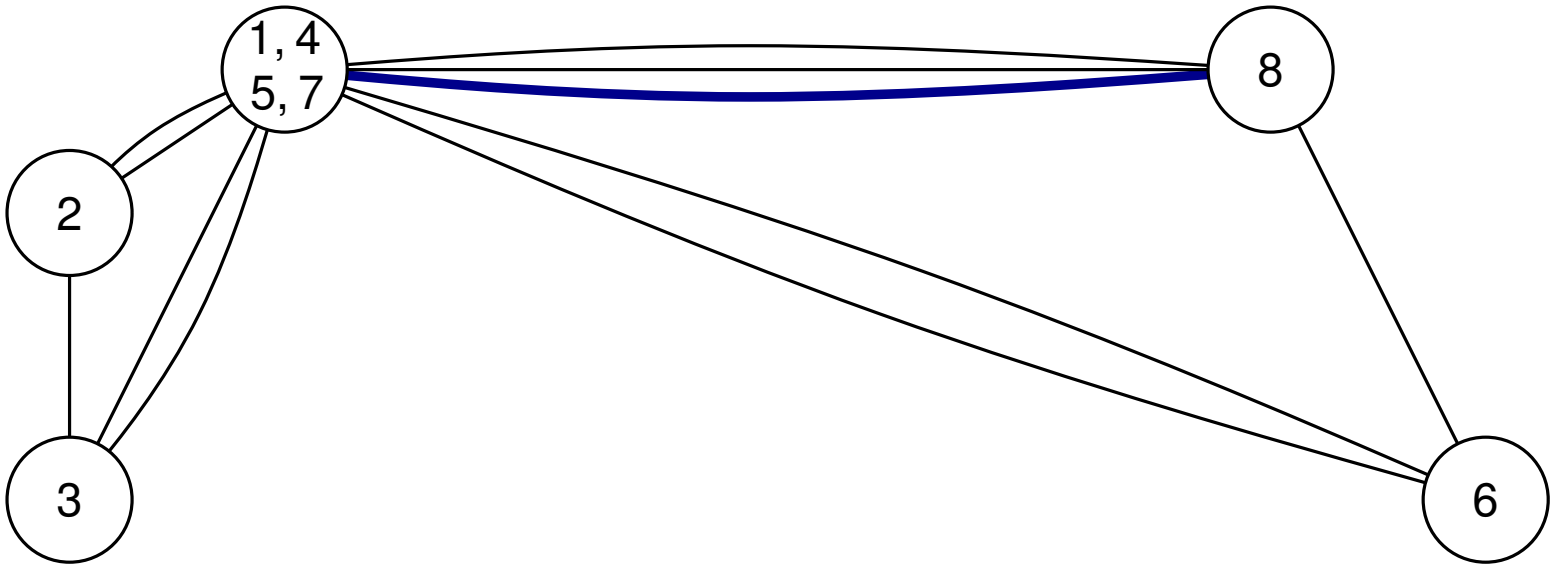
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

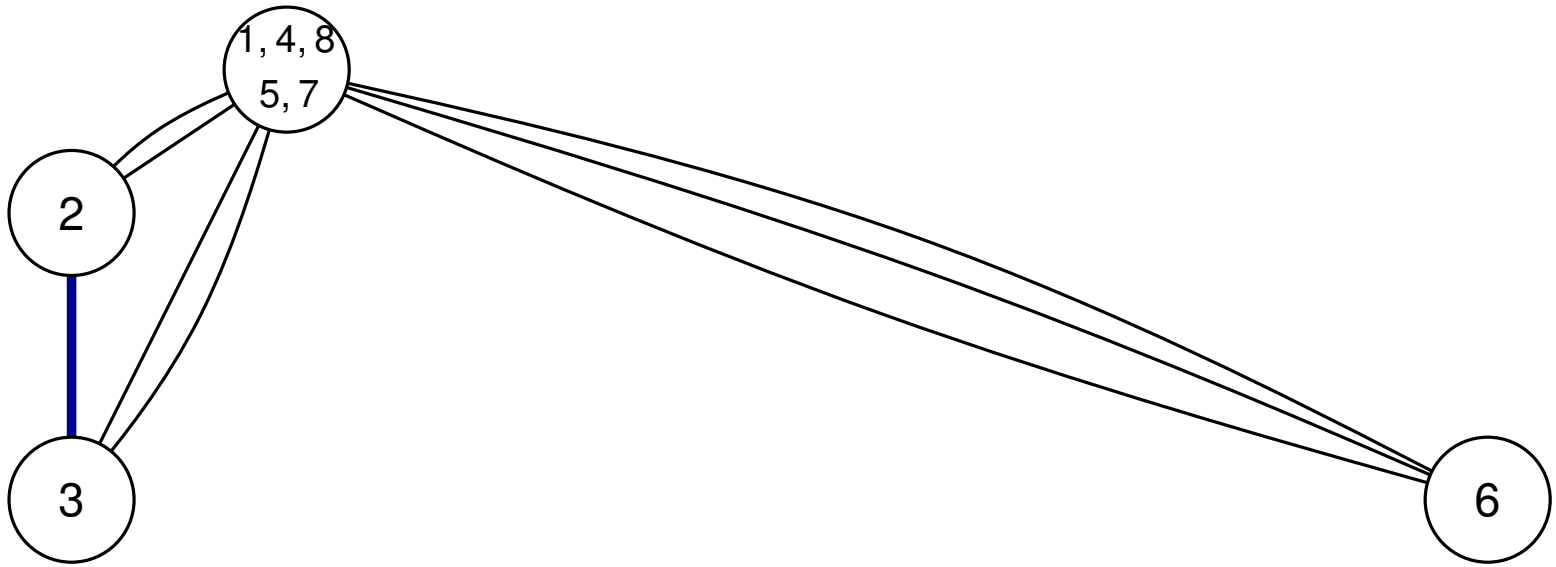
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

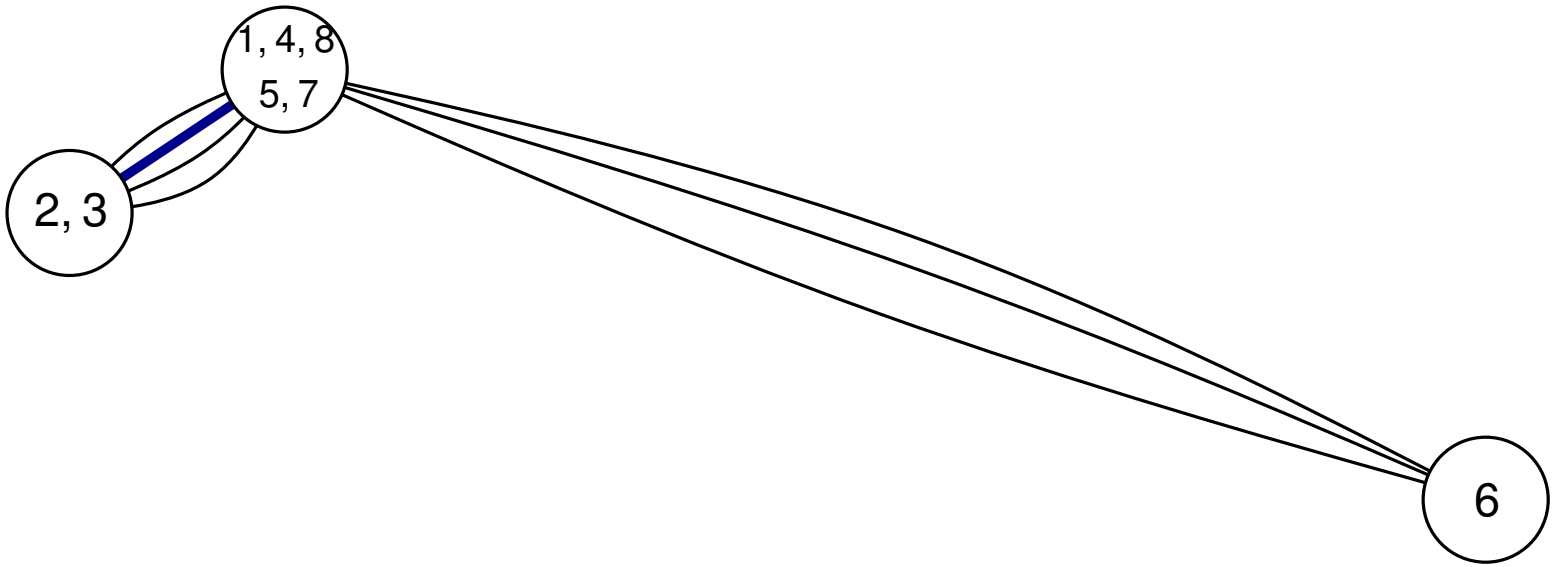
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

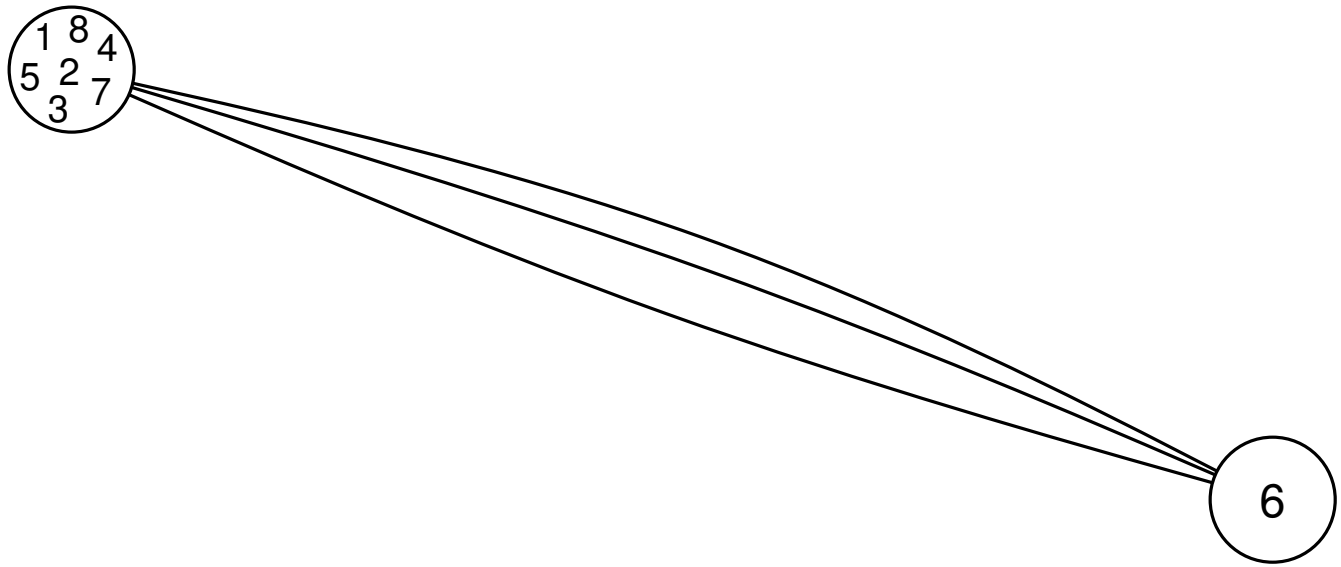
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Algorithmus

RANDOM MINCUT

Eingabe: Multigraph $G = (V, E)$

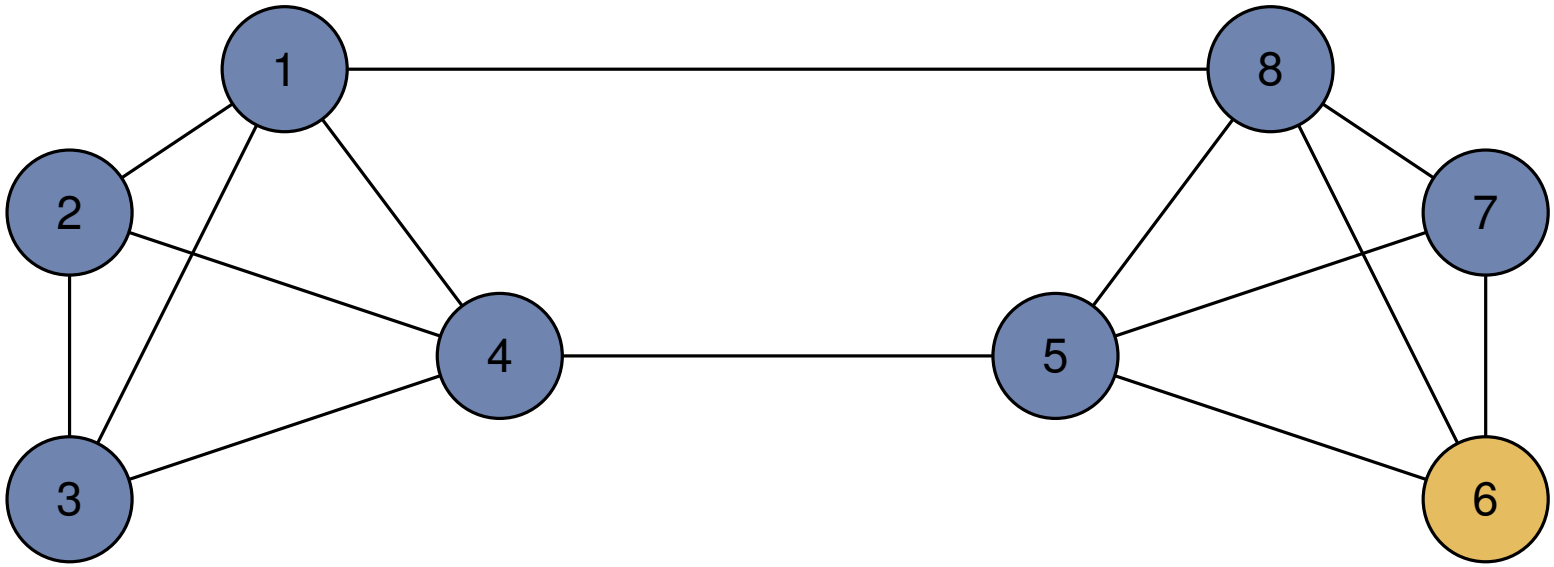
Ausgabe: Schnitt in Form eines Graphen mit zwei Superknoten

solange $|V| > 2$ **tue**

  $e \leftarrow$ zufällige Kante in E

  Bilde neuen Graph $G = (V, E)$, der entsteht, wenn die Endknoten von e verschmolzen werden und alle Kanten zwischen Endknoten von e entfernt werden

Gebe V zurück.



Qualität von RANDOM MINCUT

Satz 8.7.
Die Wahrscheinlichkeit, dass RANDOM MINCUT einen bestimmten minimalen Schnitt (der möglicherweise auch der einzige ist) findet, mit der Bedingung, dass alle Kanten die gleiche Wahrscheinlichkeit haben gewählt zu werden, ist größer als $\frac{2}{n^2}$, wobei $|V| = n$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Qualität von RANDOM MINCUT

Satz 8.7.
Die Wahrscheinlichkeit, dass RANDOM MINCUT einen bestimmten minimalen Schnitt (der möglicherweise auch der einzige ist) findet, mit der Bedingung, dass alle Kanten die gleiche Wahrscheinlichkeit haben gewählt zu werden, ist größer als $\frac{2}{n^2}$, wobei $|V| = n$.

Folgerung 8.8.
Wendet man RANDOM MINCUT nur $n-\ell$ Schritte lang an, d.h. man stoppt, wenn ℓ Knoten übrig sind, so ist die Wahrscheinlichkeit, dass bis dahin keine Kante eines bestimmten minimalen Schnitts (V_1, V_2) gewählt wurde, mindestens

$$\frac{\binom{\ell}{2}}{\binom{n}{2}}, \qquad \text{d.h. in } \Omega\left(\left(\frac{\ell}{n}\right)^2\right) .$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenfassung

- Wenn Wahl einer zufälligen Kante in $\mathcal{O}(n)$ realisierbar, dann hat der Algorithmus eine Laufzeit von $\mathcal{O}(n^2)$.

$\Rightarrow$ Bessere Laufzeit als deterministische Variante ($\mathcal{O}(n^2 \log n + n \cdot m)$), siehe Skript, Kapitel 3.

- Wendet man RANDOM MINCUT $\frac{n^2}{2}$ mal unabhängig voneinander an, so ergibt sich:

$$Pr[\text{Bestimmter Schnitt nicht gefunden}] = \left(1 - \frac{2}{n^2}\right)^{\frac{n^2}{2}} < \frac{1}{e}$$

Eulersche Zahl

- Allerdings $\mathcal{O}(n^4)$-Algorithmus

$\Rightarrow$ Schlechter als deterministische Variante.

Ein effizienterer randomisierter MinCut-Algorithmus

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Fast Random MinCut

Eingabe: Graph $G = (V, E)$ als Multigraph, $|V| = n$

Ausgabe: Schnitt

wenn $n \leq 6$ **dann**

  berechne direkt deterministisch einen MinCut

sonst

  $\ell \leftarrow \left\lceil \frac{n}{\sqrt{2}} \right\rceil$

  $G_1 \leftarrow$ Random MinCut(bis ℓ Knoten übrig)

  $G_2 \leftarrow$ Random MinCut(bis ℓ Knoten übrig)

  $C_1 \leftarrow$ Fast Random MinCut(G_1) (rekursiv)

  $C_2 \leftarrow$ Fast Random MinCut(G_2) (rekursiv)

  Gib den kleineren der beiden Schnitte C_1 und C_2 aus.

Fast Random MinCut

Eingabe: Graph $G = (V, E)$ als Multigraph, $|V| = n$

Ausgabe: Schnitt

wenn $n \leq 6$ **dann**

  berechne direkt deterministisch einen MinCut

sonst

  $\ell \leftarrow \left\lceil \frac{n}{\sqrt{2}} \right\rceil$

  $G_1 \leftarrow$ Random MinCut(bis ℓ Knoten übrig)
  $G_2 \leftarrow$ Random MinCut(bis ℓ Knoten übrig) — A

  $C_1 \leftarrow$ Fast Random MinCut(G_1) (rekursiv)
  $C_2 \leftarrow$ Fast Random MinCut(G_2) (rekursiv) — B

  Gib den kleineren der beiden Schnitte C_1 und C_2 aus.

Satz 8.9: Fast Random MinCut hat eine Laufzeit $\mathcal{O}(n^2 \log n)$.

Beweis: Laufzeit $T(n)$ ergibt sich aus folgender Rekursionsabschätzung:

$$T(n) = \underbrace{2 \cdot T\left(\left\lceil \frac{n}{\sqrt{2}} \right\rceil\right)}_{\text{B}} + \underbrace{c \cdot n^2}_{\text{A}}$$

Fast Random MinCut

Eingabe: Graph $G = (V, E)$ als Multigraph, $|V| = n$

Ausgabe: Schnitt

wenn $n \leq 6$ **dann**

  berechne direkt deterministisch einen MinCut

sonst

  $\ell \leftarrow \left\lceil \frac{n}{\sqrt{2}} \right\rceil$

  $G_1 \leftarrow$ Random MinCut(bis ℓ Knoten übrig)

  $G_2 \leftarrow$ Random MinCut(bis ℓ Knoten übrig)

  $C_1 \leftarrow$ Fast Random MinCut(G_1) (rekursiv)

  $C_2 \leftarrow$ Fast Random MinCut(G_2) (rekursiv)

  Gib den kleineren der beiden Schnitte C_1 und C_2 aus.

Satz 8.10: Die Wahrscheinlichkeit, dass Fast Random MinCut einen minimalen Schnitt findet, ist in $\Omega\left(\frac{1}{\log n}\right)$.

Maximum Satisfiability Problem

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problemdefinition

Problem MAXIMUM SATISFIABILITY (MAXSAT):
Gegeben: Menge von m Klauseln über n Variablen.
Gesucht: Wahrheitsbelegung, die eine maximale Anzahl von Klauseln erfüllt.

Bereits $\mathcal{NP}$-schwer, wenn Anzahl der Literale auf zwei pro Klausel beschränkt.

→ MAX-2-SAT

Beispiel:

1. Klausel: $X_1 \vee \overline{X_2}$ 2. Klausel: $\overline{X_1} \vee \overline{X_2}$

3. Klausel: $X_1 \vee X_2$ 4. Klausel: $\overline{X_1} \vee X_3$

5. Klausel: $X_2 \vee \overline{X_3}$

Nicht alle Klauseln gleichzeitig erfüllbar:

- Für $X_1 = \mathit{falsch}$ kann 1. Klausel nicht mit 3. Klausel gleichzeitig erfüllt sein.
- Für $X_1 = \mathit{wahr}$ kann 5. Klausel nicht mit 2. und 4. Klausel gleichzeitig erfüllt sein.

Optimale Belegung: $X_1 = \mathit{wahr}$, $X_2 = \mathit{falsch}$, $X_3 = \mathit{wahr}$

Algorithmus Random Sat

Vorgehen: Für jede Variable $x \in V$ setze $x := \mathit{wahr}$ mit der Wahrscheinlichkeit $\frac{1}{2}$.

Satz 8.16.
Für eine Instanz I von MAX SAT mit m Klauseln, in der jede Klausel mindestens k Literale enthält, erfüllt der erwartete Wert der Lösung von RANDOM SAT:

$$E[X_{RS}(I)] \geq \left(1 - \frac{1}{2^k}\right) \cdot m,$$

wobei $X_{RS}(I)$ die Zufallsvariable bezeichnet, die den Wert der Lösung von RANDOM SAT bei der Eingabe von I angibt.

Beweis:

- Wahrscheinlichkeit, dass Klausel mit k Literalen nicht erfüllt wird, ist $\frac{1}{2^k}$.
- Entsprechend ist Wahrscheinlichkeit, dass Klausel mit mindestens k Literalen erfüllt wird mindestens $1 - \frac{1}{2^k}$.
- Damit ist der erwartete Beitrag einer Klausel zu $E[X_{RS}(I)]$ mindestens $1 - \frac{1}{2^k}$.
- Es folgt die Behauptung.

Algorithmus Random Sat

Vorgehen: Für jede Variable $x \in V$ setze $x :=$ *wahr* mit der Wahrscheinlichkeit $\frac{1}{2}$.

Satz 8.16.
Für eine Instanz I von MAX SAT mit m Klauseln, in der jede Klausel mindestens k Literale enthält, erfüllt der erwartete Wert der Lösung von RANDOM SAT:

$$E[X_{RS}(I)] \geq \left(1 - \frac{1}{2^k}\right) \cdot m,$$

wobei $X_{RS}(I)$ die Zufallsvariable bezeichnet, die den Wert der Lösung von RANDOM SAT bei der Eingabe von I angibt.

Korollar 8.17: RANDOM SAT ist 2-approximativ, d.h.

$$\frac{OPT(I)}{E[X_{\mathrm{RS}(I)}]} \leq 2$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner