Julius-Maximilians-
UNIVERSITÄT WÜRZBURG
Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität
Institut für Informatik

Algorithmische Graphentheorie

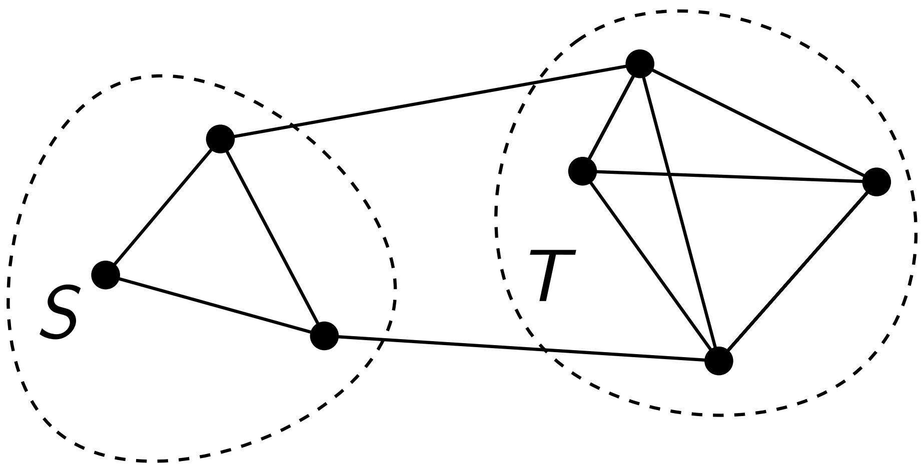
Sommersemester 2021

8. Vorlesung

Randomisierte Algorithmen für MinCut

MinCut – kleinste Schnitte

Def. Gegeben sei ein ungerichteter Multigraph $G = (V, E)$. Gesucht ist eine Zerlegung (S, T) von V mit $S, T \neq \emptyset$, so dass die Anzahl der Kanten $uv \in E$ mit $u \in S$ und $v \in T$ möglichst klein ist.



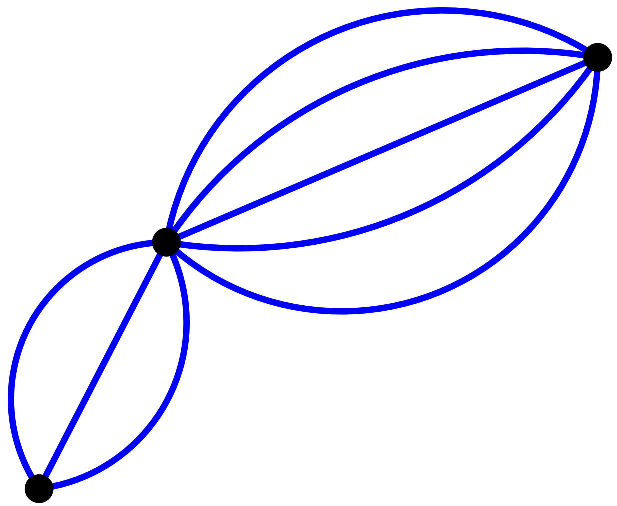
Motivation:

Z.B. Finden von Schwachstellen in Kommunikationsnetzwerken.

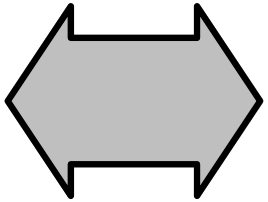
Beachte: Im Gegensatz zu s-t-Schnitten ist hier *kein* zu trennendes Knotenpaar (s, t) vorgegeben.

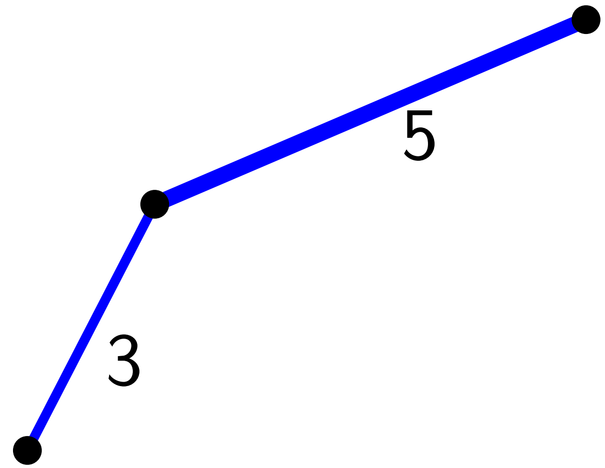
O.B.d.A. Multigraph G zusammenhängend, und $n := |V| \geq 2$.

Multigraphen vs. Kantengewichte



Algorithmenbeschreibung
Analyse





Implementierung
(vgl. Übung)

Gewichtetes $\textsc{MinCut}$:

... minimiere Kosten $\displaystyle\sum_{\substack{u \in S, v \in T \\ uv \in E}} c(uv)$, wobei $c\colon E \to \mathbb{N}$.

... Algorithmen funktionieren auch für gewichtetes $\textsc{MinCut}$

Naiver (deterministischer) Ansatz

Übung:
Geben Sie einen Polynomialzeit-Algorithmus für $\textsc{MinCut}$ an!

- Für jedes Paar $\{s, t\} \in \binom{V}{2}$,
  berechne kleinsten s-t-Schnitt mittels Edmonds-Karp.
- Gib den kleinsten dieser s-t-Schnitte zurück.

Gesamtlaufzeit: $O(V^2 \cdot VE^2) \subseteq O(V^7)$.

Es geht auch in $O(V^6)$ Zeit...

– Wähle s fest.
– Berechne den kleinsten (s, t)-Schnitt für jedes $t \in V \setminus \{s\}$.

Algorithmus Contract

David R. Karger

Contract(zsghd. Multigraph $G = (V, E)$)

$H \leftarrow G$

while H hat mehr als zwei Knoten **do**

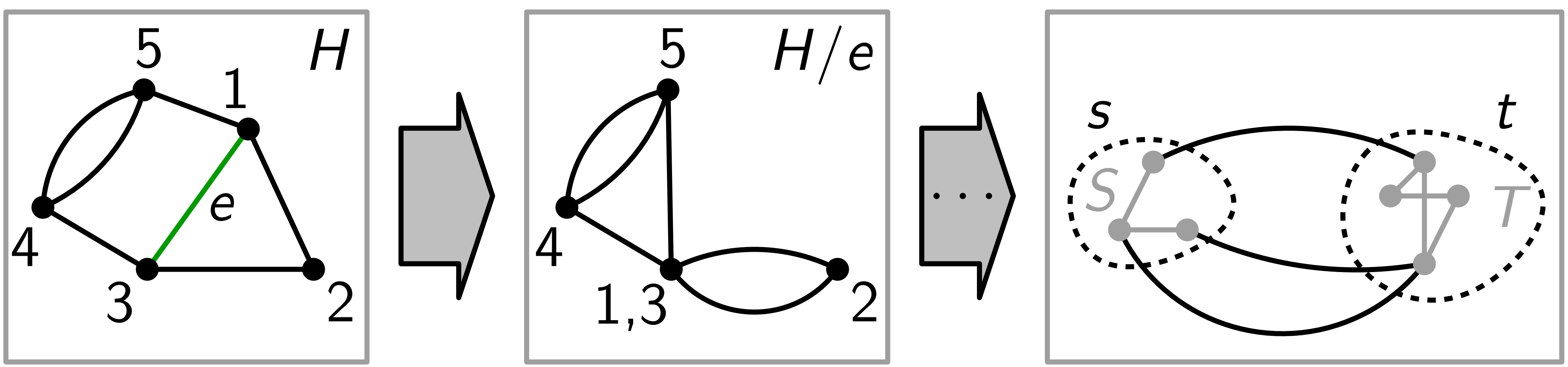
  wähle Kante e in H zufällig und gleichverteilt

  $H \leftarrow H/e$ /* kontrahiere e */

return Zerlegung (S, T) von G, die den beiden letzten Knoten in H entspricht.

vgl. deterministischer Algorithmus!

Ein einfacher **randomisierter** Algorithmus [Karger SODA'93]



Laufzeit

$\textsc{Contract}$(zsghd. Multigraph $G = (V, E)$)

$H \leftarrow G$
while H hat mehr als zwei Knoten **do**
  wähle Kante e in H zufällig und gleichverteilt
  $H \leftarrow H/e$
return Zerlegung (S, T) von G, die den beiden letzten Knoten in H entspricht.

- Anzahl der Knoten sinkt in jeder Iteration um 1.
- Jede Iteration benötigt $O(E)$ Zeit.

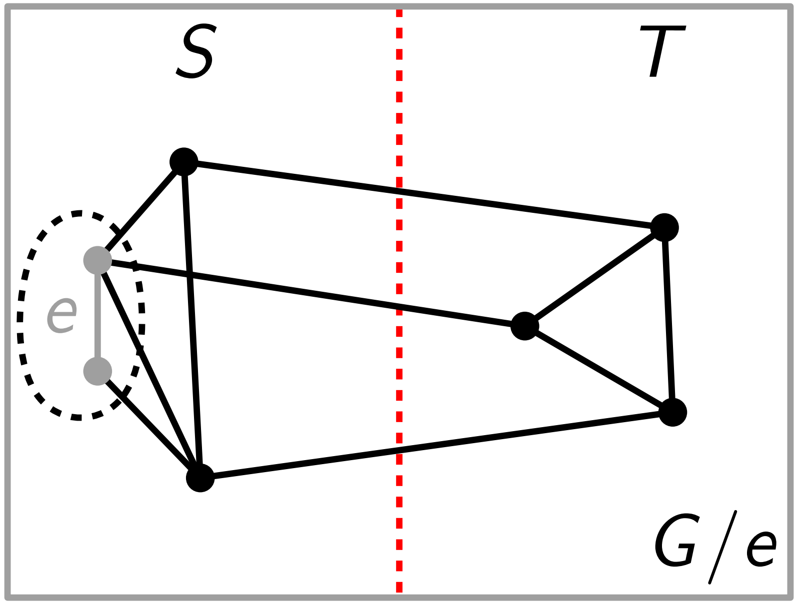
  Übg.: Implementierung mit $O(V)$ Zeit pro Iteration möglich!

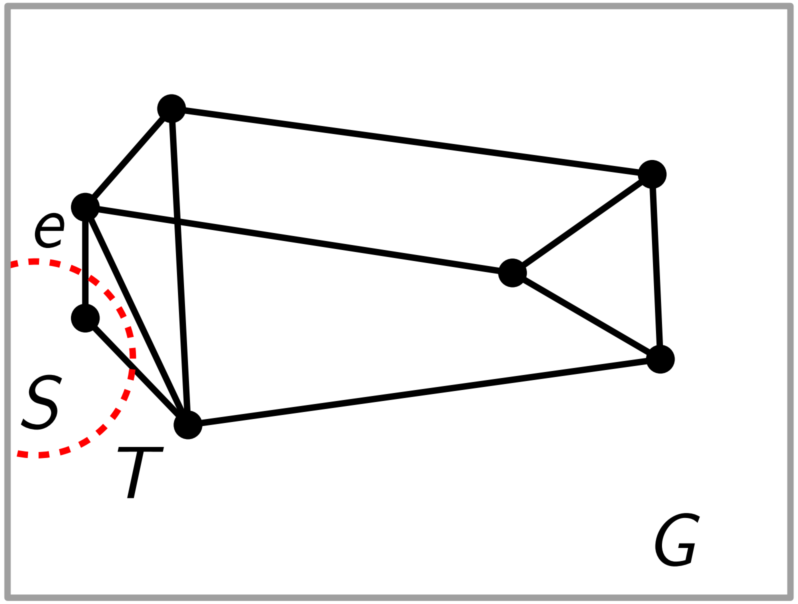
Gesamtlaufzeit: $O(V^2)$

Beobachtung Kantenkontraktion

Jeder Schnitt in G/e korrespondiert zu Schnitt gleicher Größe in G.

$\rightsquigarrow$ Größe eines kleinsten Schnitts steigt (schwach) monoton während Ausführung von $\textsc{Contract}$





Erfolgswahrscheinlichkeit – Einschätzung

Satz. Sei (S, T) ein kleinster Schnitt. Die Wahrscheinlichkeit, dass $\textsc{Contract}$ diesen Schnitt findet, ist $\geq \frac{2}{n(n-1)}$.

Diese Wahrscheinlichkeit ist **überraschend hoch**, da es $(2^n - 2)/2 = 2^{n-1} - 1$ viele Schnitte gibt.

Die Wahrscheinlichkeit, dass ein zufällig und gleichverteilt gewählter Schnitt genau (S, T) trifft, ist also nur $\frac{1}{2^{\Omega(n)}}$, also **exponentiell klein**.

Erfolgswahrscheinlichkeit – Beweis

Satz. Sei (S, T) ein kleinster Schnitt. Die Wahrscheinlichkeit, dass $\textsc{Contract}$ diesen Schnitt findet, ist $\geq \frac{2}{n(n-1)}$.

Beweis.

Sei (S, T) ein kleinster Schnitt, $C = \{\, uv \in E \mid u \in S, v \in T \,\}$ und $k = |C|$.

Graph H_i (H zu Beginn von Iteration i) hat $n - i + 1$ Knoten.

Kleinster Schnitt in H_i hat Größe $\geq k$.

$\Rightarrow$ Jeder Knoten u in H_i hat Grad $\geq k$.

sonst hätte Schnitt $(u, V - u)$ Größe $< k$

$\Rightarrow$ H_i hat $\geq k(n - i + 1)/2$ Kanten.

Erfolgswahrscheinlichkeit – Beweis

Satz. Sei (S, T) ein kleinster Schnitt. Die Wahrscheinlichkeit, dass $\textsc{Contract}$ diesen Schnitt findet, ist $\geq \frac{2}{n(n-1)}$.

Beweis. (Forts.)

H_i hat $\geq k(n-i+1)/2$ Kanten.

Sei $\mathcal{E}_i$ Ereignis, dass in Iteration i in H_i *keine* Kante aus C kontrahiert wird.

WK, dass in Iteration i keine Kante aus C kontrahiert wird (vorausgesetzt es wurde bislang noch keine solche kontrahiert):

Kardinalität von C Gesamtzahl der Kanten in H_i

$$\mathbf{Pr}\left[\mathcal{E}_i \,\middle|\, \bigcap_{j=1}^{i-1} \mathcal{E}_j\right] \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$$

Erfolgswahrscheinlichkeit – Beweis

Satz. Sei (S, T) ein kleinster Schnitt. Die Wahrscheinlichkeit, dass CONTRACT diesen Schnitt findet, ist $\geq \frac{2}{n(n-1)}$.

Beweis.

$$\mathbf{Pr}\left[\mathcal{E}_i \,\middle|\, \bigcap_{j=1}^{i-1} \mathcal{E}_j\right] \geq 1 - \frac{2}{n-i+1}$$

Wahrscheinlichkeit, dass CONTRACT *keine* Kante aus C kontrahiert:

$$\mathbf{Pr}\left[\bigcap\nolimits_{i=1}^{n-2} \mathcal{E}_i\right] \geq \prod_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1}$$

$$= \frac{\cancel{(n-2)} \cdot \cancel{(n-3)} \cdot \ldots \cdot \cancel{3} \cdot 2 \cdot 1}{n \cdot (n-1) \cdot \cancel{(n-2)} \cdot \ldots \cdot \cancel{3}} = \frac{2}{n(n-1)} \qquad \square$$

Verringerung der Fehlerwahrscheinlichkeit

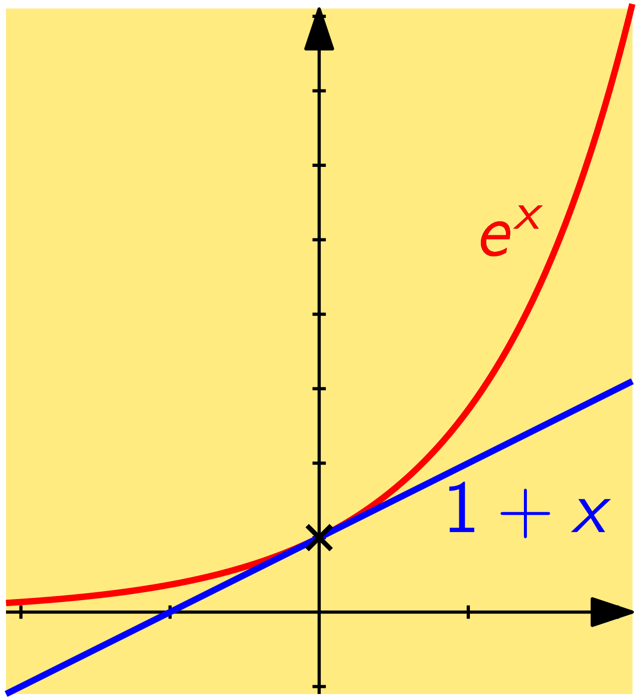
Erfolgswahrscheinlichkeit $\frac{2}{n(n-1)} \geq \frac{2}{n^2}$ zu klein für praktische Zwecke.

$\Rightarrow$ Wiederhole $n^2 \ln n$ mal und gib beste Lösung aus.

Fehlerwahrscheinlichkeit ist höchstens

$$\left(1 - \frac{2}{n^2}\right)^{n^2 \ln n} \leq e^{-\frac{2n^2 \ln n}{n^2}} = e^{-2 \ln n} = \frac{1}{n^2}$$

$1 + x \leq e^x$
für alle $x \in \mathbb{R}$



$\Rightarrow$ **Erfolgs**wahrscheinlichkeit $1 - \frac{1}{n^2}$ konvergiert schnell gegen 1.

Resümee

Es handelt sich um einen sog. **Monte-Carlo-Algorithmus**, da randomisiert und nicht immer korrekt.

Eine andere Art randomisierter Algorithmen sind sog. **Las-Vegas-Algorithmen**, bei denen das Ergebnis immer korrekt ist, aber die Laufzeit eine Zufallsvariable ist (Bsp. RandomizedQuickSort in der ADS).

Aber: Der Alg. ist mit **hoher Wahrscheinlichkeit** korrekt; Fehlerwahrscheinlichkeit konvergiert mit steigendem n gegen 0.

Gesamtlaufzeit $O(n^4 \log n)$ deutlich besser als $O(n^6)$ bei naivem deterministischen Ansatz.

Außerdem ist der randomisierte Algorithmus viel einfacher!

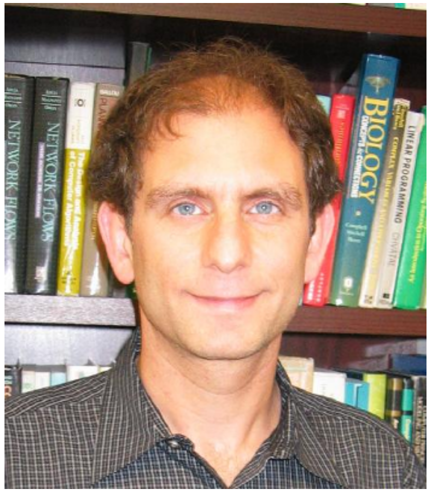
Satz. Es gibt einen randomisierten Algorithmus für MinCut mit Laufzeit $O(n^4 \log n)$ und Erfolgswahrscheinlichkeit $1 - 1/n^2$.

Es geht noch deutlich schneller!

Satz. Es gibt einen randomisierten Algorithmus für MinCut mit $O(n^2 \log^3 n)$ Laufzeit, der mit hoher Wahrscheinlichkeit einen kleinsten Schnitt ermittelt.

[Karger & Stein, STOC'93]

David R. Karger



Clifford Stein

Dies ist überraschend schnell, da ein Graph $\Theta(n^2)$ Kanten haben kann. In diesem Fall ist die Laufzeit beinahe linear!

Partielle Ausführung Contract

Erfolgswahrscheinlichkeit Contract

sinkt mit wachsendem i

$$\prod_{i=1}^{n-2} \boxed{\left(1 - \frac{2}{n-i+1}\right)} = \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdot \ldots \cdot \frac{1}{3}$$

Idee: Ändere Strategie mit wachsender Iteration i.

Contract(G, t): Stoppe, wenn Graph noch t Knoten enthält.

Schätze Wahrscheinlichkeit ab, dass Contract(G, t) keine Kante aus optimalem Schnitt C kontrahiert:

$$\mathbf{Pr}\left[\bigcap_{i=1}^{n-t} \mathcal{E}_i\right] \geq \prod_{i=1}^{n-t} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} \qquad (\star)$$

$$= \frac{\cancel{(n-2)} \cdot \cancel{(n-3)} \cdot \ldots \cdot t \cdot (t-1)}{n \cdot (n-1) \cdot \cancel{(n-2)} \cdot \ldots \cdot \cancel{(t+1)}} = \boxed{\frac{t(t-1)}{n(n-1)}}$$

Algorithmus FastCut

FastCut(zsghd. Multigraph $G = (V, E)$)

$n \leftarrow |V|$
if $n \leq 6$ **then**
  löse Problem „brute force“
else
  $t \leftarrow \lceil 1 + n/\sqrt{2} \rceil \approx 0,7 \cdot n$
  $G_1 \leftarrow \textsc{Contract}(G, t)$
  $G_2 \leftarrow \textsc{Contract}(G, t)$
  $(S_1, T_1) \leftarrow \textsc{FastCut}(G_1)$
  $(S_2, T_2) \leftarrow \textsc{FastCut}(G_2)$
  return den kleineren der Schnitte (S_1, T_1), (S_2, T_2)

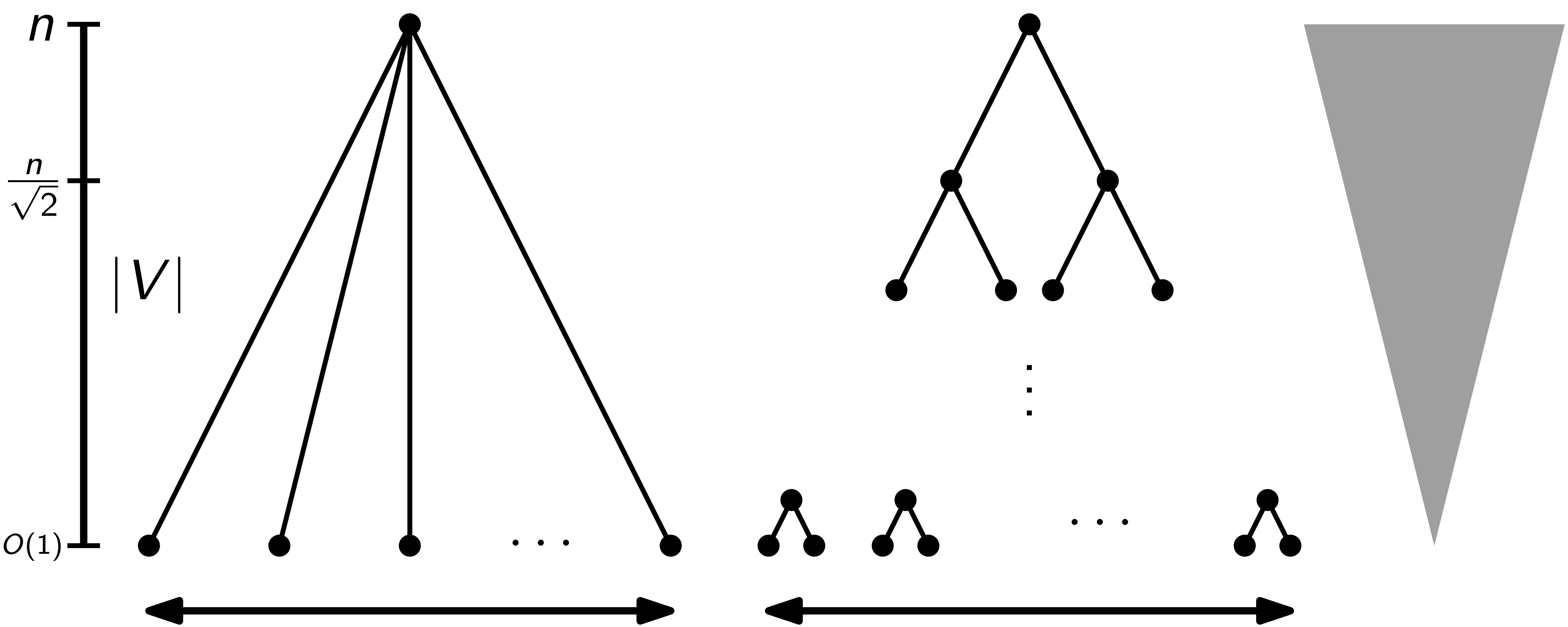
unabhängige Ausführungen

Laufzeit?

Intuition FastCut vs. Contract

$$\prod_{i=1}^{n-2}\left(1-\frac{2}{n-i+1}\right)$$ Erfolgswahrscheinlichkeit

Contract (mit Wh.) FastCut (einmal) Ersparnis



$O(n^2 \log n)$ Schnitte $O(n^2)$ Schnitte, da $2^{\log_{\sqrt{2}} n} = n^2$.

Laufzeit

Satz. FastCut läuft in $O(n^2 \log n)$ Zeit.

```
FastCut(zsghd. Multigraph G = (V, E))
  n ← |V|
  if n ≤ 6 then
  |   löse Problem „brute force“
  else
  |   t ← ⌈1 + n/√2⌉  ≈ 0,7 · n
  |   G1 ← Contract(G, t)
  |   G2 ← Contract(G, t)
  |   (S1, T1) ← FastCut(G1)
  |   (S2, T2) ← FastCut(G2)
  |   return kleineren von (S1, T1), (S2, T2)
```

Beweis. Rekurrenz

$$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + O(n^2)$$

Erinnerung Mastertheorem (2. Fall):

$$T(n) = aT(n/b) + f(n) \text{ mit } f(n) = \Theta(n^{\log_b a})$$

$$\Rightarrow T(n) = \Theta(f(n) \cdot \log n).$$

In unserem Fall: $a = 2$, $b = \sqrt{2}$, $f(n) = O(n^2)$.

Wegen $\log_{\sqrt{2}} 2 = 2$ folgt die Behauptung. $\square$

Erfolgswahrscheinlichkeit (I)

Satz. $\textsc{FastCut}$ liefert mit Wahrscheinlichkeit $\Omega(1/\log n)$ einen kleinsten Schnitt.

Beweis.

Betrachte Graph G mit n Knoten bei rekursivem Aufruf.

Seien G_1, G_2 die Graphen nach Anw. von $\textsc{Contract}(H, t)$.

Wahrscheinlichkeit, dass kleinster Schnitt aus G in G_1 „überlebt" ist nach $(\star)$ mindestens

$$\frac{t(t-1)}{n(n-1)} = \frac{\lceil 1 + n/\sqrt{2}\rceil(\lceil 1 + n/\sqrt{2}\rceil - 1)}{n(n-1)} \geq \frac{(n/\sqrt{2})^2}{n^2} = \frac{1}{2}.$$

Erfolgswahrscheinlichkeit (II)

Seien G_1, G_2 die Graphen nach $2\times$ $\textsc{Contract}(G, t)$.

WK, dass kleinster Schnitt in G_1 überlebt, ist $\geq 1/2$.

$\textsc{FastCut}$ findet kleinsten Schnitt in G, wenn ein solcher in G_1 *oder* G_2 überlebt *und* der rekursive Aufruf im entsprechenden Graphen G_i ebenfalls einen kleinsten Schnitt findet.

Sei $\tau = \Theta(\log n)$ die Tiefe der Rekursion von $\textsc{FastCut}(G)$.

$\Rightarrow$ untere Schranke $p(\tau)$ für die Erfolgs-WK auf Tiefe τ:

$$p(\tau+1) = 1 - \underbrace{\left(\overbrace{1 - \frac{1}{2}p(\tau)}^{\text{Fehler-WK } \textsc{FastCut}(G_1)}\right)^2}_{\text{Fehler-WK auf Tiefe } \tau+1} = p(\tau) - \frac{p(\tau)^2}{4}, \text{ wobei } p(0) = 1.$$

Erfolgswahrscheinlichkeit (III)

Sei $p(\tau)$ untere Schranke für die Erfolgs-WK:

$$p(\tau+1) = p(\tau) - \frac{p(\tau)^2}{4} \quad \text{mit } p(0) = 1$$

Setze $q(\tau) = 4/p(\tau) - 1$. (Äquivalent: $p(\tau) = 4/(q(\tau)+1)$)
Einsetzen ergibt $q(0) = 3$ und

$$\begin{aligned} q(\tau+1) &= \frac{4}{p(\tau+1)} - 1 = \frac{4}{p(\tau) - p(\tau)^2/4} - 1 \\ &= \frac{1}{1/(q(\tau)+1) - 1/(q(\tau)+1)^2} - 1 \\ &= \frac{(q(\tau)+1)^2 - q(\tau)}{q(\tau)} = q(\tau) + 1 + \frac{1}{q(\tau)} \end{aligned}$$

Erfolgswahrscheinlichkeit (IV)

Es gilt $q(\tau+1) = q(\tau) + 1 + \frac{1}{q(\tau)}$ mit $q(0) = 3$.

Zeige per Induktion, dass $\tau < q(\tau) < \tau + h_{\tau-1} + c$ für $\tau \geq 2$ und genügend große Konstante $c > 0$.

i-te harmonische Zahl
$h_i := \sum_{j=1}^{i} \frac{1}{j} = \Theta(\log i)$

Induktionsanfang ($\tau = 2$)*:*
$2 < 3 = q(0) \leq q(2) \leq c$

Induktionsschritt ($\tau \to \tau + 1$)*:*

Bem. Wenn wir unten nur z.B. $q(\tau) \leq 27\tau$ brauchen – warum zeigen wir oben viel mehr?

$$q(\tau+1) \geq q(\tau) + 1 \overset{\text{IV}}{>} \tau + 1$$

$$q(\tau+1) \underset{\text{s.o.}}{=} q(\tau) + 1 + \frac{1}{q(\tau)} \overset{\text{IV}}{<} (\tau + h_{\tau-1} + c) + 1 + \frac{1}{\tau} = (\tau+1) + h_\tau + c$$

$$\Rightarrow p(\tau) = \frac{4}{q(\tau)+1} = \Omega\left(\frac{1}{\log n}\right) \qquad \square$$

$\tau = \Theta(\log n)$ und $q(\tau) \in O(\tau)$

Verringerung der Fehlerwahrscheinlichkeit

Wiederhole FastCut $\log^2 n$ mal.

Die Erfolgswahrscheinlichkeit pro Ausführung ist $\geq C/\log n$ für ein $C > 0$.

Gesamtfehlerwahrscheinlichkeit:

$$\left(1 - \frac{C}{\log n}\right)^{\log^2 n} \leq e^{-C \log n} = \frac{1}{n^C}$$

$1 + x \leq e^x$
für alle $x \in \mathbb{R}$

Satz. Es gibt einen randomisierten Algorithmus für MinCut mit $O(n^2 \log^3 n)$ Laufzeit, der mit hoher Wahrscheinlichkeit einen kleinsten Schnitt ermittelt.

Und noch schneller!

Satz. Es gibt einen randomisierten Algorithmus für MinCut mit $O(m \log^3 n)$ Laufzeit, der mit hoher Wahrscheinlichkeit einen kleinsten Schnitt ermittelt.

Beweis. [Karger, STOC'96] □

Diese Laufzeit ist auch für Graphen mit $m = o(n^2)$ Kanten fast linear!

Zum Vergleich:

Satz. Es gibt einen einfachen deterministischen Algorithmus für MinCut mit $O(mn + n^2 \log n)$ Laufzeit.
[Stoer & Wagner, JACM'97]

…ist in LEDA (C++) und jgraphT (Java) implementiert.