Tópicos de Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

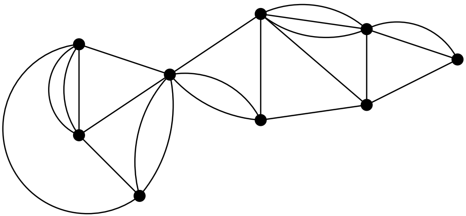
AULA 8

Corte mínimo em grafos

KT Secs 13.2

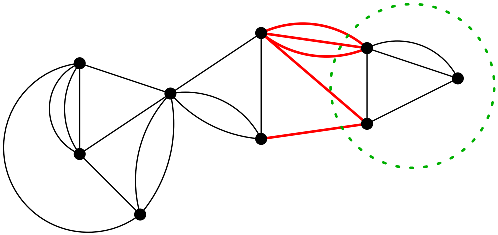
Cortes em grafos

G: grafo (não orientado) possivelmente com arestas paralelas.



Cortes em grafos

G: grafo (não orientado) possivelmente com arestas paralelas.



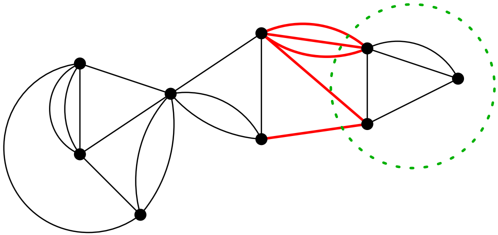
Conjunto C de arestas é um corte
se existe um conjunto $S \subseteq V_G$ tal que

$$C = \{e \in E_G : e \text{ tem uma ponta em } S \text{ e outra fora de } S\}.$$

Ou seja, $C = \delta_G(S) = \delta_G(V_G \setminus S)$.

Cortes em grafos

G: grafo (não orientado) possivelmente com arestas paralelas.



Conjunto C de arestas é um corte
se existe um conjunto $S \subseteq V_G$ tal que

$$C = \{e \in E_G : e \text{ tem uma ponta em } S \text{ e outra fora de } S\}.$$

Ou seja, $C = \delta_G(S) = \delta_G(V_G \setminus S)$.

Corte é não trivial se $S \neq \emptyset$ e $S \neq V_G$.

Corte mínimo

G: grafo possivelmente com arestas paralelas.

$C \subseteq E_G$ é um corte não trivial se existe $S \subsetneq V_G$ não vazio tal que

$$C = \{e \in E_G : e \text{ tem uma ponta em } S \text{ e outra fora de } S\}$$

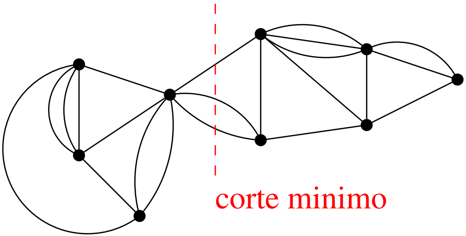
Corte mínimo

G: grafo possivelmente com arestas paralelas.

$C \subseteq E_G$ é um corte não trivial se existe $S \subsetneq V_G$ não vazio tal que

$$C = \{e \in E_G : e \text{ tem uma ponta em } S \text{ e outra fora de } S\}$$

Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.



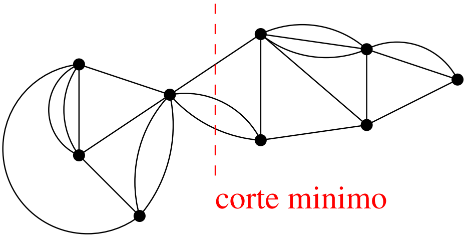
Corte mínimo

G: grafo possivelmente com arestas paralelas.

$C \subseteq E_G$ é um corte não trivial se existe $S \subsetneq V_G$ não vazio tal que

$$C = \{e \in E_G : e \text{ tem uma ponta em } S \text{ e outra fora de } S\}$$

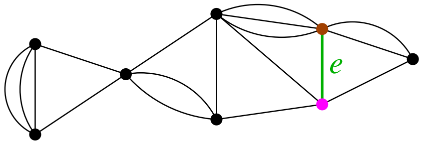
Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.



Esta aula: algoritmo aleatorizado que devolve um corte mínimo (não trivial) com alta probabilidade.

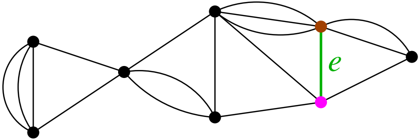
Contração de arestas

Dado um grafo G e uma aresta $e = uv$,
a contração $G|e$ é o grafo (V', E') onde...

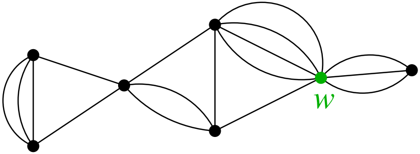


Contração de arestas

Dado um grafo G e uma aresta $e = uv$,
a contração $G|e$ é o grafo (V', E') onde...

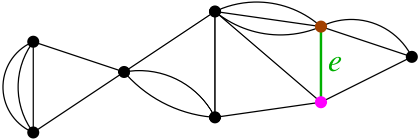


$V' = V \setminus \{u, v\} \cup \{w\}$ e $E' = E_G \setminus \{f : \mathrm{pontas}(f) = \{u, v\}\}$
e cada aresta f em E' tem as mesmas pontas que f em E_G
exceto por u e v, que são substituídos por w.

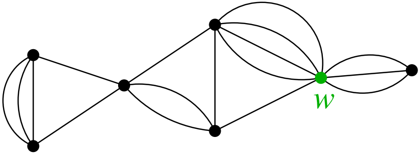


Contração de arestas

Dado um grafo G e uma aresta $e = uv$,
a contração $G|e$ é o grafo (V', E') onde...



$V' = V \setminus \{u, v\} \cup \{w\}$ e $E' = E_G \setminus \{f : \mathrm{pontas}(f) = \{u, v\}\}$
e cada aresta f em E' tem as mesmas pontas que f em E_G
exceto por u e v, que são substituídos por w.



Proposição: Se C é um corte em $G|e$, então C é um corte em G.

Algoritmo de Karger

Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.

Algoritmo de Karger

Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.

KARGER (G)
1 **enquanto** $|V_G| > 2$ **faça**
2 $\quad e \leftarrow$ RANDOM-E(E_G)
3 $\quad G \leftarrow G|e$
4 **devolva** E_G

Algoritmo de Karger

Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.

KARGER (G)
1 **enquanto** $|V_G| > 2$ **faça**
2 $\quad e \leftarrow$ RANDOM-E(E_G)
3 $\quad G \leftarrow G|e$
4 **devolva** E_G

Na linha 2, note que E_G é sempre não vazio pois G é conexo.

Pela proposição, o algoritmo devolve um corte de G,
e esse corte é não trivial.

Algoritmo de Karger

Problema: Dado um grafo conexo G com pelo menos dois vértices, encontrar um corte C não trivial com $|C|$ mínimo.

```
KARGER (G)
1    enquanto |V_G| > 2 faça
2        e ← RANDOM-E(E_G)
3        G ← G|e
4    devolva E_G
```

Na linha 2, note que E_G é sempre não vazio pois G é conexo.

Pela proposição, o algoritmo devolve um corte de G,
e esse corte é não trivial.

Claramente o algoritmo consume tempo polinomial.
(O número de iteração é $|V_G| - 2$.)

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

Vamos mostrar que

$$\Pr\{C_K \text{ não é um corte não trivial mínimo}\} < q(n) < 1,$$

onde $n = |V_G|$.

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

Vamos mostrar que

$$\Pr\{C_K \text{ não é um corte não trivial mínimo}\} < q(n) < 1,$$

onde $n = |V_G|$.

Executa-se o algoritmo r vezes
e devolve-se o menor corte produzido.

A chance do corte não trivial devolvido não ser mínimo será $< q(n)^r$.

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

O corte C não é a resposta do algoritmo sse alguma aresta de C foi contraída.

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

O corte C não é a resposta do algoritmo sse
alguma aresta de C foi contraída.

Seja E_i o evento:

uma aresta de C não é sorteada na iteração i.

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

O corte C não é a resposta do algoritmo sse alguma aresta de C foi contraída.

Seja E_i o evento:

uma aresta de C não é sorteada na iteração i.

Vamos delimitar inferiormente $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

O corte C não é a resposta do algoritmo sse alguma aresta de C foi contraída.

Seja E_i o evento:

uma aresta de C não é sorteada na iteração i.

Vamos delimitar inferiormente $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Tal probabilidade é igual a

$$\Pr\{E_1\} \cdot \Pr\{E_2 | E_1\} \cdots \Pr\{E_{n-2} | E_1 \cap E_2 \cap \cdots \cap E_{n-3}\}.$$

Análise do algoritmo

Seja C um corte não trivial mínimo de G.

O corte C não é a resposta do algoritmo sse
alguma aresta de C foi contraída.

Seja E_i o evento:

uma aresta de C não é sorteada na iteração i.

Vamos delimitar inferiormente $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Tal probabilidade é igual a

$$\Pr\{E_1\} \cdot \Pr\{E_2 | E_1\} \cdots \Pr\{E_{n-2} | E_1 \cap E_2 \cap \cdots \cap E_{n-3}\}.$$

Para começar, como delimitar $\Pr\{E_1\}$?

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Como delimitar $\Pr\{E_1\}$?

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Como delimitar $\Pr\{E_1\}$?

Como C é mínimo, $\delta_G(v) \geq k$ para todo vértice v.

Logo $|E_G| \geq kn/2$.

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Como delimitar $\Pr\{E_1\}$?

Como C é mínimo, $\delta_G(v) \geq k$ para todo vértice v.

Logo $|E_G| \geq kn/2$.

Portanto $\Pr\{\bar{E}_1\} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$.

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Como delimitar $\Pr\{E_1\}$?

Como C é mínimo, $\delta_G(v) \geq k$ para todo vértice v.

Logo $|E_G| \geq kn/2$.

Portanto $\Pr\{\bar{E}_1\} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$.

Ou seja, $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Como delimitar $\Pr\{E_1\}$?

Como C é mínimo, $\delta_G(v) \geq k$ para todo vértice v.

Logo $|E_G| \geq kn/2$.

Portanto $\Pr\{\bar{E}_1\} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$.

Ou seja, $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Como delimitar $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\}$?

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Vimos que $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Como delimitar $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\}$?

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Vimos que $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Como delimitar $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\}$?

No começo da iteração $i+1$, temos que $|E_G| \geq k(n-i)/2$.

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Vimos que $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Como delimitar $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\}$?

No começo da iteração $i + 1$, temos que $|E_G| \geq k(n - i)/2$.

Logo $\Pr\{\bar{E}_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \leq \frac{k}{\frac{k(n-i)}{2}} = \frac{2}{n-i}$.

Análise do algoritmo

C: um corte não trivial mínimo de G e $k = |C|$.

E_i: uma aresta de C não é sorteada na iteração i.

Vimos que $\Pr\{E_1\} \geq 1 - \frac{2}{n}$.

Como delimitar $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\}$?

No começo da iteração $i + 1$, temos que $|E_G| \geq k(n-i)/2$.

Logo $\Pr\{\bar{E}_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \leq \frac{k}{\frac{k(n-i)}{2}} = \frac{2}{n-i}$.

Ou seja, $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Ou seja, delimitar $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Já sabemos que $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Ou seja, delimitar $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Já sabemos que $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Então

$$\Pr\{C_K = C\} \quad \geq \quad \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right)$$

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Ou seja, delimitar $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Já sabemos que $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Então

$$
\begin{aligned}
\Pr\{C_K = C\} &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) \\
&= \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right)
\end{aligned}
$$

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Ou seja, delimitar $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Já sabemos que $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Então

$$
\begin{aligned}
\Pr\{C_K = C\} &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) \\
&= \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3}
\end{aligned}
$$

Análise do algoritmo

Seja C_K o corte devolvido por KARGER (G).

C: um corte não trivial mínimo de G.

Queremos delimitar inferiormente $\Pr\{C_K = C\}$.

Ou seja, delimitar $\Pr\{E_1 \cap E_2 \cap \cdots \cap E_{n-2}\}$.

Já sabemos que $\Pr\{E_{i+1} | E_1 \cap E_2 \cap \cdots \cap E_i\} \geq 1 - \frac{2}{n-i}$.

Então

$$
\begin{aligned}
\Pr\{C_K = C\} &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) \\
&= \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)}.
\end{aligned}
$$

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\}$

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\} \;\leq\; q(n)$.

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\} \;\leq\; q(n)$.

Se executarmos o algoritmo $r = c\frac{n(n-1)}{2} \ln n$ vezes, então

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\} \;\leq\; q(n)$.

Se executarmos o algoritmo $r = c\frac{n(n-1)}{2} \ln n$ vezes, então

$$
\begin{aligned}
\Pr\{C_K^* \text{ não é um corte não trivial mínimo}\} &\leq q(n)^r \\
&= \left(1 - \frac{2}{n(n-1)}\right)^{c\frac{n(n-1)}{2}\ln n} \\
&< \left(\frac{1}{e}\right)^{c \ln n} \;=\; \frac{1}{n^c}.
\end{aligned}
$$

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\} \;\leq\; q(n)$.

Se executarmos o algoritmo $r = c\frac{n(n-1)}{2} \ln n$ vezes, então

$$
\begin{aligned}
\Pr\{C_K^* \text{ não é um corte não trivial mínimo}\} &\leq q(n)^r \\
&= \left(1 - \frac{2}{n(n-1)}\right)^{c\frac{n(n-1)}{2}\ln n} \\
&< \left(\frac{1}{e}\right)^{c\ln n} = \frac{1}{n^c}.
\end{aligned}
$$

Erra com probabilidade polinomialmente pequena.

Conclusão

Logo $\Pr\{C_K \neq C\} \;\leq\; 1 - \frac{2}{n(n-1)} \;=\; q(n)$
e $\Pr\{C_K$ não é um corte não trivial mínimo$\} \;\leq\; q(n)$.

Se executarmos o algoritmo $r = c\frac{n(n-1)}{2} \ln n$ vezes, então

$$\begin{aligned}
\Pr\{C_K^* \text{ não é um corte não trivial mínimo}\} \;&\leq\; q(n)^r \\
&= \left(1 - \frac{2}{n(n-1)}\right)^{c\frac{n(n-1)}{2}\ln n} \\
&< \left(\frac{1}{e}\right)^{c \ln n} \;=\; \frac{1}{n^c}.
\end{aligned}$$

Erra com probabilidade polinomialmente pequena.

Como r é polinomial em n e em c, o algoritmo é polinomial.

Uma pergunta...

G: grafo conexo com n vértices

Quantos cortes não triviais mínimos diferentes podem existir?

Uma pergunta...

G: grafo conexo com n vértices

Quantos cortes não triviais mínimos diferentes podem existir?

Como $\Pr\{C_K = C\} \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$,

Uma pergunta...

G: grafo conexo com n vértices

Quantos cortes não triviais mínimos diferentes podem existir?

Como $\Pr\{C_K = C\} \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$,

há no máximo $\binom{n}{2}$ cortes mínimos em G.

Uma pergunta...

G: grafo conexo com n vértices

Quantos cortes não triviais mínimos diferentes podem existir?

Como $\Pr\{C_K = C\} \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$,

há no máximo $\binom{n}{2}$ cortes mínimos em G.

Se G for por exemplo um circuito com n vértices...

Uma pergunta...

G: grafo conexo com n vértices

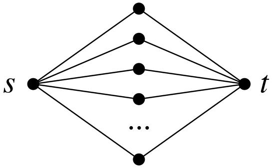
Quantos cortes não triviais mínimos diferentes podem existir?

Como $\Pr\{C_K = C\} \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$,

há no máximo $\binom{n}{2}$ cortes mínimos em G.

Se G for por exemplo um circuito com n vértices...

Quantos st-cortes mínimos diferentes podem existir em G?



Uma pergunta...

G: grafo conexo com n vértices

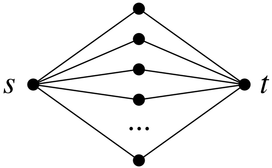
Quantos cortes não triviais mínimos diferentes podem existir?

Como $\Pr\{C_K = C\} \geq \frac{2}{n(n-1)} = 1/\binom{n}{2}$,

há no máximo $\binom{n}{2}$ cortes mínimos em G.

Se G for por exemplo um circuito com n vértices...

Quantos st-cortes mínimos diferentes podem existir em G?



Um número exponencial em n...

Consumo de tempo

```
KARGER (G)
    enquanto |V_G| > 2 faça
        e ← RANDOM-E(E_G)
        G ← G|e
    devolva E_G
```

Não é difícil pensar em uma implementação que consome tempo $\mathrm{O}(|E_G| \lg |E_G|)$ e espaço extra $\mathrm{O}(|V_G|)$.

(Mantenha o grau corrente dos vértices. Junte sempre a lista mais curta à mais longa. Cada célula poderá mudar para outra lista no máximo $\lg |E_G|$ vezes, e o tempo é proporcional a estas mudanças.)

Consumo de tempo

KARGER (G)

1 **enquanto** $|V_G| > 2$ **faça**

2 $\quad e \leftarrow$ RANDOM-E(E_G)

3 $\quad G \leftarrow G|e$

4 **devolva** E_G

Não é difícil pensar em uma implementação que consome tempo $\mathrm{O}(|E_G| \lg |E_G|)$ e espaço extra $\mathrm{O}(|V_G|)$.

(Mantenha o grau corrente dos vértices. Junte sempre a lista mais curta à mais longa. Cada célula poderá mudar para outra lista no máximo $\lg |E_G|$ vezes, e o tempo é proporcional a estas mudanças.)

Existe uma implementação que consome tempo e espaço $\mathrm{O}(|E_G|)$.

Como escolher uma aresta com probabilidade uniforme?

```
KARGER (G)
    enquanto |V_G| > 2 faça
        e ← RANDOM-E(E_G)
        G ← G|e
    devolva E_G
```

Como escolher uma aresta com probabilidade uniforme?

KARGER (G)
enquanto $|V_G| > 2$ **faça**
    $e \leftarrow$ RANDOM-E(E_G)
    $G \leftarrow G|e$
devolva E_G

Gere uma permutação aleatória das arestas no início do processo
e vá pegando a próxima aresta (que não virou laço) a cada iteração.

Como escolher uma aresta com probabilidade uniforme?

KARGER (G)

1 **enquanto** $|V_G| > 2$ **faça**
2 $\quad e \leftarrow$ RANDOM-E(E_G)
3 $\quad G \leftarrow G|e$
4 **devolva** E_G

Gere uma permutação aleatória das arestas no início do processo e vá pegando a próxima aresta (que não virou laço) a cada iteração.

Queremos determinar o prefixo mais curto dessa permutação cujas arestas induzem exatamente um grafo com duas componentes.

Pode-se fazer uma busca binária pelo comprimento ℓ desse prefixo: determine a cada iteração o número de componentes que o prefixo corrente induz. Se induz mais de duas, então ℓ está à direita; se induz uma, então ℓ está à esquerda.

Isso leva a uma outra implementação $\mathrm{O}(|E_G| \lg |E_G|)$.

Implementação linear

Gere uma permutação aleatória das arestas do grafo.

Seja $m = |E_G|$.

Implementação linear

Gere uma permutação aleatória das arestas do grafo.

Seja $m = |E_G|$.

Calcule quantas componentes as primeiras $m/2$ arestas induzem.

Se elas induzem uma componente,
jogue as demais $m/2$ arestas fora em tempo $\mathrm{O}(m)$.

Se elas induzem mais que duas componentes,
contraia todas elas simultaneamente em tempo $\mathrm{O}(m)$.

Repita o processo no grafo resultante que tem no máximo $m/2$ arestas.

Consumo de tempo de uma execução: $m + \frac{m}{2} + \frac{m}{4} + \cdots = \mathrm{O}(m)$.

Das $r = c\frac{n(n-1)}{2} \ln n$ execuções do algoritmo: $\mathrm{O}(n^2 m \lg n)$.

Aprimoramento de Karger e Stein

Considere a versão do algoritmo de Karger que para quando $|V_G| = t$.

Chame-a de KARGER(G, t) e seja $n = |V_G|$ no momento da chamada.

Aprimoramento de Karger e Stein

Considere a versão do algoritmo de Karger que para quando $|V_G| = t$.

Chame-a de KARGER(G, t) e seja $n = |V_G|$ no momento da chamada.

Prove que a chance de nenhuma aresta de um corte mínimo C de G ter sido contraída numa chamada de KARGER(G, t) é pelo menos

$$\frac{\binom{t}{2}}{\binom{n}{2}}.$$

Note que esse valor se torna menor que $1/2$ quando t é cerca de $n/\sqrt{2}$.

Aprimoramento de Karger e Stein

Considere a versão do algoritmo de Karger que para quando $|V_G| = t$.

Chame-a de KARGER(G, t) e seja $n = |V_G|$ no momento da chamada.

Prove que a chance de nenhuma aresta de um corte mínimo C de G ter sido contraída numa chamada de KARGER(G, t) é pelo menos

$$\frac{\binom{t}{2}}{\binom{n}{2}}.$$

Note que esse valor se torna menor que $1/2$ quando t é cerca de $n/\sqrt{2}$.

Karger e Stein usaram essa observação para projetar uma versão do algoritmo de Karger mais rápida.

Aprimoramento de Karger e Stein

FASTMINCUT (G)

se $|V_G| \leq 6$

  devolva MINCUT(G)

$t \leftarrow \lceil 1 + |V_G|/\sqrt{2} \rceil$

$G_1 \leftarrow$ KARGER(G, t)

$G_2 \leftarrow$ KARGER(G, t)

return $\min\{$FASTMINCUT(G_1), FASTMINCUT$(G_2)\}$

Aprimoramento de Karger e Stein

FASTMINCUT (G)

se $|V_G| \leq 6$

  devolva MINCUT(G)

$t \leftarrow \lceil 1 + |V_G|/\sqrt{2} \rceil$

$G_1 \leftarrow$ KARGER(G, t)

$G_2 \leftarrow$ KARGER(G, t)

return min{FASTMINCUT(G_1), FASTMINCUT(G_2)}

O consumo de tempo deste algoritmo é $\mathrm{O}(n^2 \lg n)$.

Aprimoramento de Karger e Stein

FASTMINCUT (G)

se $|V_G| \leq 6$
    devolva MINCUT(G)
$t \leftarrow \lceil 1 + |V_G|/\sqrt{2} \rceil$
$G_1 \leftarrow$ KARGER(G, t)
$G_2 \leftarrow$ KARGER(G, t)
return $\min\{$FASTMINCUT(G_1), FASTMINCUT$(G_2)\}$

O consumo de tempo deste algoritmo é $\mathrm{O}(n^2 \lg n)$.

A probabilidade dele devolver um corte mínimo é $P(n) = \Omega(\frac{1}{\lg n})$.

Aprimoramento de Karger e Stein

FASTMINCUT (G)
    se $|V_G| \leq 6$
        devolva MINCUT(G)
    $t \leftarrow \lceil 1 + |V_G|/\sqrt{2} \rceil$
    $G_1 \leftarrow$ KARGER(G, t)
    $G_2 \leftarrow$ KARGER(G, t)
    return $\min\{$FASTMINCUT(G_1), FASTMINCUT$(G_2)\}$

O consumo de tempo deste algoritmo é $\mathrm{O}(n^2 \lg n)$.

A probabilidade dele devolver um corte mínimo é $P(n) = \Omega(\frac{1}{\lg n})$.

Para tornar a probabilidade de erro $\mathrm{O}(1/n)$,
pode-se rodar este algoritmo $\mathrm{O}(\lg n / P(n))$ vezes,
o que resulta em um consumo de tempo $\mathrm{O}(n^2 \lg^3 n)$.

Há um pouco mais de detalhes na página Karger's algorithm da wiki.