CSE P521: Applied Algorithms

Instructor: Prof. James R. Lee

TAs: Evan McCarty (head), Jeffrey Hon

Office hours: TBA

Class e-mail list: Sign up at course site if you didn't receive "hello" email

Discussion board: Accessible from course homepage; intended for "unsupervised" discussion of course topics

Grading: 5-6 Homeworks (60%), Project (40%)

Homework will be assigned on **Thursdays**; due next Thursday

There will be a **homework out tomorrow**.

Collaboration policy on website; all homework submitted electronically

[Prefer typeset solutions; scans of neat handwriting acceptable]

Project will be described in 3rd lecture; must work in pairs

Instructor: Prof. James R. Lee

TAs: Evan McCarty (head), Jeffrey Hon

Office hours: TBA

Expected background:
- discrete math (CSE 311)
- basic probability theory (CSE 312)
- undergrad algs & data structures (CSE 332)
- “mathematical maturity” [this is a theory course]

Course materials:
- There is no textbook
- Lecture notes and supplementary reading posted on course site
- Some lectures will have **required** preparatory reading [will send email]

Questions?

Modern algorithms

Approximation, randomization

Inputs are huge, noisy, dynamic, incomplete, high-dimensional, arrive online

Nuanced tradeoffs: Efficiency, profit, correctness

Tools of algorithmic analysis

Course cannot be comprehensive

Goal is exposure to a sample of key ideas, techniques, philosophies

Mathematical explanations

Prove that things work when we can

Develop a theoretical framework for understanding/designing solutions

Hashing

Universal and perfect hashing

Load balancing, the power of two choices

Streaming algorithms

Locality sensitive hashing, high-dimensional search

Spectral algorithms

Singular-value decomposition (SVD)

Principal component analysis

Spectral partitioning

Linear programming

Formulating LPs; relaxations and approximation

Duality theory

Gradient descent

Online optimization

Regret minimization

Boosting, multiplicative weights

Algorithmic game theory

Algorithms in the face of economic incentives

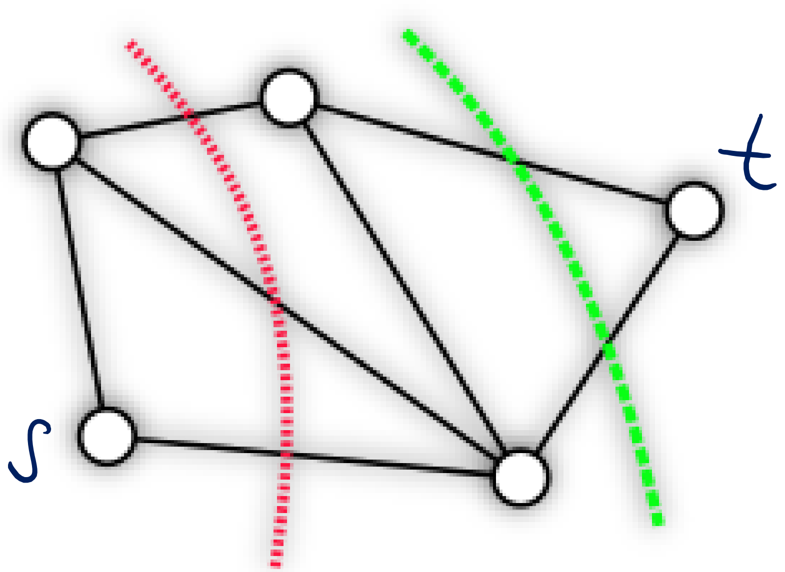
Exploiting selfish agents

The Global Min-cut problem

non-empty

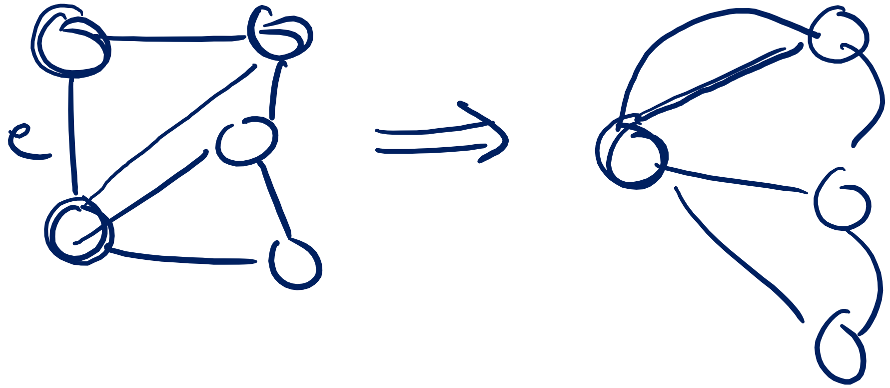
Input: An undirected graph $G = (V, E)$

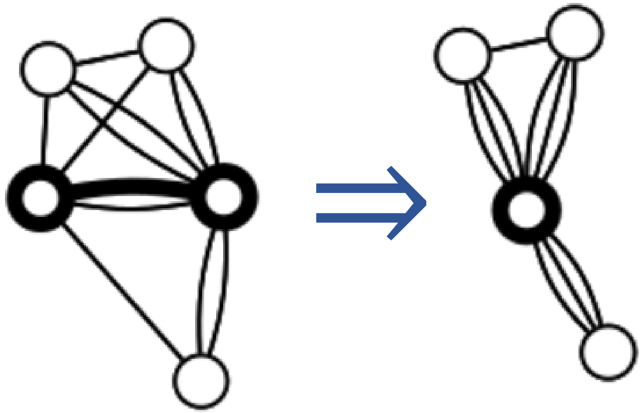
Output: A partition of the graph into two pieces $V = S \cup \bar{S}$ so the number of cut edges is minimized



Contraction operation:

For an edge $e \in E$, write G/e for the new graph formed by contracting the edge e.

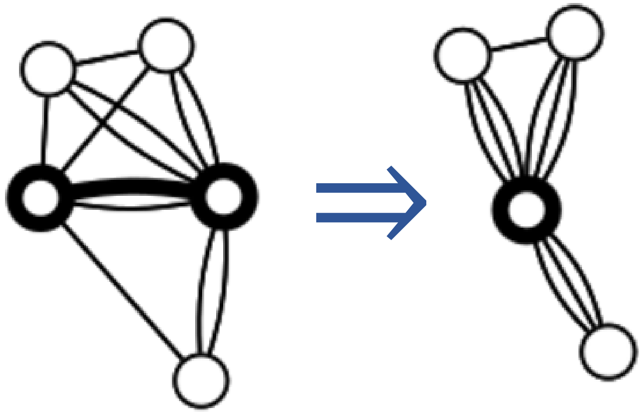




```
procedure contract(G = (V, E)):
while |V| > 2
    choose e ∈ E uniformly at random
    G ← G/e
return the only cut in G
```

Contraction operation:

For an edge $e \in E$, write G/e for the new graph formed by contracting the edge e.

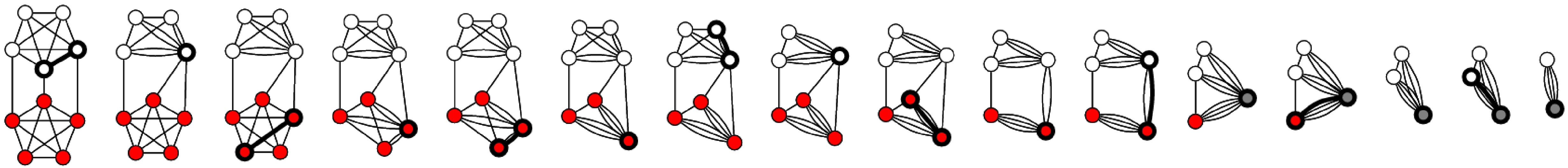


```
procedure contract(G = (V, E)):
while |V| > 2
    choose e ∈ E uniformly at random
    G ← G/e
return the only cut in G
```

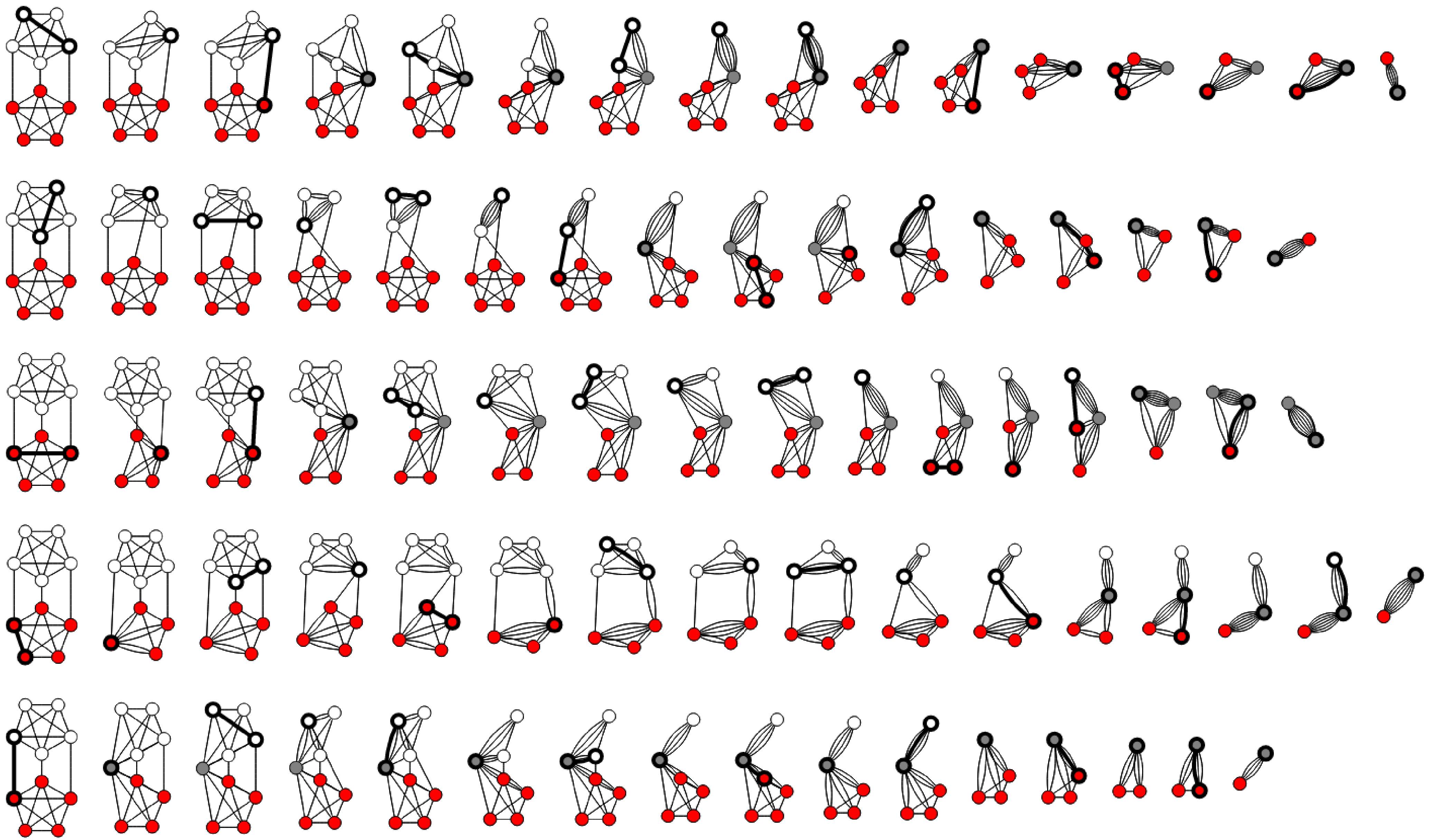
How many times does the while loop execute?

$|V| - 2$ times

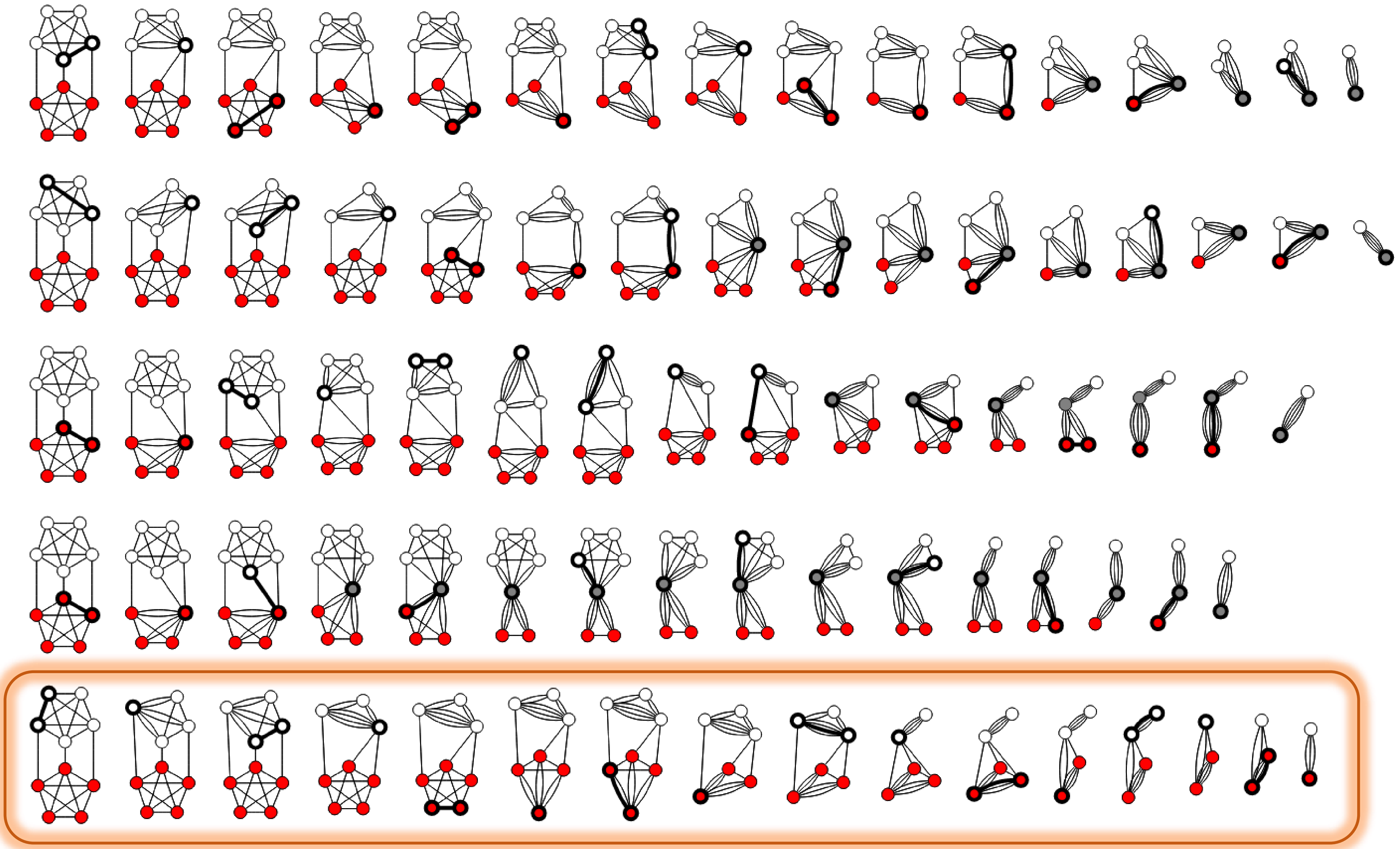
```
procedure contract(G = (V, E)):
while |V| > 2
    choose e ∈ E uniformly at random
    G ← G/e
return the only cut in G
```



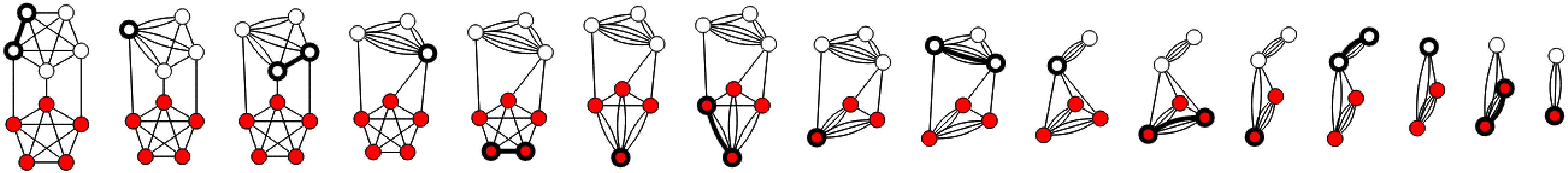
Karger's randomized min cuts



Karger's randomized min cuts



$G = (V, E) \qquad n = |V|$



Fix a min-cut $(S, \bar{S})$. Let $L = \#$ edges across $(S, \bar{S})$

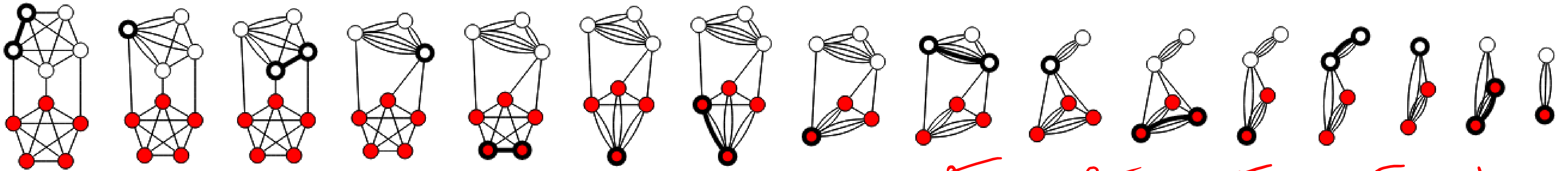
A_i = event that we contract an $(S, \bar{S})$ edge in step i

$$|E| \geq \frac{L \cdot |V|}{2} \geq \frac{n \cdot L}{2}$$

$$\mathbb{P}[A_1] = \frac{L}{|E|} \leq \frac{L}{\frac{n \cdot L}{2}} = \frac{2}{n}.$$

$$\mathbb{P}\left(\bigcap_{i=1}^{n-2} (\neg A_i)\right)$$

$$\mathbb{P}[\neg A_1] \geq 1 - \frac{2}{n}, \quad \mathbb{P}[A_2 \mid \neg A_1] = ?$$



$$\frac{\cancel{n-2}}{n}\cdot\frac{\cancel{n-3}}{n-1}\cdot\frac{\cancel{n-4}}{\cancel{n-2}}\cdot\frac{n-5}{\cancel{n-3}}\cdots\frac{1}{3}\;\boxed{=\frac{2}{n(n-1)}}$$

$$\mathbb{P}[\neg A_1 \wedge \neg A_2 \wedge \dots \wedge \neg A_{n-2}]$$

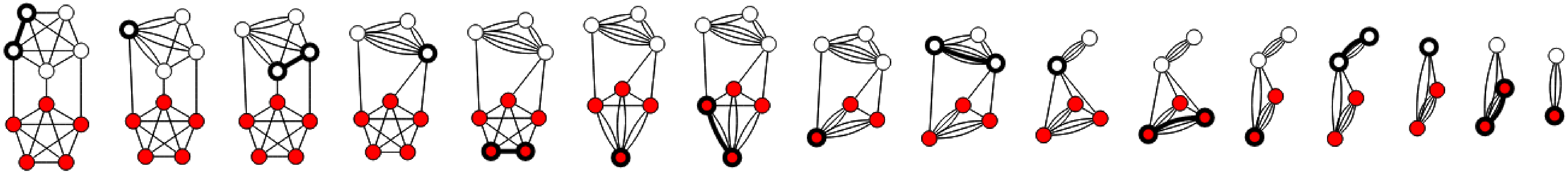
$$= \mathbb{P}[\neg A_1]\cdot \mathbb{P}[\neg A_2 \mid \neg A_1]\cdot \mathbb{P}[\neg A_3 \mid \neg A_1, \neg A_2]$$

$$\cdots\ \mathbb{P}[\neg A_{n-2} \mid \neg A_1, \dots, \neg A_{n-1}]$$

$$\mathbb{P}[\neg A_1] \geq 1 - \frac{2}{n}$$

$$\mathbb{P}[A_k \mid \neg A_1 \wedge \dots \wedge \neg A_{k-1}] = \frac{L}{\#\text{edges left in phase } k} \leq \frac{L}{\frac{L(n-k+1)}{2}} = \frac{2}{n-k+1}$$

$$\Rightarrow \mathbb{P}[\neg A_k \mid \neg A_1, \dots, \neg A_{k-1}] \geq 1 - \frac{2}{n-k+1} = \frac{n-k-1}{n-k+1}$$



Theorem: For any min-cut $(S, \bar{S})$, Karger's algorithm returns $(S, \bar{S})$ with probability at least

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

Corollary: Any graph has at most $\binom{n}{2}$ global min-cuts.

$1 - x \leq e^{-x}$ for $x \in [0,1]$

Theorem: For any min-cut $(S, \bar{S})$, Karger's algorithm returns $(S, \bar{S})$ with probability at least

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

If we run the algorithm K times and output the smallest cut from all K runs, the probability we **fail** to find a min cut is at most:

$$\mathbb{P}(\text{fail } K \text{ times}) \leq \left(1 - \frac{2}{n(n-1)}\right)^K \leq e^{\frac{-2K}{n(n-1)}} < \frac{1}{n^2}$$

If we choose $K \approx n^2 \log n$

Theorem: If we run the algorithm $K \approx n^2 \log n$ times, then

$$\Pr[\text{find a min cut}] \geq 1 - 1/n^2$$

Total running time?

One run can be implemented to run in time $O(n^2)$, so the total running time is $O(n^4 \log n)$ [pretty slow]

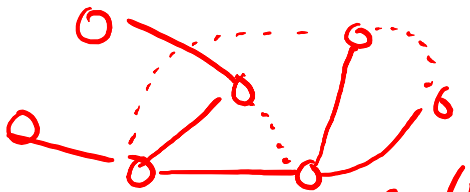
For fans of undergraduate algorithms:

Can use Kruskal’s algorithm for minimum spanning trees.

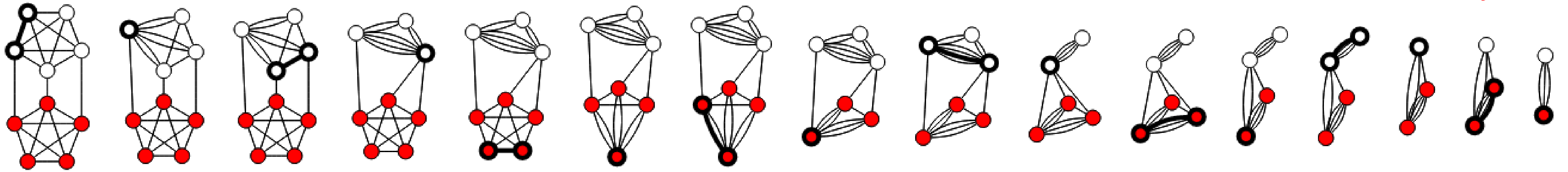
one way to implement

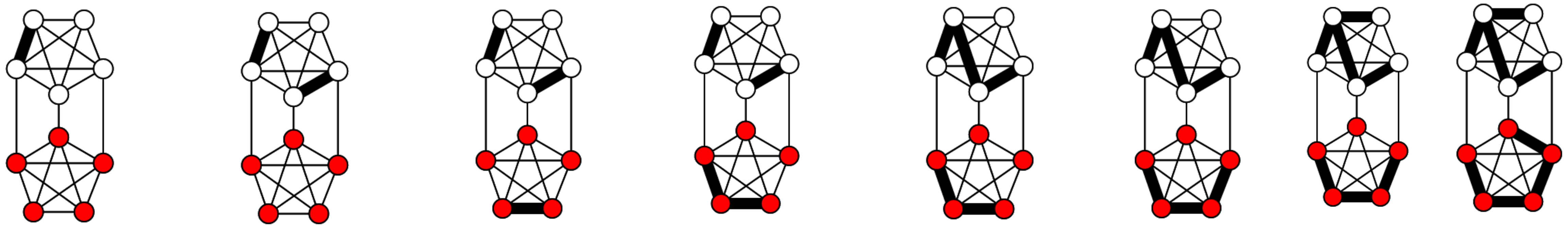
For fans of undergraduate algorithms:

Can use Kruskal's algorithm for minimum spanning trees.



$O(|E| \log |E|)$





Theorem: If we run the algorithm $K \approx n^2 \log n$ times, then

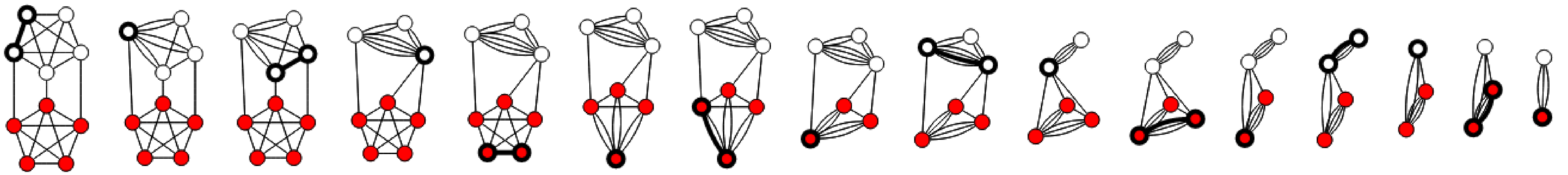
$$\Pr[\text{find a min cut}] \geq 1 - 1/n^2$$

Total running time?

One run can be implemented to run in time $O(n^2)$, so the total running time is $O(n^4 \log n)$ [pretty slow]

Improvement:

There is an algorithm that runs in time $O\left(n^2 (\log n)^3\right)$ and finds a global min-cut with probability close to 1.



$\frac{k(k-1)}{2}$ /

Observation:

$\rightarrow$

For any $k < n$: $\Pr[\neg\mathcal{A}_1 \wedge \neg\mathcal{A}_2 \wedge \cdots \wedge \neg\mathcal{A}_k] \geq \frac{\binom{k}{2}}{\binom{n}{2}}$

$\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{n-k-1}{n-k+1}$

$\approx \frac{(n-k-1)(n-k)}{n(n-1)}$

Therefore if $k \approx n/\sqrt{2}$, then the probability a min-cut $(S, \bar{S})$ remains after k steps is at least

$$\frac{\frac{k(k-1)}{2}}{\frac{n(n-1)}{2}} \approx \frac{\left(\frac{n}{\sqrt{2}}\right)^2}{n^2} = \frac{1}{2}$$

$k = \frac{n}{2}$ $\qquad \approx \frac{\left(\frac{n}{2}\right)^2}{n^2} \approx \frac{1}{4}$

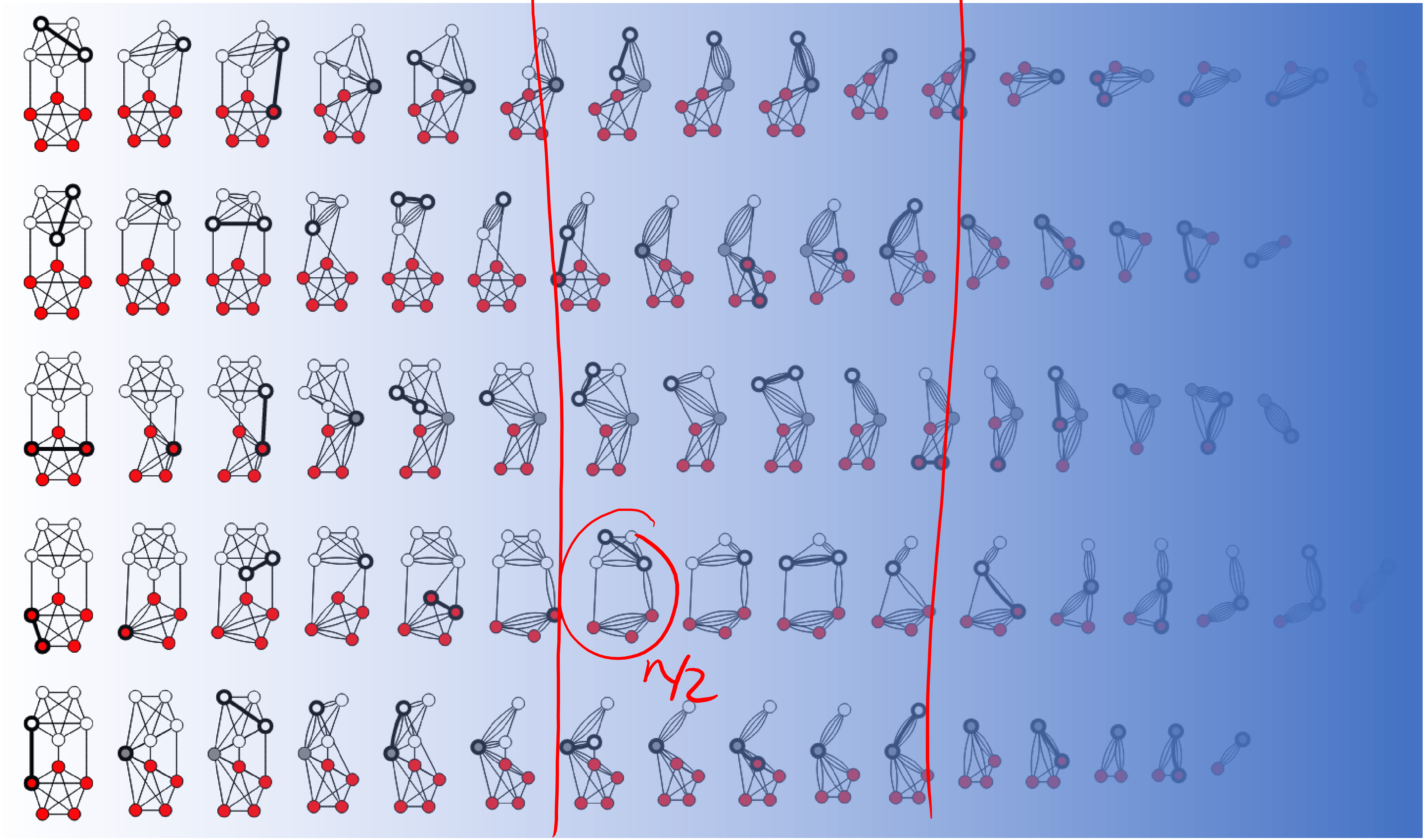
$k = n\left(1 - \frac{1}{\sqrt{2}}\right)$

Karger's randomized min cuts

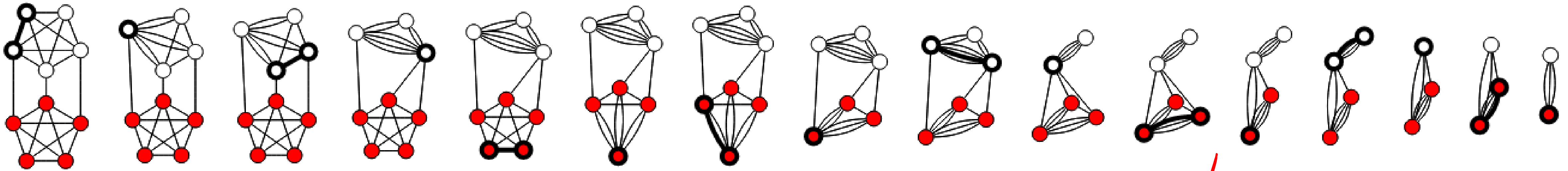
100

$k = n/2$ 100

$k = 3n/4$



$n/2$



$$\left(1 - \frac{1}{\log n}\right)^{\log n} \approx \frac{1}{e}$$

Observation:

For any $k < n$: $\Pr[\neg \mathcal{A}_1 \wedge \neg \mathcal{A}_2 \wedge \cdots \wedge \neg \mathcal{A}_k] \geq \frac{\binom{k}{2}}{\binom{n}{2}}$

Therefore if $k \approx n/\sqrt{2}$, then the probability a min-cut $(S, \bar{S})$ remains after k steps is at least

$$\frac{\frac{k(k-1)}{2}}{\frac{n(n-1)}{2}} \approx \frac{\left(\frac{n}{\sqrt{2}}\right)^2}{n^2} = \frac{1}{2}$$

So things are going better early on…

How to exploit this for a much faster algorithm?

levels of recursion: $\log_{\sqrt{2}} n$

$p(n) \geq 1/\log n$

```
procedure contract(G = (V, E), t):
while |V| > t
    choose e ∈ E uniformly at random
    G ← G/e
return G
```

```
procedure fastmincut(G = (V, E)):
if |V| ≤ 6:
    return mincut(V)
else:
    t ← ⌈1 + |V|/√2⌉
    G₁ ← contract(G, t)
    G₂ ← contract(G, t)
    return min {fastmincut(G₁), fastmincut(G₂)}
```

$t \leftarrow |V|/2$

$G_1 \leftarrow$ ⋰

$G_4 \leftarrow$

Probability we find a specific min-cut $(S, \bar{S})$ is given by the recurrence relation:

$$p(n) = 1 - \left(1 - \frac{1}{2} p\left(1 + \frac{n}{\sqrt{2}}\right)\right)^2$$

$$q(n) = 1 - \left(1 - \frac{1}{4} q\left(\frac{n}{2}\right)\right)^4$$

Running time of fastmincut:

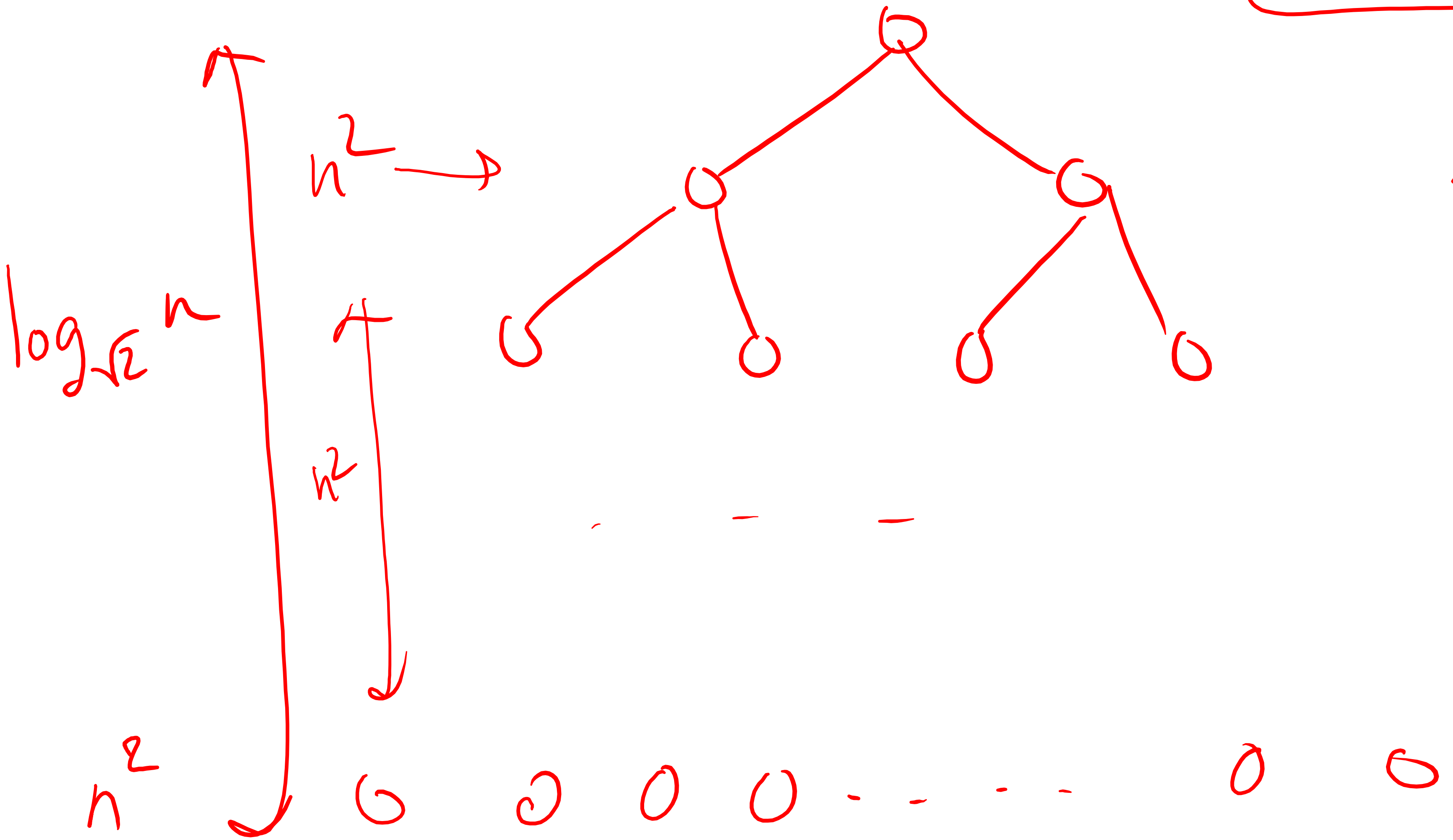
$$T(n) = 2T\left(1 + \frac{n}{\sqrt{2}}\right) + O(n^2)$$

$1 \cdot n$ $\quad$ $4 \; n/2$

$2 \; n/\sqrt{2}$ $\quad$ $8 \; n/2\sqrt{2}$

$$T(n) \le 2T(n/\sqrt{2}) + O(n^2)$$

$$\Longrightarrow \boxed{T(n) \le O(n^2 \log n)}$$



$$2^{\log_2(n^2)} = 2^{2\log_2 n}$$

$$n^2 = 2^{\log_{\sqrt{2}} n}$$

after the break: **hashing** and **sketching**