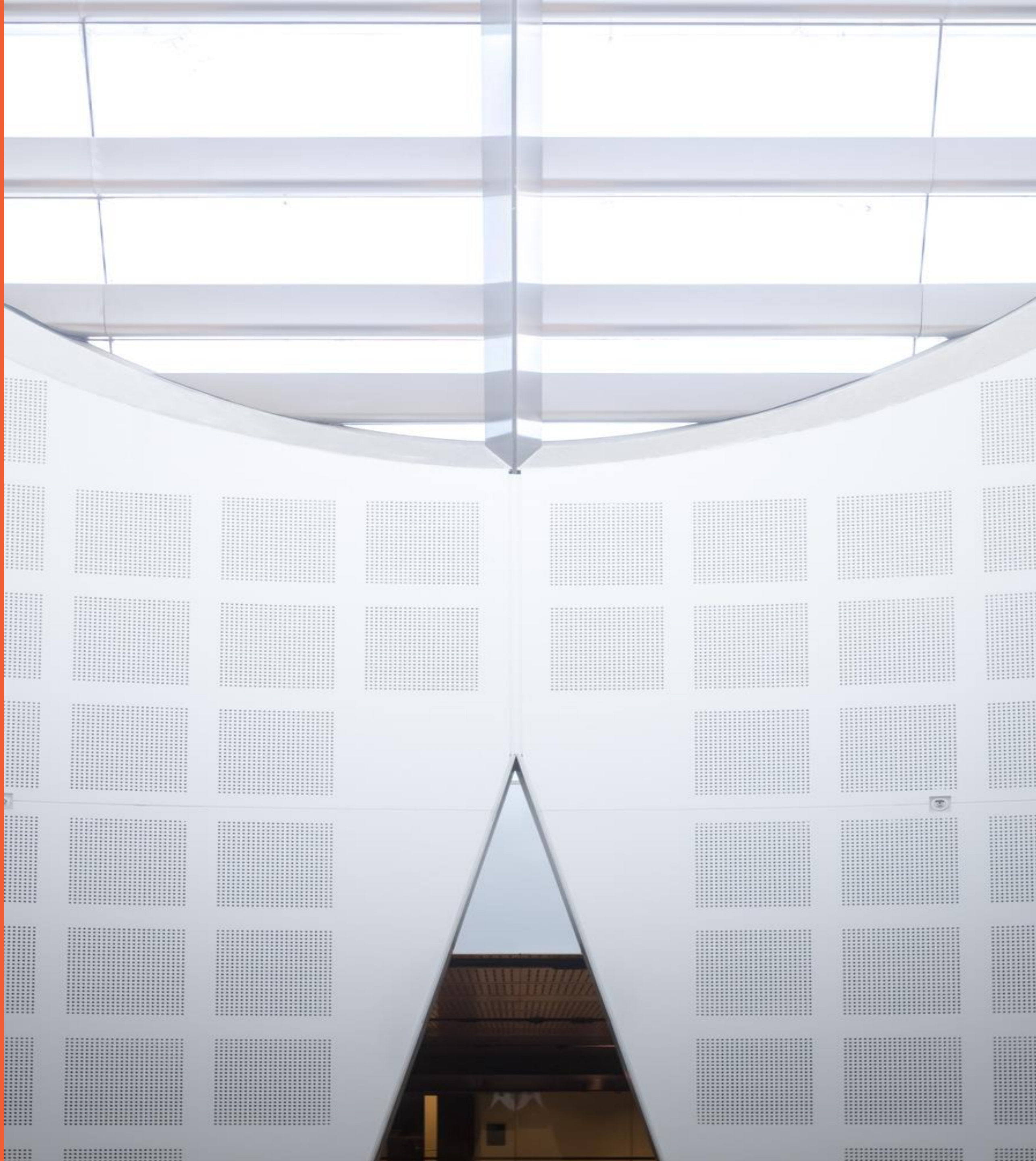
COMPx270: Randomised and Advanced Algorithms

Lecture 5: Graph algorithms

Clément Canonne

School of Computer Science

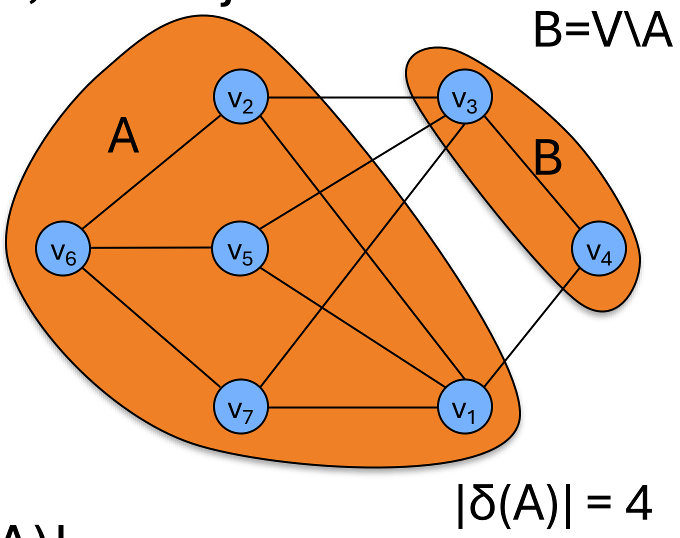
THE UNIVERSITY OF SYDNEY



Global Minimum Cut

Input: A connected, undirected graph $G = (V, E)$.

For a set $A \subseteq V$ let $\delta(A) = \{(u,v) \in E : u \in A, v \in V \setminus A\}$.

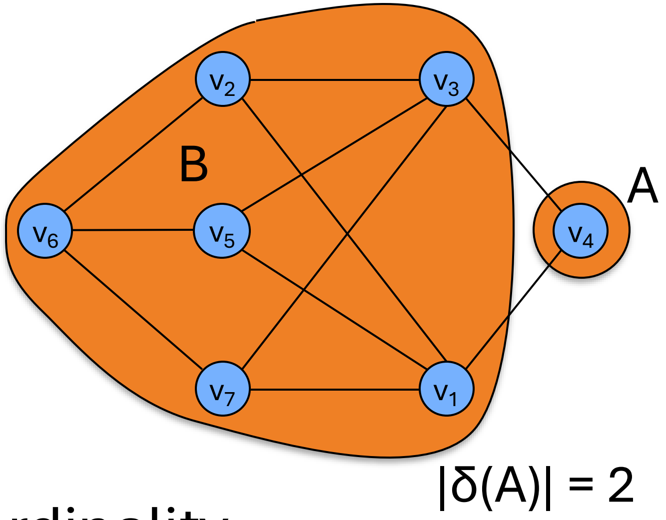


Aim: Find a cut (A, B) minimizing $|\delta(A)|$.

Global Minimum Cut

Input: A connected, undirected graph $G = (V, E)$.

For a set $A \subseteq V$ let $\delta(A) = \{(u,v) \in E : u \in A, v \in V \setminus A\}$.



Aim: Find a cut (A, B) of minimum cardinality.

Global Minimum Cut

Applications: Partitioning items in a database, identifying clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

- Replace every edge (u, v) with two directed edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cut separating s from each other vertex $v \in V$.

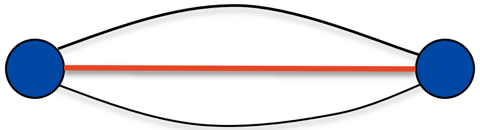
Running time: $O((n-1) \cdot \text{MaxFlows})$

Global Minimum Cut

Running time: $O((n-1) \cdot \text{MaxFlows})$

Karger's Contraction Algorithm

Definition. A multigraph is a graph that allows multiple edges between a pair of vertices.



Karger’s Contraction Algorithm

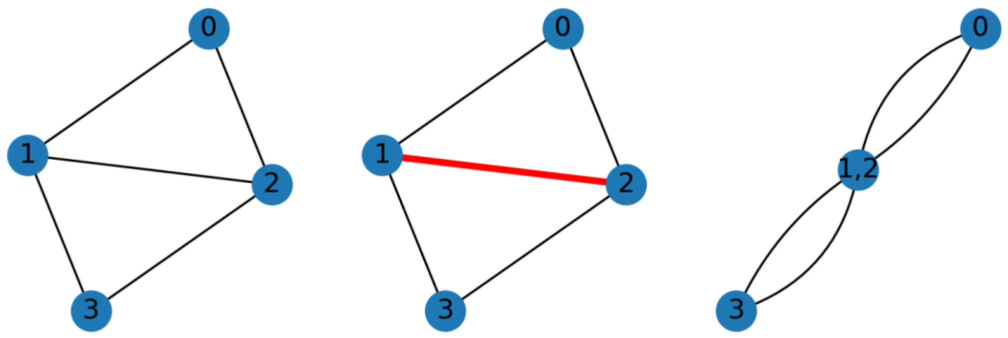
Definition. A multigraph is a graph that allows multiple edges between a pair of vertices.

Karger's Contraction Algorithm

Let G=(V,E) be a multigraph (without self-loops).

The contraction of an edge e=(u,v)∈E gives G\e

- Replace u and v by single new super-node w
- Replace all edges (u,x) or (v,x) with an edge (w,x)
- Remove self-loops to w



Karger's Contraction Algorithm

Require: multigraph $G = (V, E)$

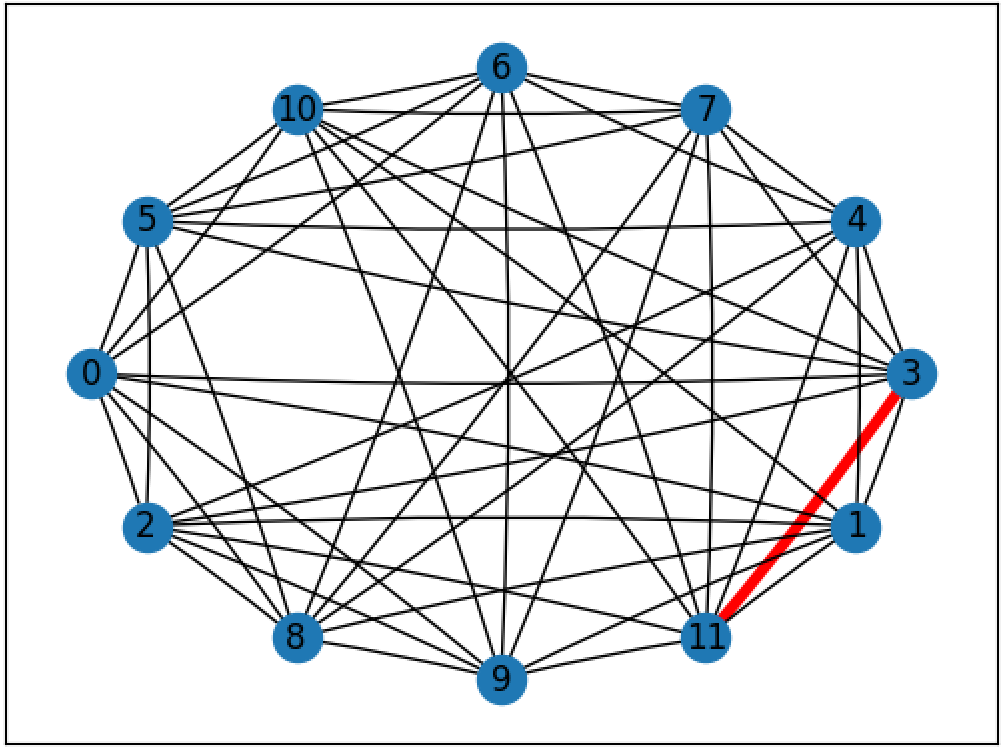
while $|V| > 2$ **do**

  Pick an edge $e \in E$ uniformly at random

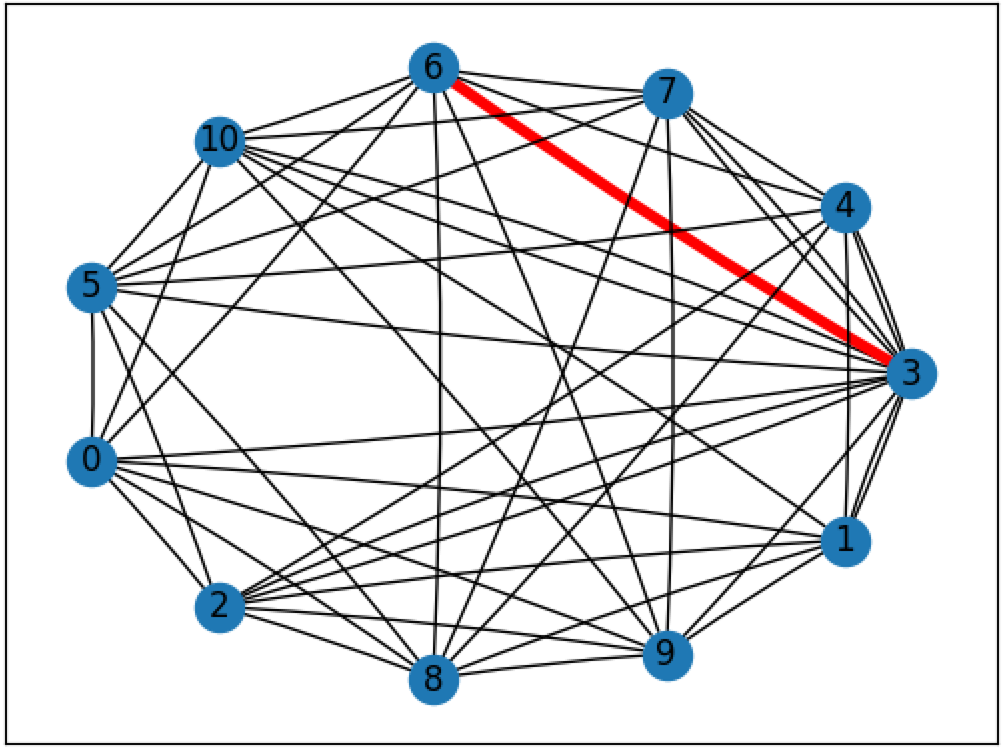
  Contract it, and let $G \leftarrow G/e$

return the cut defined by the remaining two vertices.

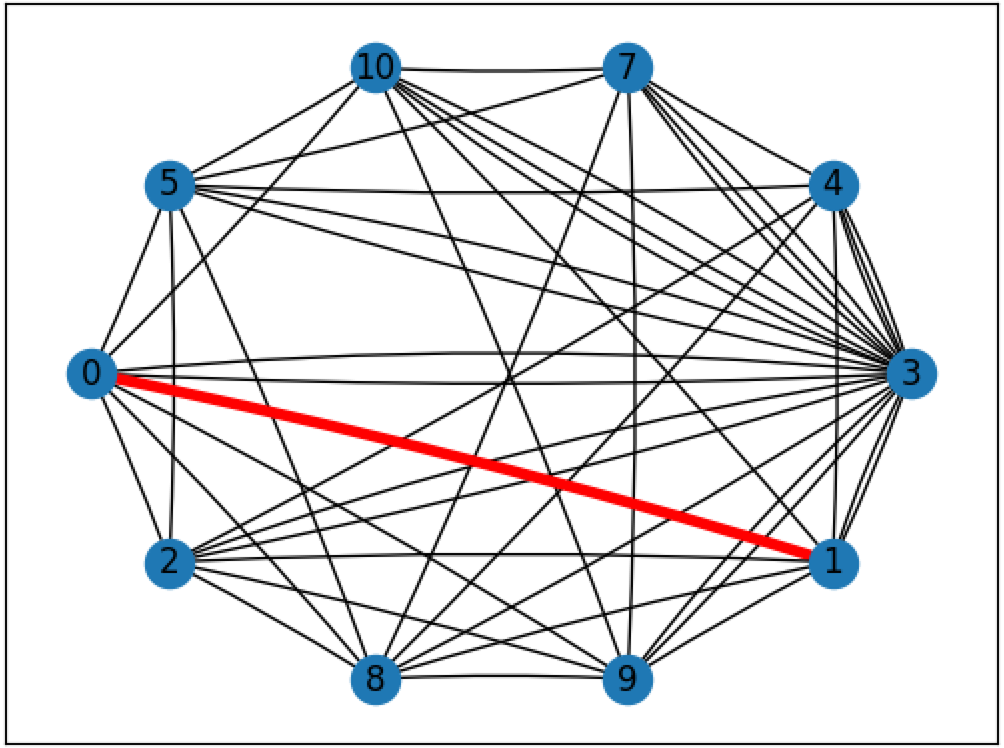
Karger's Contraction algorithm



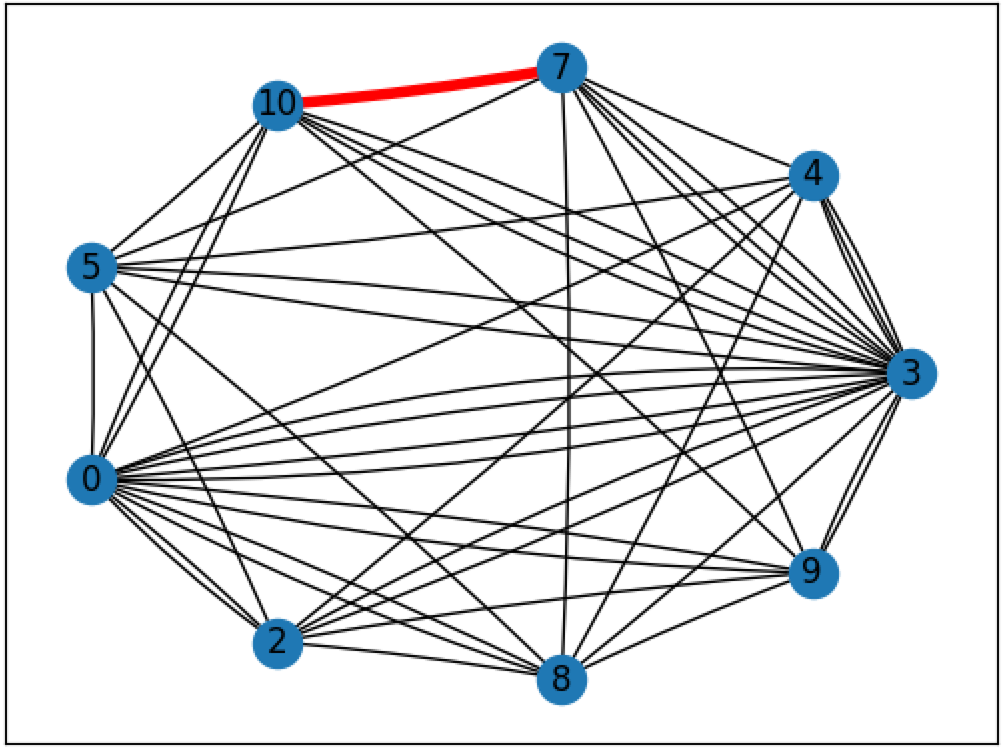
Karger’s Contraction algorithm



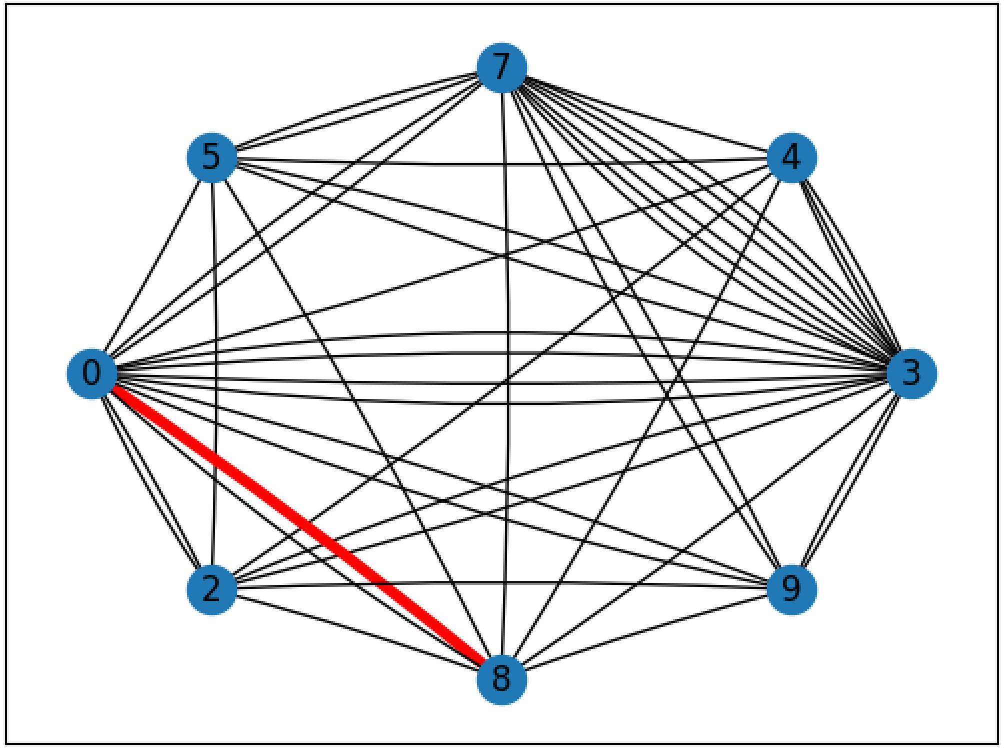
Karger's Contraction algorithm



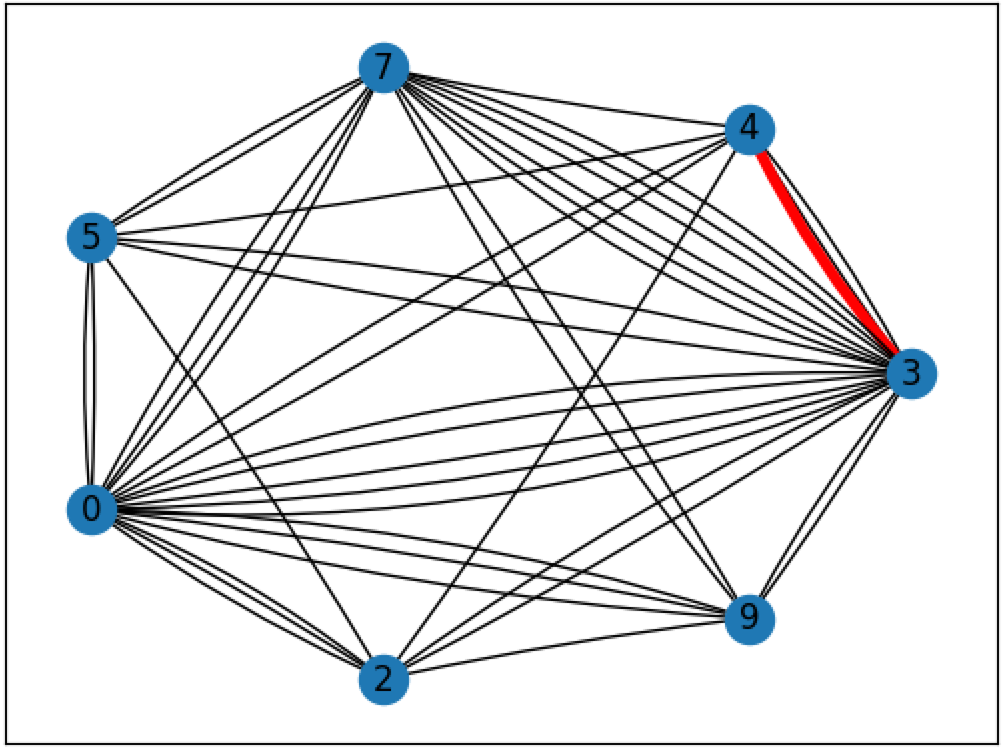
Karger’s Contraction algorithm



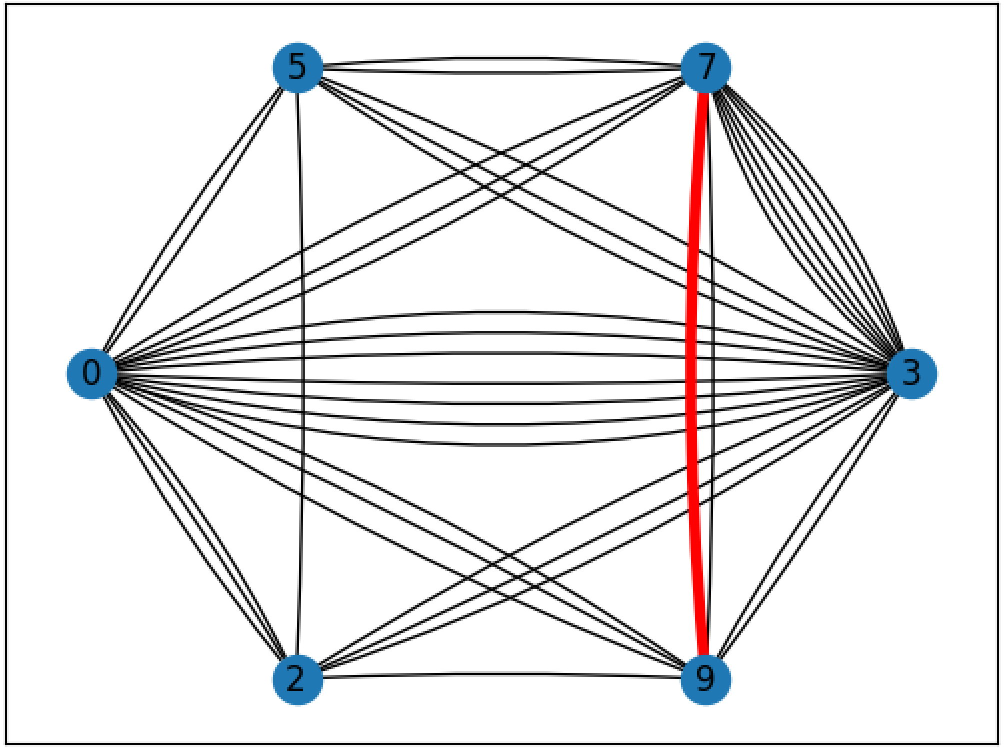
Karger's Contraction algorithm



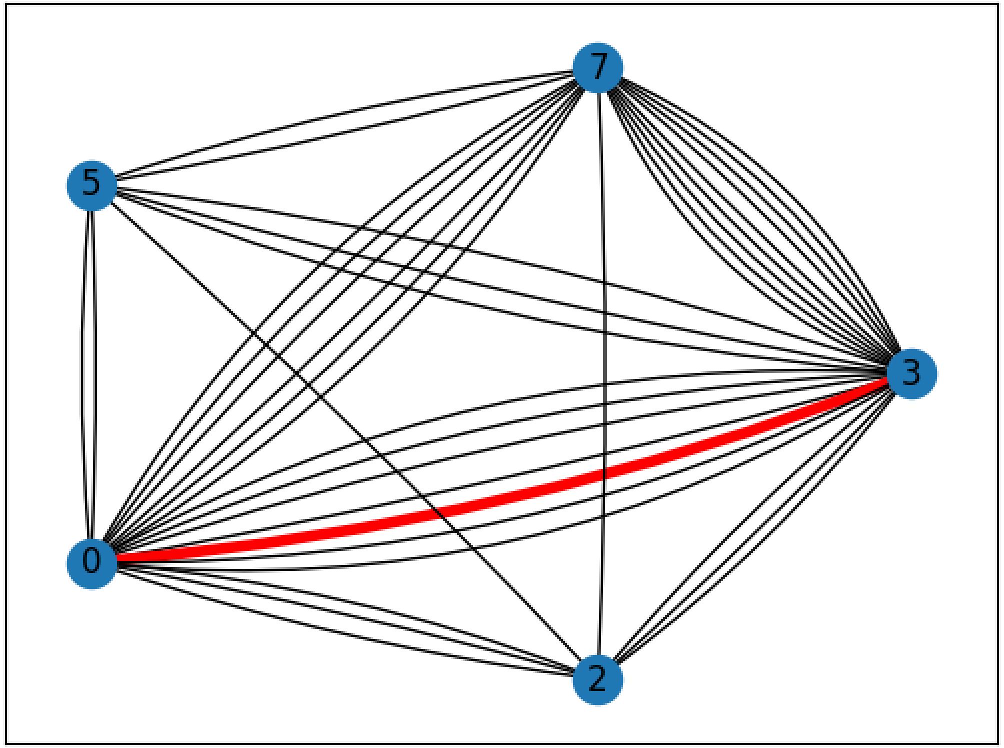
Karger’s Contraction algorithm



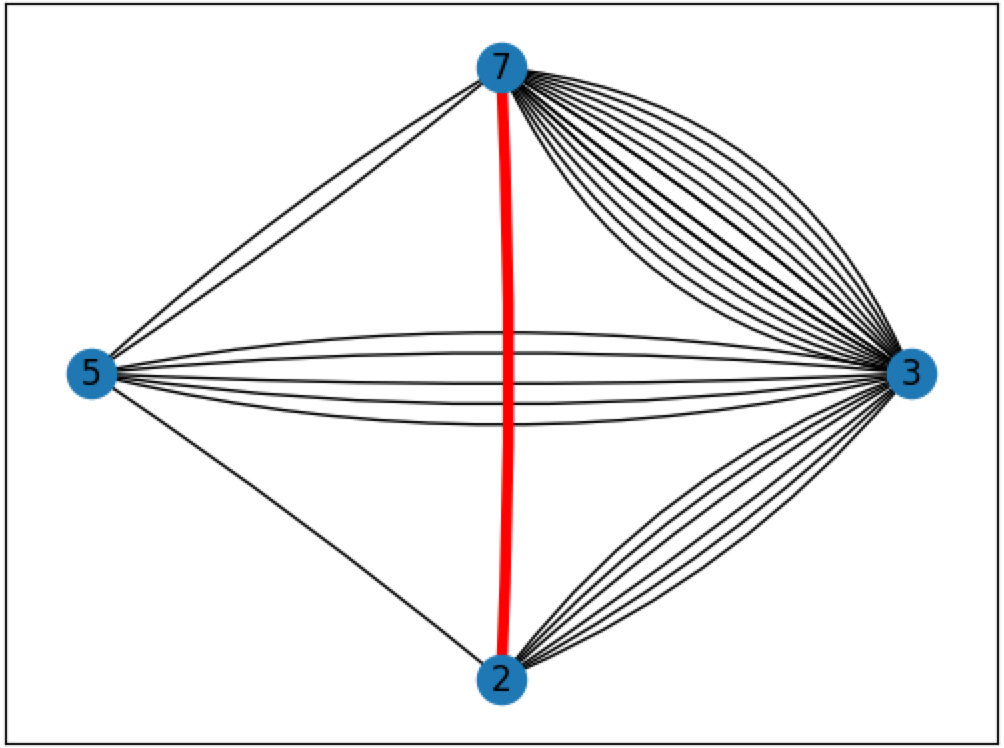
Karger's Contraction algorithm



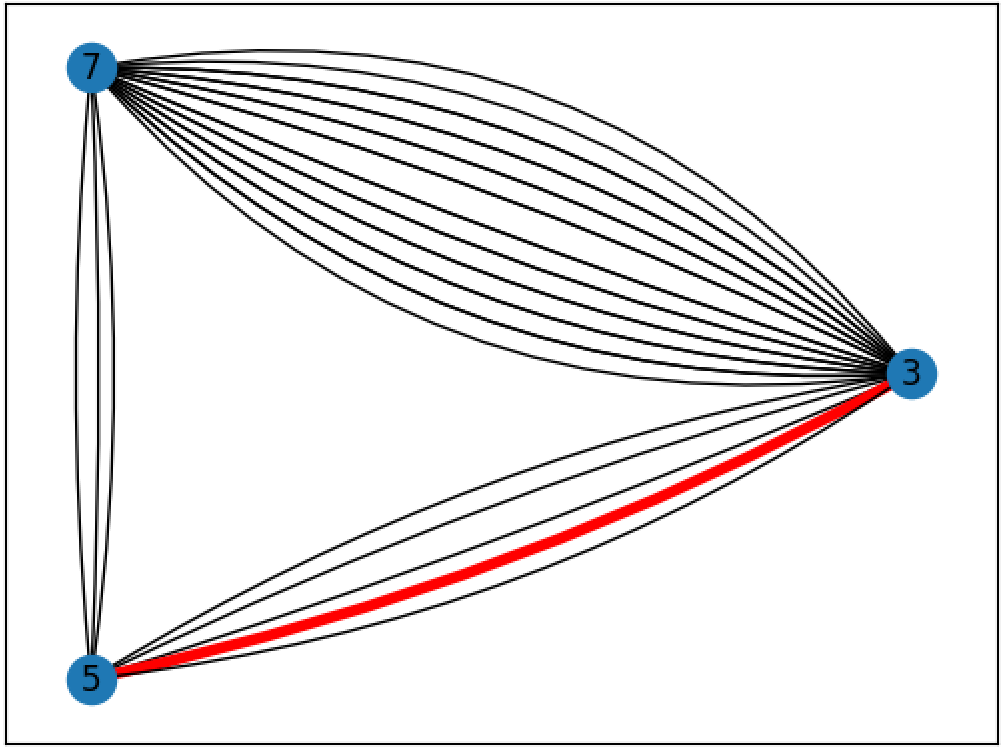
Karger’s Contraction algorithm



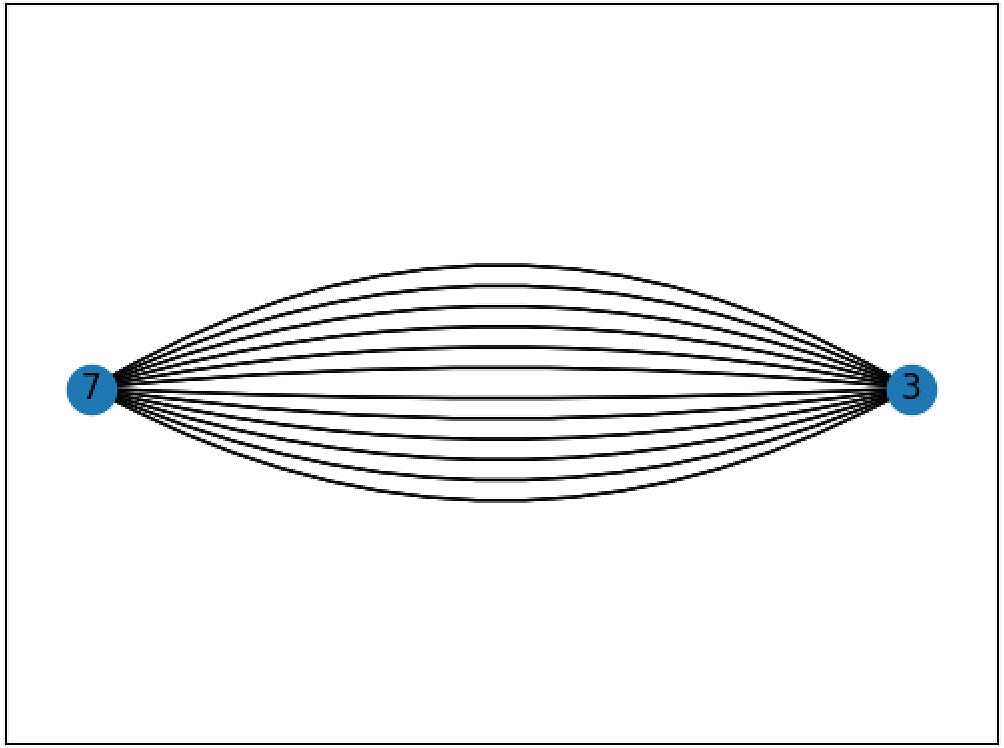
Karger's Contraction algorithm



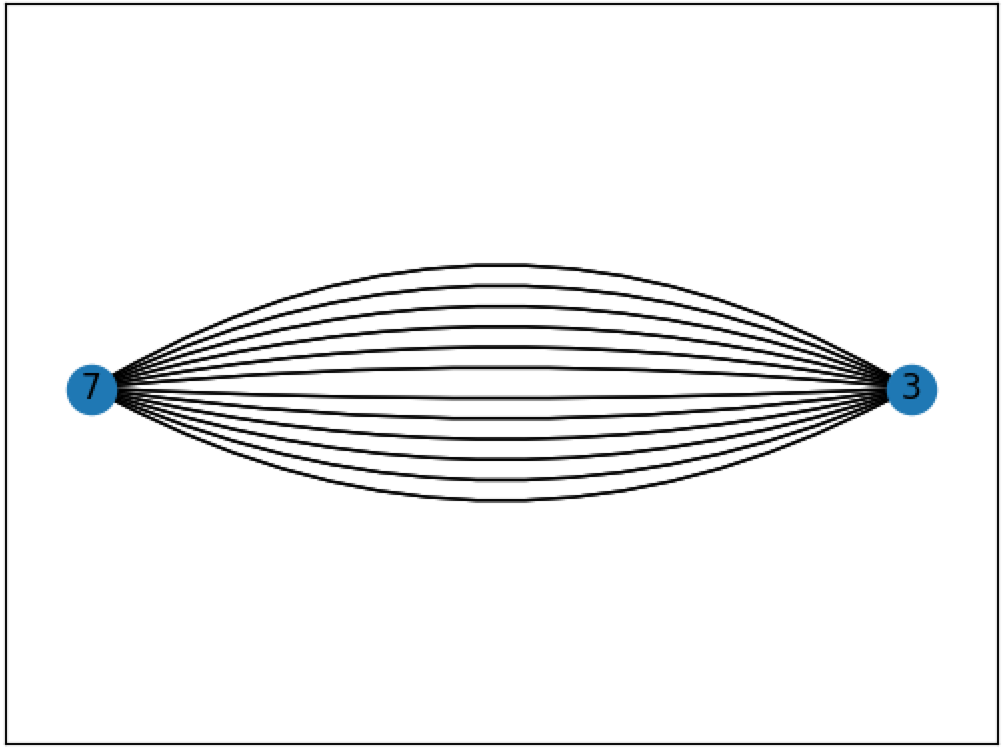
Karger's Contraction algorithm



Karger’s Contraction algorithm



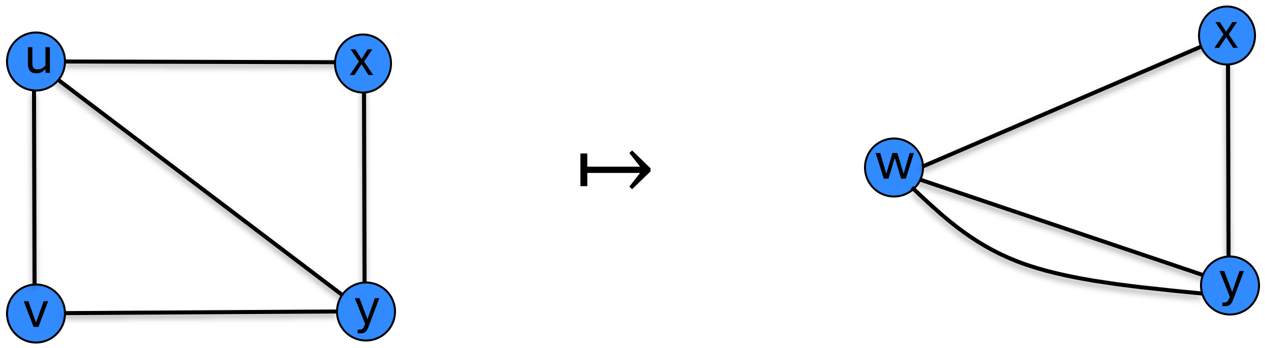
Karger's Contraction algorithm



The End.

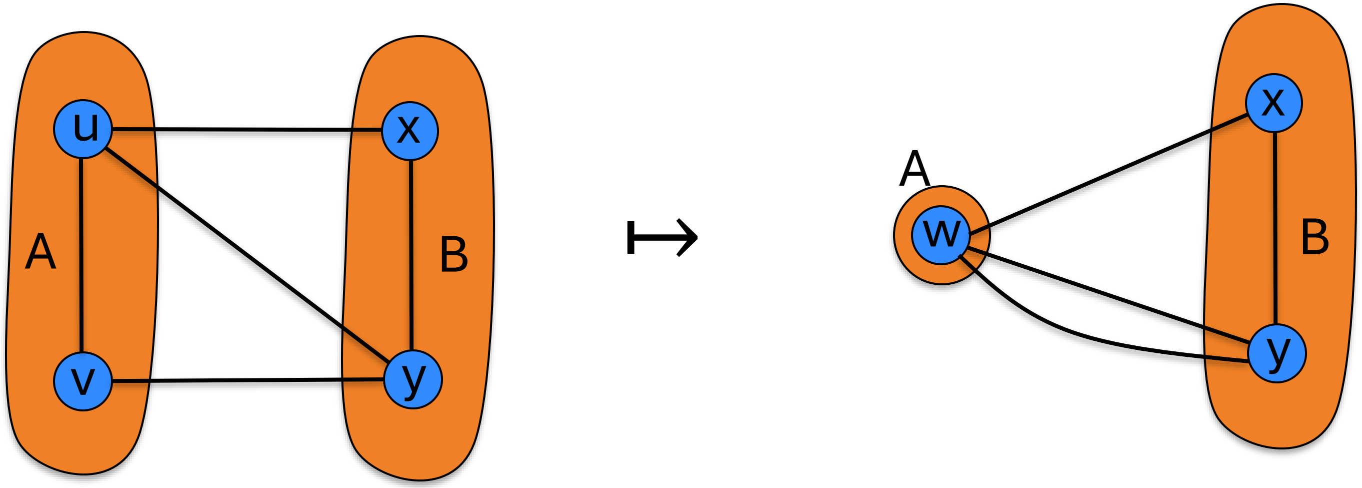
Karger's Contraction Algorithm

Observation: An edge (u,v) contraction preserves the cuts (A,B) where u and v are both in A or both in B.



Karger’s Contraction Algorithm

Observation: An edge (u,v) contraction preserves the cuts (A,B) where u and v are both in A or both in B.



Karger's Contraction Algorithm

Observation: An edge (u,v) contraction preserves the cuts (A,B) where u and v are both in A or both in B.

$$\boxed{\begin{array}{c}\text{If } u,v \in A \text{ then } \delta_G(A) = \delta_{G\backslash e}(A).\\ \text{(with u and v replaced with ``uv'')}\end{array}}$$

Karger's Contraction Algorithm

Observation: If (A,B) is a minimum cut, then we are less likely to choose an edge (u,v) crossing it!

Karger’s Contraction Algorithm

Require: multigraph $G = (V, E)$

while $|V| > 2$ **do**

  Pick an edge $e \in E$ uniformly at random

  Contract it, and let $G \leftarrow G/e$

return the cut defined by the remaining two vertices.

Claim. This algorithm has a reasonable chance of finding a min cut.

Key claim

Claim. If C is a min-cut, then the algorithm returns it with probability at least $2/n^2$.

Key claim

Claim. If C is a min-cut, then the algorithm returns it with probability at least $2/n^2$.

Proof.

Amplification

To amplify the probability of success, run the contraction algorithm many times.

Require: multigraph $G = (V, E)$, integer T

1: **for** $1 \leq t \leq T$ **do** ▷ Use fresh (independent) random bits for each

2: Run Algorithm on G, let C_t be the output

3: **return** the smallest cut among all cuts $C_1, \ldots, C_T$ obtained

---

Amplification

To amplify the probability of success, run the contraction algorithm many times.

Claim. If we repeat the contraction algorithm $r\binom{n}{2}$ times with independent random choices, the probability that all runs fail is at most $(1/e)^r$.

Karger’s Contraction Algorithm

Require: multigraph $G = (V, E)$

while $|V| > 2$ **do**

  Pick an edge $e \in E$ uniformly at random

  Contract it, and let $G \leftarrow G/e$

return the cut defined by the remaining two vertices.

Running time?

Karger's Contraction Algorithm

Require: multigraph $G = (V, E)$

1: **while** $|V| > 2$ **do**

2: Pick an edge $e \in E$ uniformly at random

3: Contract it, and let $G \leftarrow G/e$

4: **return** the cut defined by the remaining two vertices.

Running time?

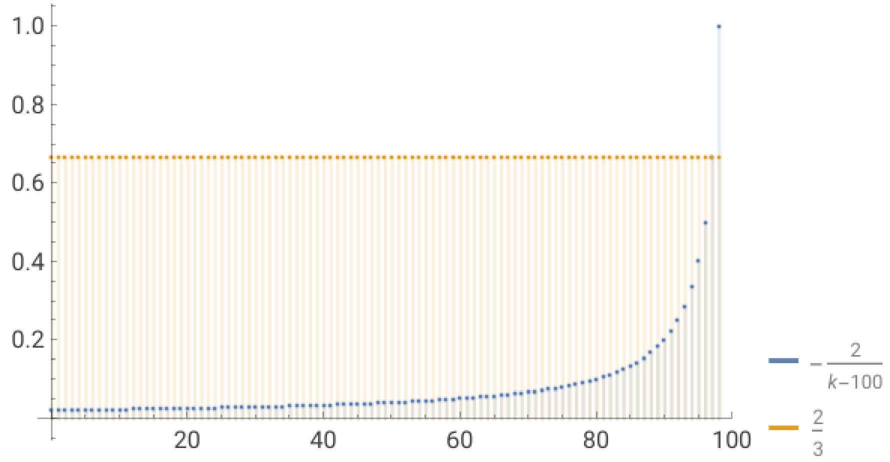
The algorithm is iterated $O(n^2 \log(1/\delta))$ times… total running time $O(n^4 \log(1/\delta))$.

Karger's Contraction Algorithm

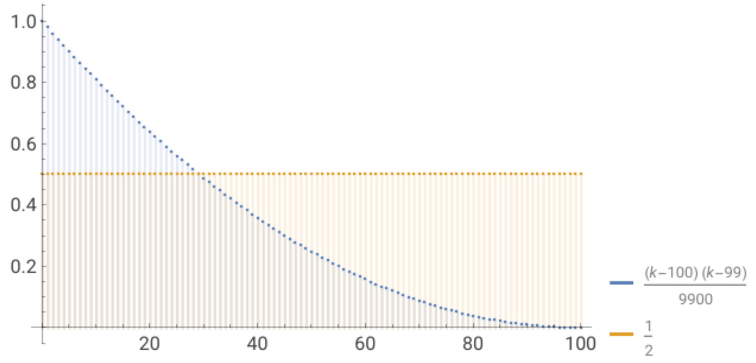
Can we do better?

Karger’s Contraction Algorithm

Can we do better?



Improved algorithm



Improvement. [Karger-Stein 1996]

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Improved algorithm

```
procedure ModifiedKarger(G = (V, E), s)
    while |V| > s do
        Pick an edge e ∈ E uniformly at random
        Contract it, and let G ← G/e
    return G
procedure KargerStein(G = (V, E))
    if |V| ≤ 6 then
        return a minimum cut                ▷ Brute-force computation
    Set s ← ⌈n/√2 + 1⌉
    ▷ Contraction
        G_1 ← ModifiedKarger(G, s)
        G_2 ← ModifiedKarger(G, s)
    ▷ Recursion
        C_1 ← KargerStein(G_1)
        C_2 ← KargerStein(G_2)
    return the smallest cut among C_1, C_2
```

Improved algorithm: Karger-Stein

```
procedure ModifiedKarger(G = (V, E), s)
    while |V| > s do
        Pick an edge e ∈ E uniformly at random
        Contract it, and let G ← G/e
    return G
procedure KargerStein(G = (V, E))
    if |V| ≤ 6 then
        return a minimum cut                    ▷ Brute-force computation
    Set s ← ⌈n/√2 + 1⌉
    ▷ Contraction
        G_1 ← ModifiedKarger(G, s)
        G_2 ← ModifiedKarger(G, s)
    ▷ Recursion
        C_1 ← KargerStein(G_1)
        C_2 ← KargerStein(G_2)
    return the smallest cut among C_1, C_2
```

Running time?

Improved algorithm: Karger-Stein

Running time?

```
procedure ModifiedKarger(G = (V, E), s)
    while |V| > s do
        Pick an edge e ∈ E uniformly at random
        Contract it, and let G ← G/e
    return G
procedure KargerStein(G = (V, E))
    if |V| ≤ 6 then
        return a minimum cut                ▷ Brute-force computation
    Set s ← ⌈n/√2 + 1⌉
    ▷ Contraction
        G1 ← ModifiedKarger(G, s)
        G2 ← ModifiedKarger(G, s)
    ▷ Recursion
        C1 ← KargerStein(G1)
        C2 ← KargerStein(G2)
    return the smallest cut among C1, C2
```

Improved algorithm: Karger-Stein

procedure ModifiedKarger($G = (V, E), s$)
    while $|V| > s$ **do**
        Pick an edge $e \in E$ uniformly at random
        Contract it, and let $G \leftarrow G/e$
    return G
procedure KargerStein($G = (V, E)$)
    if $|V| \leq 6$ **then**
        return a minimum cut   ▷ Brute-force computation
    Set $s \leftarrow \lceil n/\sqrt{2} + 1 \rceil$
    ▷ **Contraction**
        $G_1 \leftarrow$ ModifiedKarger(G, s)
        $G_2 \leftarrow$ ModifiedKarger(G, s)
    ▷ **Recursion**
        $C_1 \leftarrow$ KargerStein(G_1)
        $C_2 \leftarrow$ KargerStein(G_2)
    return the smallest cut among C_1, C_2

Success probability?

Improved algorithm: Karger-Stein

Success probability?

procedure ModifiedKarger($G = (V, E), s$)
  while $|V| > s$ **do**
    Pick an edge $e \in E$ uniformly at random
    Contract it, and let $G \leftarrow G/e$
  return G
procedure KargerStein($G = (V, E)$)
  if $|V| \leq 6$ **then**
    return a minimum cut ▷ Brute-force computation
  Set $s \leftarrow \lceil n/\sqrt{2} + 1 \rceil$
  ▷ **Contraction**
    $G_1 \leftarrow$ ModifiedKarger(G, s)
    $G_2 \leftarrow$ ModifiedKarger(G, s)
  ▷ **Recursion**
    $C_1 \leftarrow$ KargerStein(G_1)
    $C_2 \leftarrow$ KargerStein(G_2)
  return the smallest cut among C_1, C_2

Improved algorithm: Karger-Stein

Theorem. [Karger-Stein 1996] The Karger-Stein algorithm runs in time $O(n^2 \log n)$ and returns a min cut with probability at least $\Omega(1/\log n)$.

Corollary. The "best-of-T" Karger-Stein algorithm runs in time $O(n^2 \log^2 n)$ and returns a min cut with probability at least 99%.

Improved algorithm: Karger-Stein

Theorem. [Karger-Stein 1996] The Karger-Stein algorithm runs in time $O(n^2 \log n)$ and returns a min cut with probability at least $\Omega(1/\log n)$.

Corollary. The "best-of-T" Karger-Stein algorithm runs in time $O(n^2 \log^2 n)$ and returns a min cut with probability at least 99%

Best known. [Karger 2000] $O(m \log^3 n)$.

And now, for something completely different

And now, for something completely different?

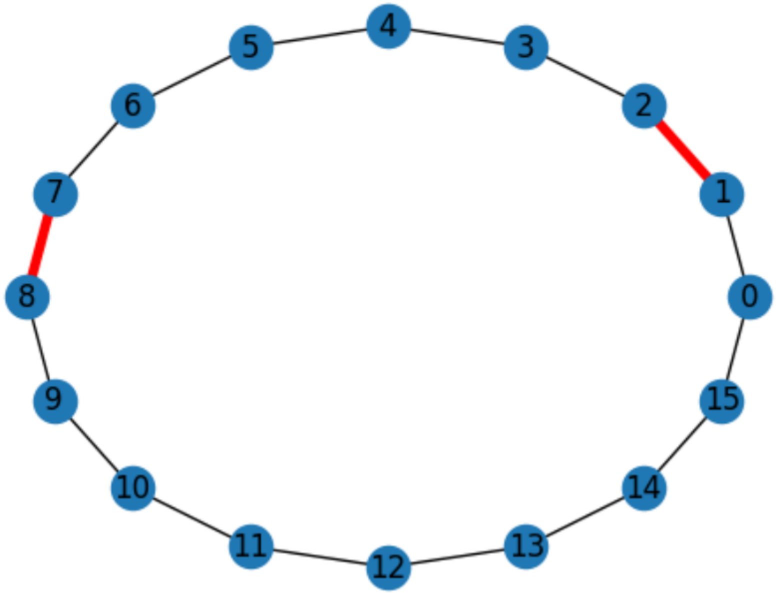
Theorem. An undirected graph G=(V,E) has at most _______ distinct min cuts.

And now, for something completely different?

Theorem. An undirected graph G=(V,E) has at most $n(n-1)/2$ distinct min cuts.

And now, for something completely different?

Theorem. An undirected graph G=(V,E) has at most n(n-1)/2 distinct min cuts. And this is tight.



And now, for something completely different?

Theorem. An undirected graph G=(V,E) has at most n(n-1)/2 distinct min cuts.

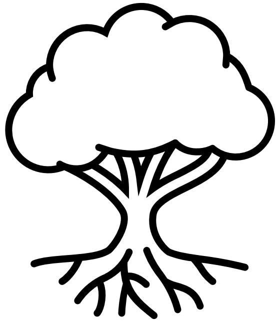
Proof.

And now, for something completely different?

How mind-blowing is this? Analyzing an algorithm proved a seemingly unrelated mathematical statement. 🤯

Compare this with last time, and the probabilistic method ("proof without algorithm!").

And now, for something else completely different!



Minimum Spanning Tree in (Expected) Linear Time: Karger-Klein-Tarjan.