Randomized Min Cut

Probability

Events, probabilities,
conditional probabilities,
union bound
Random variables,
expected value,
linearity of expectation
Markov's inequality

Algorithms

Approx. 3-SAT,

Big idea: we only want the average

Linearity of Expectation
$E[X+Y]=E[X]+E[Y]$ (absolutely useful)
"average sum = sum of averages"

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$ $(i=1,\dots,m)$

$E[Y_i] = 7/8$ (1 clause case)

$E[\#\text{clauses satisfied}] = E\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} E[Y_i] = 7/8\,m$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$Z = \sum_{i=1}^{m} Y_i = \#\text{ clauses satisfied}$

$E[Z] = \frac{1}{2}E[Z \mid x_1=T] + \frac{1}{2}E[Z \mid x_1=F]$

$\Rightarrow$ either $E[Z \mid x_1=T]$ or $E[Z \mid x_1=F]$ is $\geq E[Z]$!

Quicksort,

Random divide and conquer

① pick random "pivot"

② Divide A into two parts

< pivot		> pivot
sort recursively		sort recursively
< pivot, sorted		> pivot, sorted

Expected running time of QuickSort

For each $i,j \in [n]$ w/ $i<j$,

$X_{ij} = \begin{cases} 1 & \text{if } i\text{th smallest element is compared to } j\text{th sma--} \\ 0 & \text{otherwise} \end{cases}$

$\sum_{ij} X_{ij}$ = total # comparisons made by the algorithm

$\approx$ total running time

$E[\sum_{ij} X_{ij}]$ = average running time

$E[\sum_{ij} X_{ij}] = \sum_{ij} E[X_{ij}]$ (Linearity of Expectation; Average sum = sum of averages)

now analyze $E[X_{ij}]$ in isolation

$E[X_{ij}] = \Pr[X_{ij}=1]$

$= \Pr[\,i\text{th element is compared to } j\text{th}\,]$

ith elem is compared w/ jth
$\Leftrightarrow$ ith or jth elem selected before any element in between
happens w/ probability $\frac{2}{j-i+1}$

$E[X_{ij}] = \frac{2}{j-i+1}$

$E[\sum_{ij} X_{ij}] = \sum_{ij} E[X_{ij}]$ (by linearity of expectation)

$= \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n}\sum_{j=1}^{n-i+1} \frac{2}{j+1}$

$\leq 2n\left(\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{n}\right)$

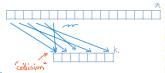
$\leq O(n \log n)$ nth harmonic number

Quickselect (HW)

Probability

Hash Functions,

Universal Hash Functions



$h: [n] \to k$ s.t. for all pairs $i, j \in [n]$,

$$\Pr[h(i) = h(j)] \leq \frac{1}{k}$$

(much weaker than ideal hash)

e.g. $h(x) = ((ax+b) \bmod p) \bmod k$

where
- p prime, $> k$
- a in $\{1, \sim, p\}$ unif. random
- b in $\{0, \sim, p-1\}$ unif. random

Markov's Inequality

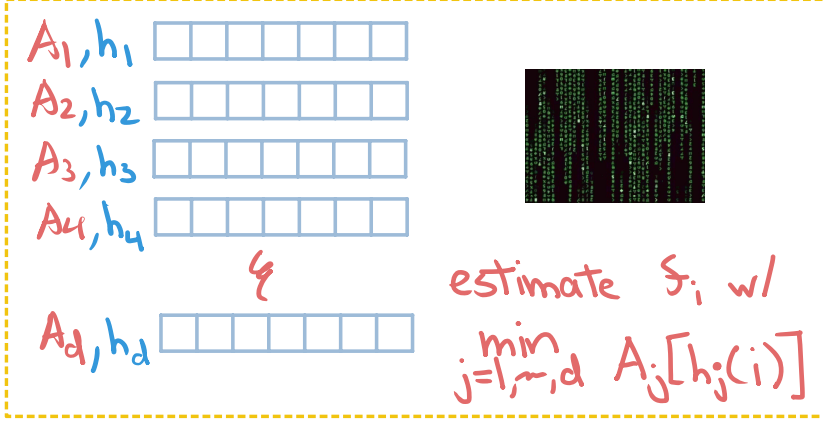
Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Algorithms

Frequency Estimation, Heavy Hitters (in streams)



estimate f_i w/

$$\min_{j=1,\sim,d} A_j[h_j(i)]$$

Probability

k-wise independence, higher order moments

(generalizating universal hash functions and Markov's inequality)

Algorithms

Hash tables w/ chaining + linear probing

k-wise independent hash functions

$h: [U] \to [m]$

s.t. for any k keys $x_1, x_2, \dots, x_k \in [U]$, k values $v_1, v_2, \dots, v_k \in [U]$,

$$P[h(x_1)=v_1, h(x_2)=v_2, \dots, h(x_k)=v_k] = \frac{1}{m^k}$$

e.g. $h(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{k-1} x^{k-1} \bmod p$

where $p \geq U$ prime,

$a_0, a_1, \dots, a_{k-1} \in [p]$ indep, uniformly random

Important lemma

k-wise independent random variables

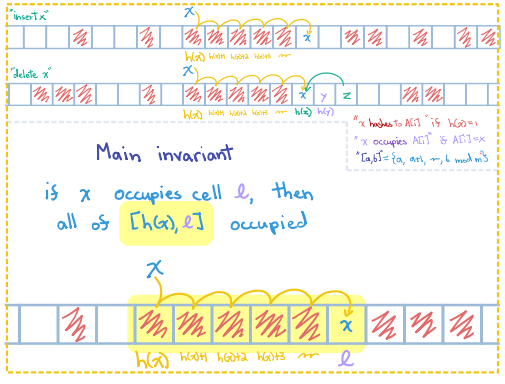
$P_r[X_{i_1}=y_1, X_{i_2}=y_2, \dots, X_{i_k}=y_k]$

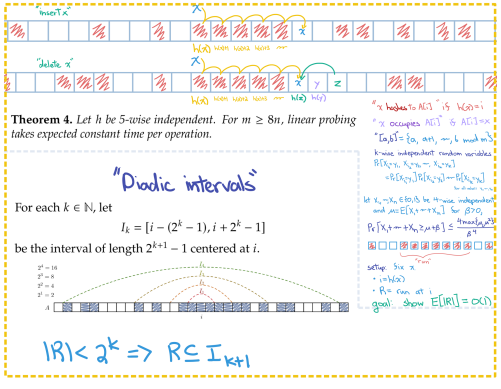
$= P_r[X_{i_1}=y_1] P_r[X_{i_2}=y_2] \cdots P_r[X_{i_k}=y_k]$

for all values $y_1, \dots, y_k$

let $X_1, \dots, X_n \in \{0,1\}$ be 4-wise independent and $\mu = E[X_1 + \dots + X_n]$. then for $\beta > 0$,

$$P_r[X_1 + \dots + X_n \geq \mu + \beta] \leq \frac{4 \max\{\mu, \mu^2\}}{\beta^4}$$





Today

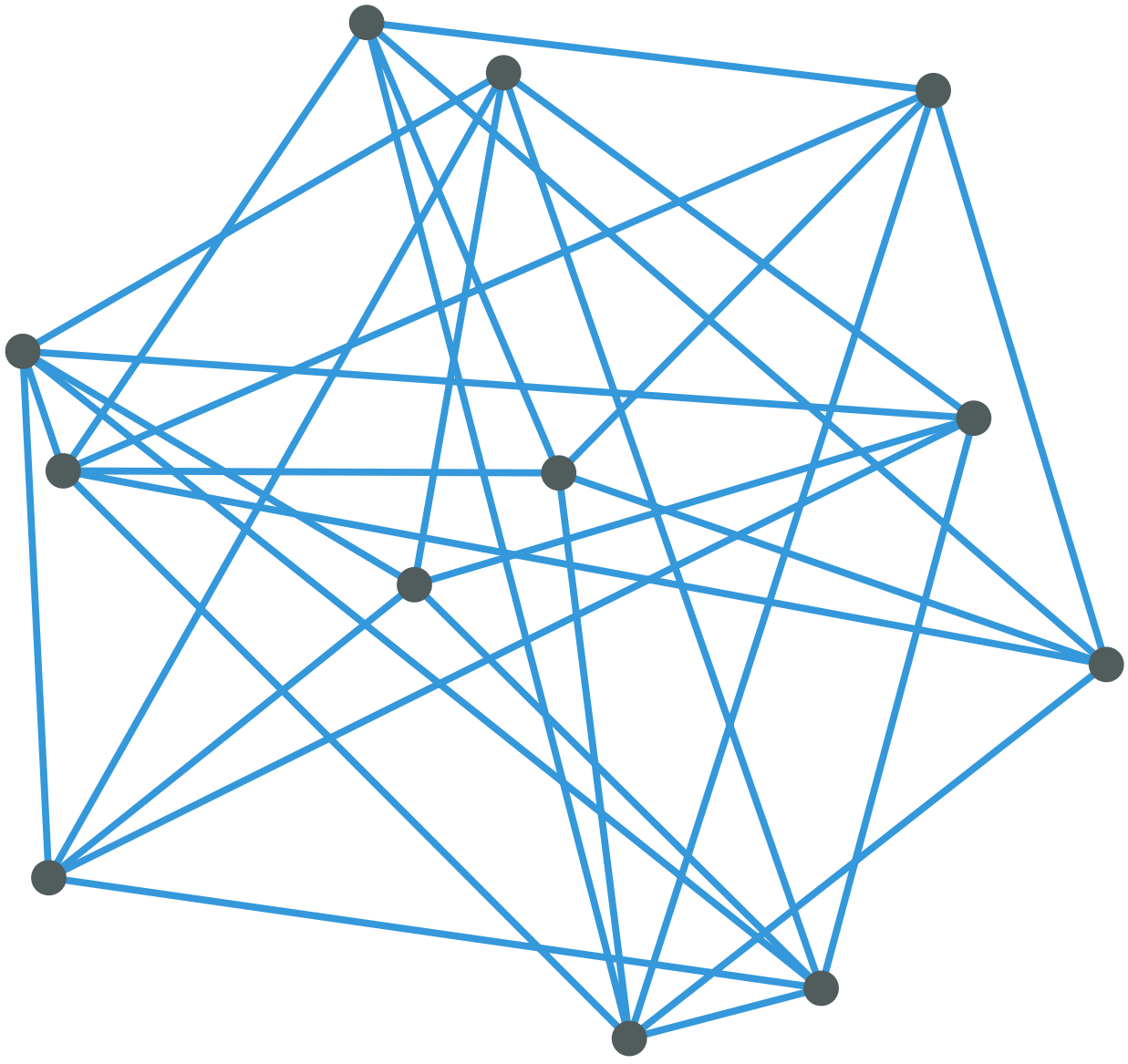
Probability

Galton-Watson processes

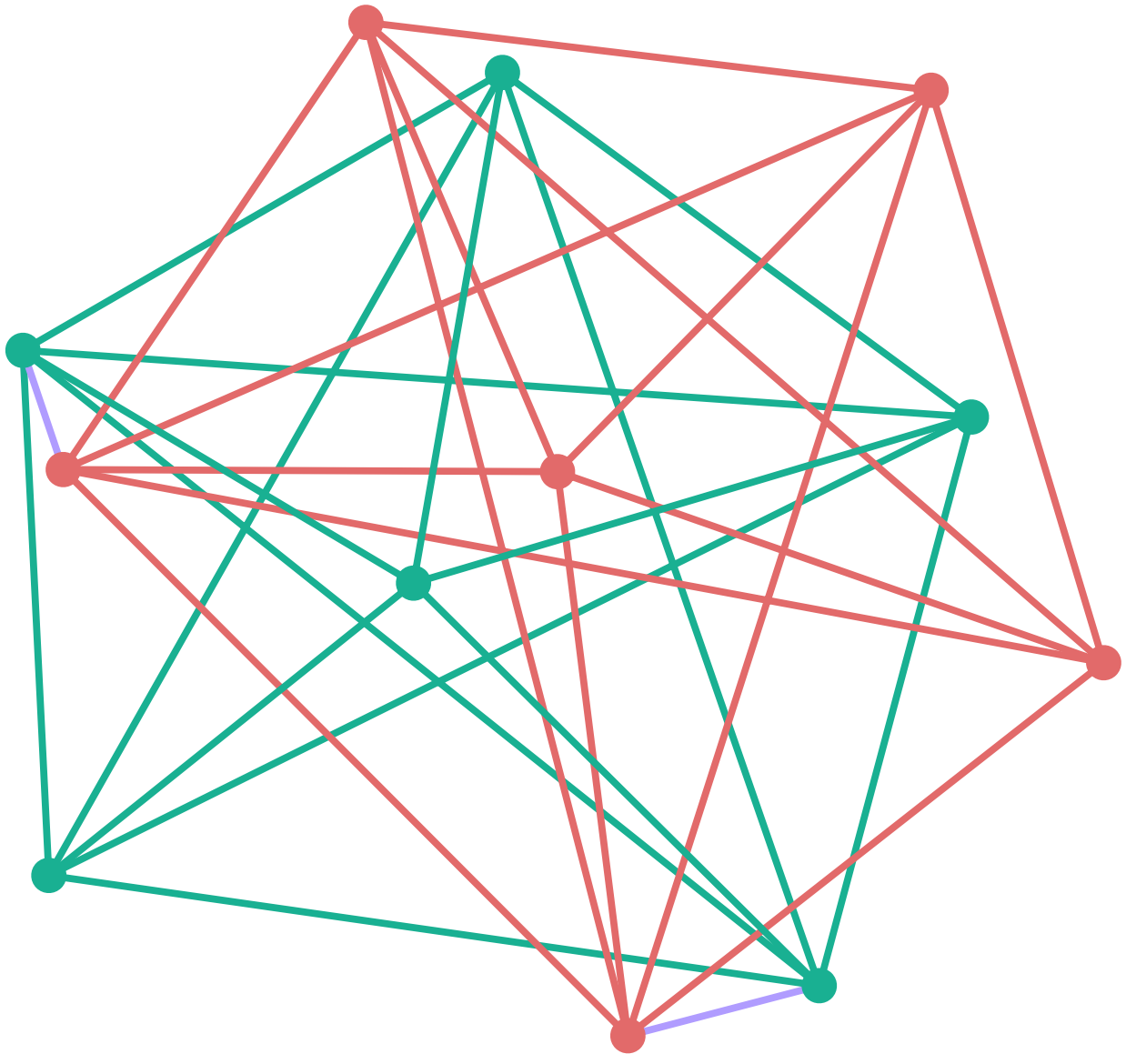
Algorithms

Random Contractions for min cut

Goal: disconnect graph by removing min # edges



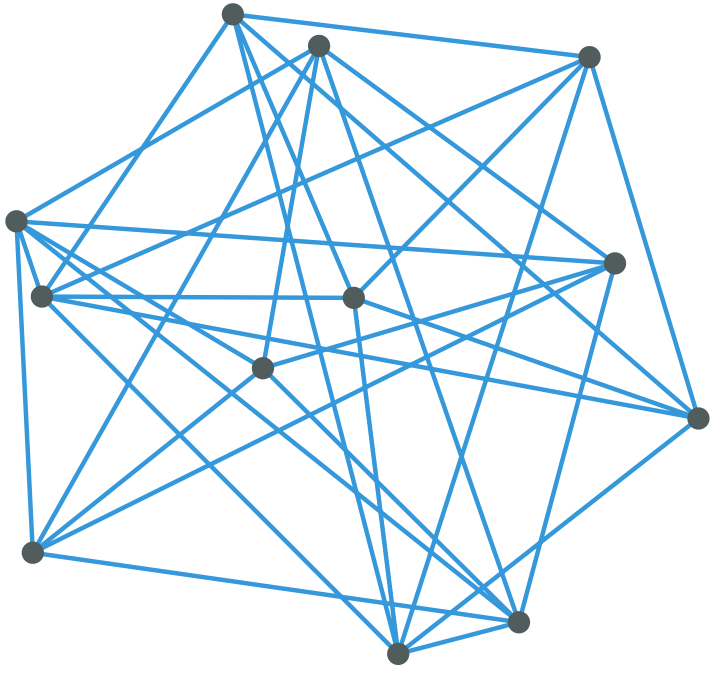
Goal: disconnect graph by removing min # edges

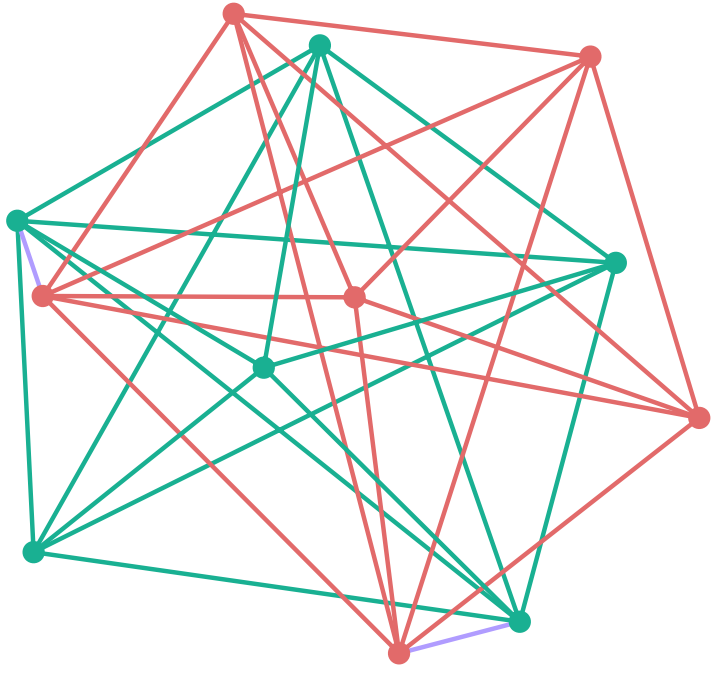


Global Minimum Cut

Input: Connected, Undirected Graph

Goal: disconnect graph by removing min # edges



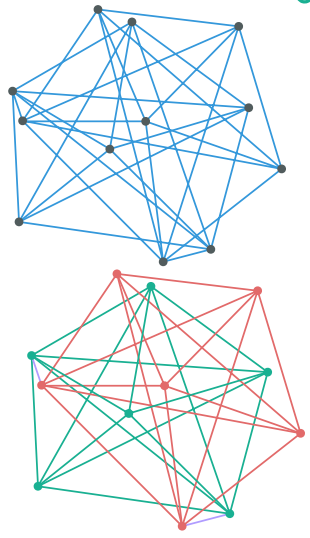


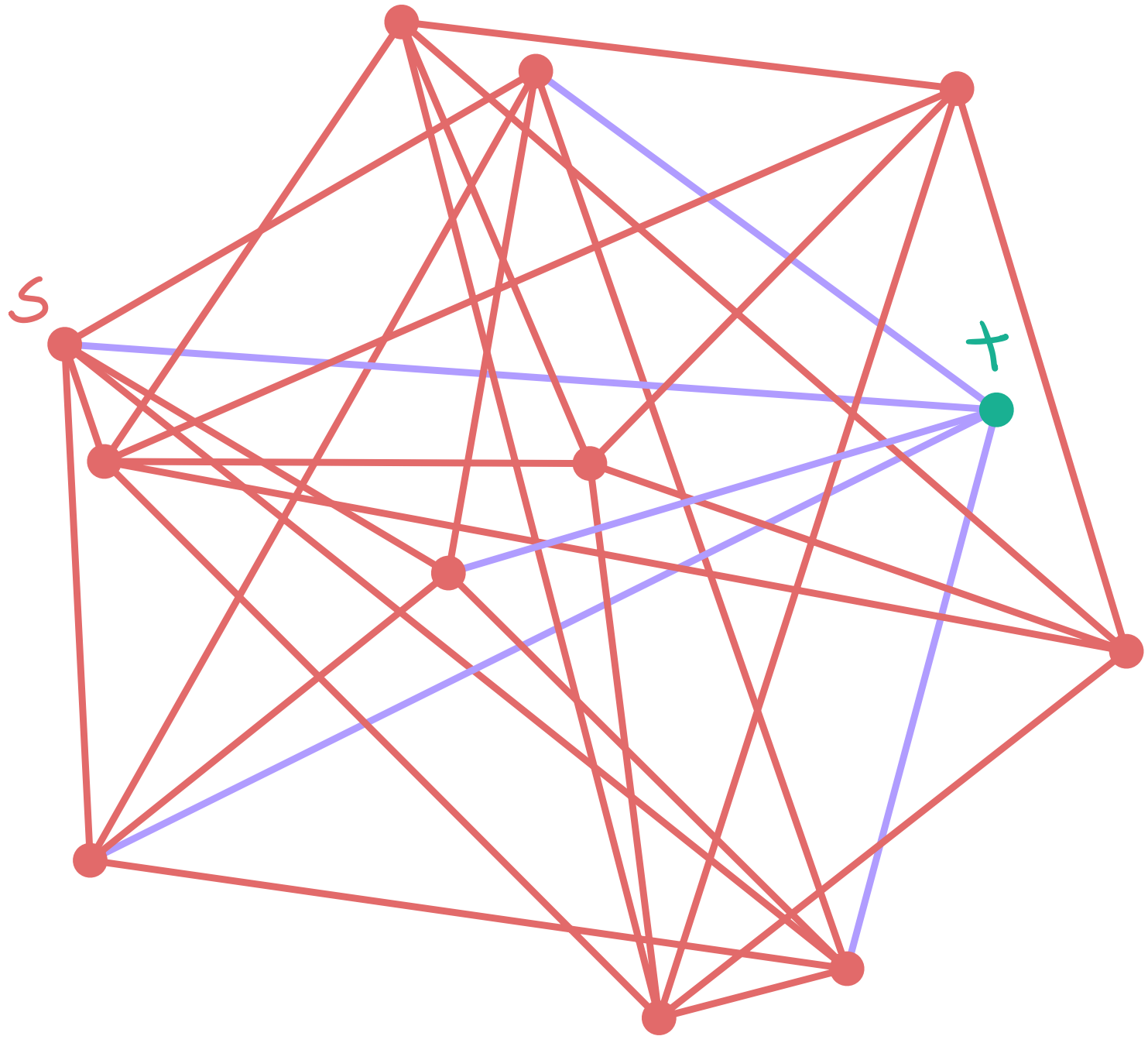
Also, "minimum weight cut" w/ edge capacities

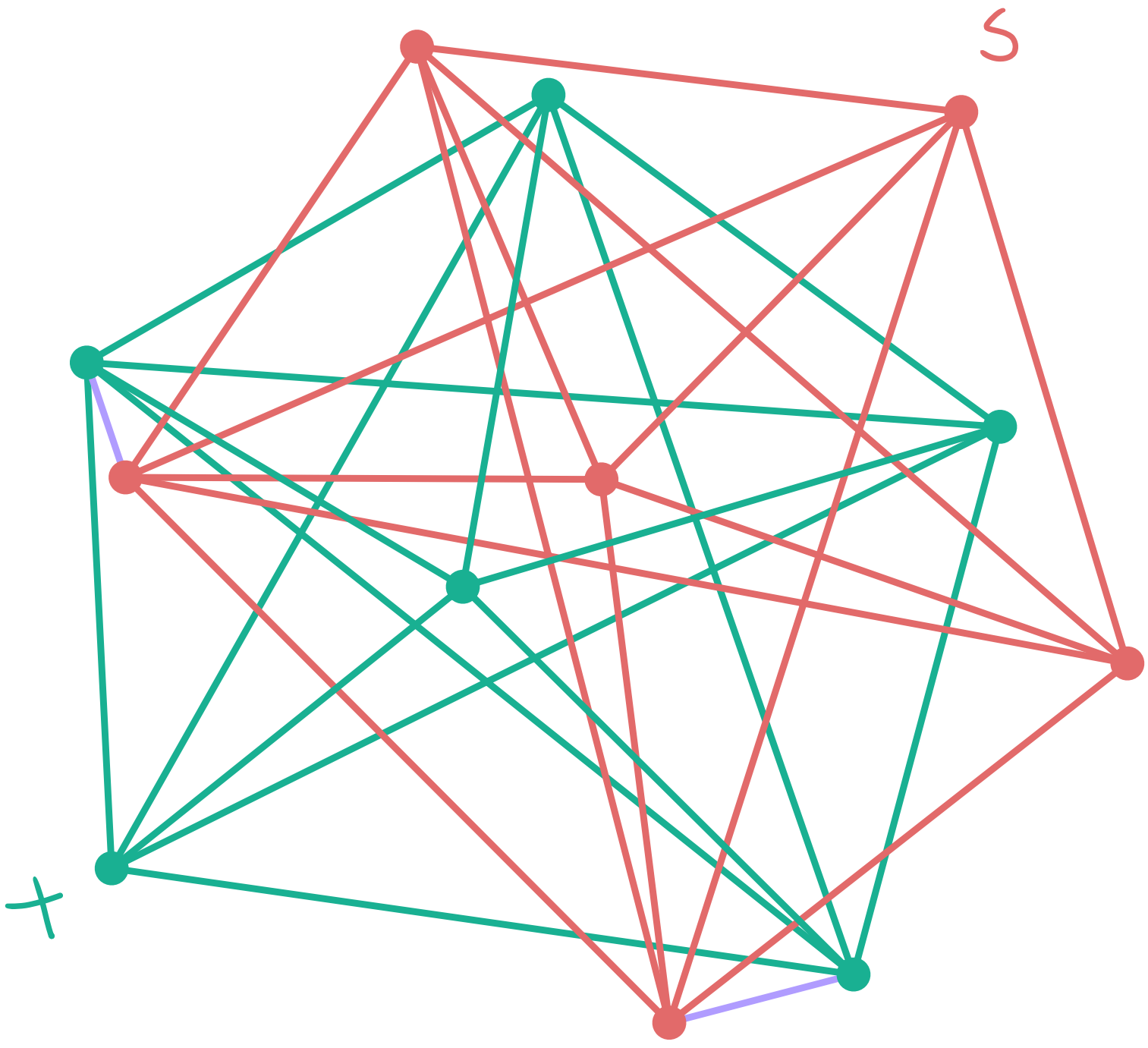
Global min-cut via max flow

1. for each pair $s, t \in V$
   a. compute min (s,t)-cut
2. return min (s,t)-cut over all s,t

Goal: disconnect graph by removing min # of edges




s
x

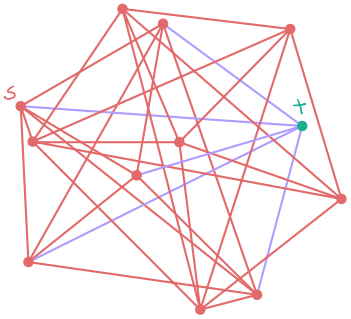

S
t

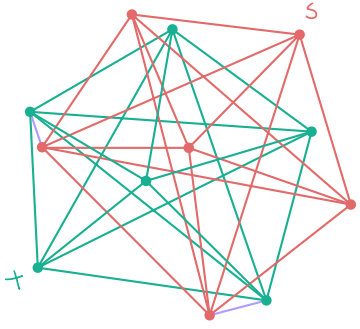
Global min-cut via max flow

① for each pair $s,t \in V$

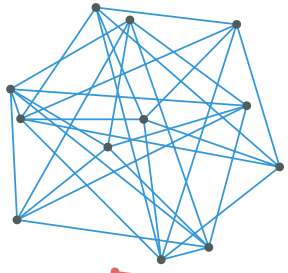
  ⓐ compute min (s,t)-cut

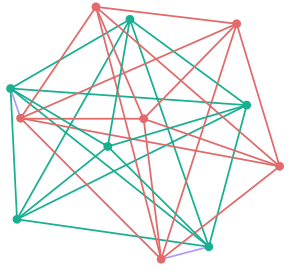
② return min (s,t)-cut over all s,t





Goal: disconnect graph by removing min # of edges





Running time

$$\binom{n}{2} \cdot MF(m,n)$$

Better?

Fix s, try all t

$$n \cdot MF(m,n)$$

"$MF(m,n)$" = time for max flow w/ m edges, n vertices

Augmenting paths	$O(\lambda m)$
Shortest Aug. Paths	$O(m^2 n)$
Fattest Aug. Paths	$O(m(m+n\log n)\log \lambda)$

Question:

Max # min-cuts in a graph?

$O(1)$
constant

$n^{O(1)}$
polynomial

$n^{\log^{O(1)} n}$
subexponential

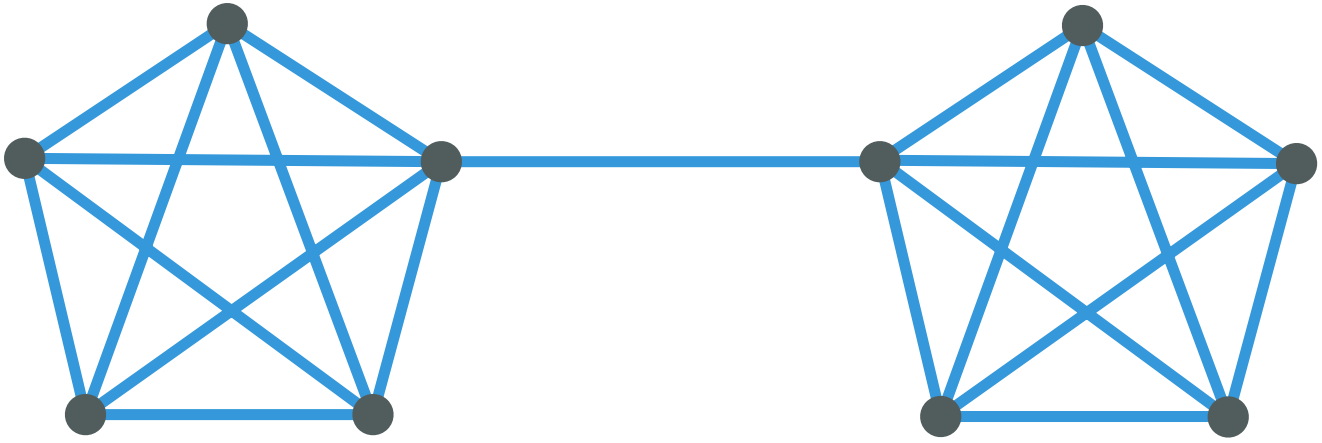
2^n
exponential

not voting

Randomly Sampling the Min-Cut

algorithm:

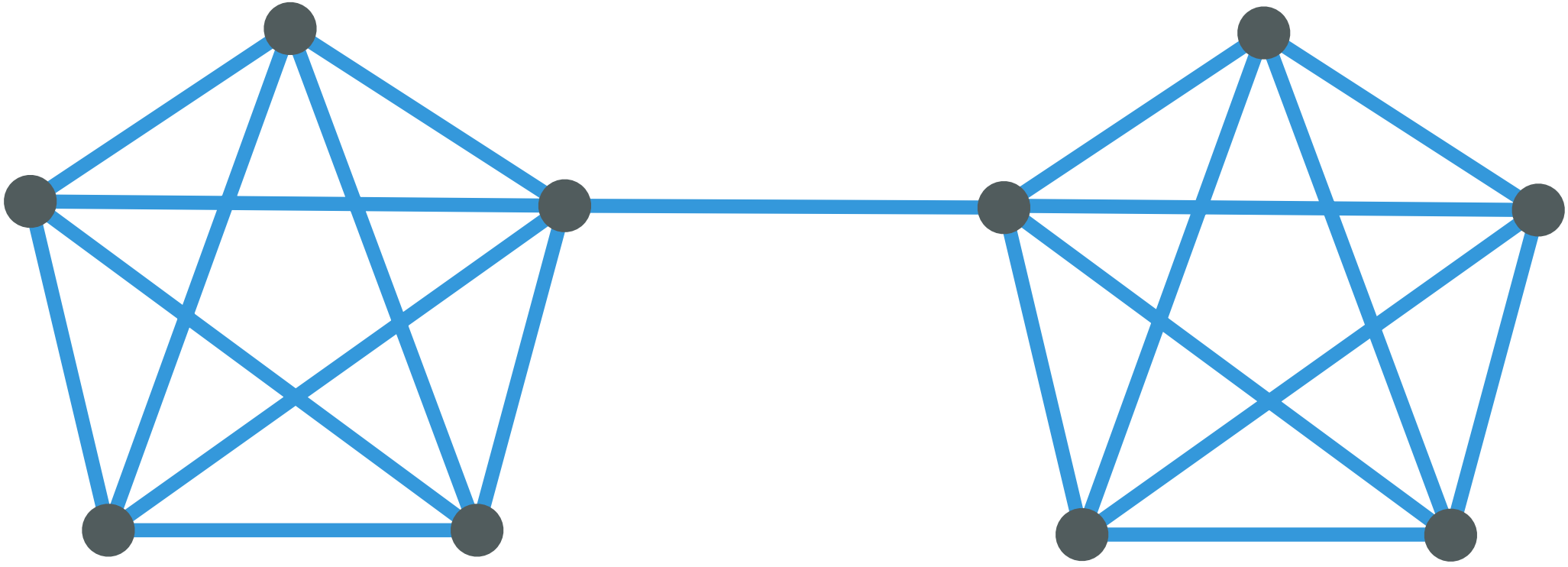
Repeatedly sample edges in proportion to their capacities until there is only one cut from which we have not yet sampled any edges. Return this cut.



or equivalently:

Independently assign every edge $e \in E$ a weight $w_e \in [0,1]$ uniformly at random. Build the minimum weight spanning tree T w/r/t w. Let e be the heaviest edge in T. Return the cut induced by the two components of $T - e$.

Independently assign every edge $e \in E$ a weight $w_e \in [0,1]$ uniformly at random. Build the minimum weight spanning tree T w/r/t w. Let e be the heaviest edge in T. Return the cut induced by the two components of $T - e$.



Randomized contractions

Input: $G=(V,E)$, weights $c \in \mathbb{R}^E_{>0}$ (capacities)

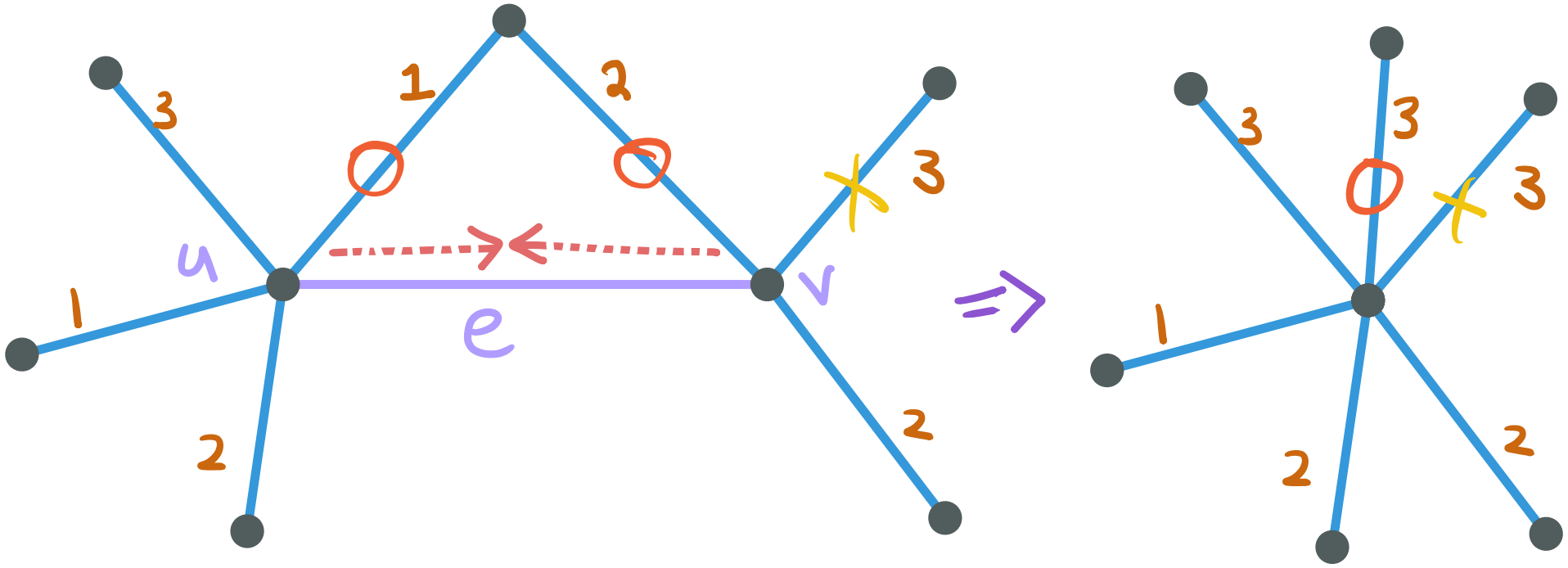
① while $|E|>1$:

Ⓐ sample $e \in E$ in proportion to c

Ⓑ "contract" e

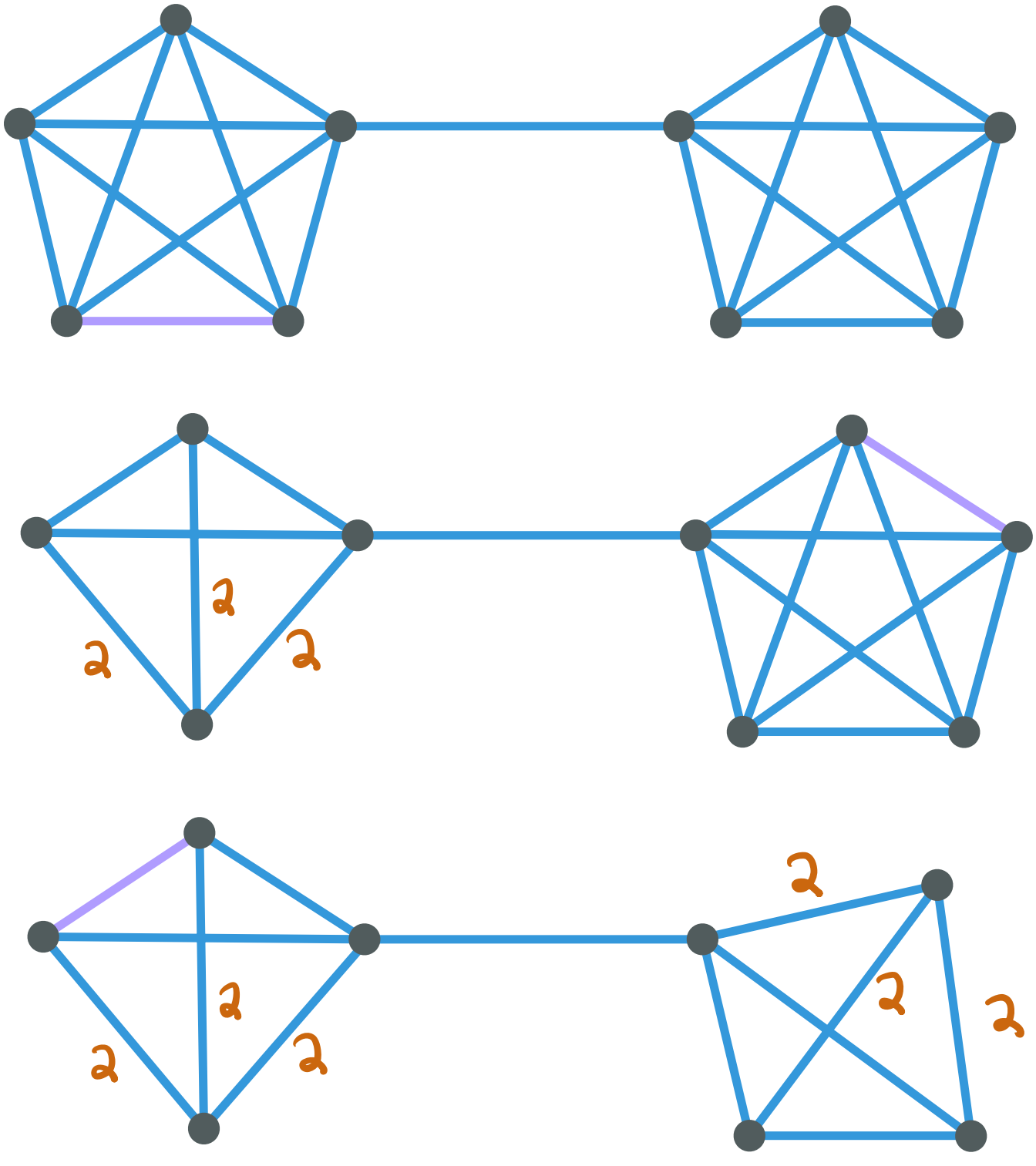
② return set of edges (in original graph) contracted to remaining edge

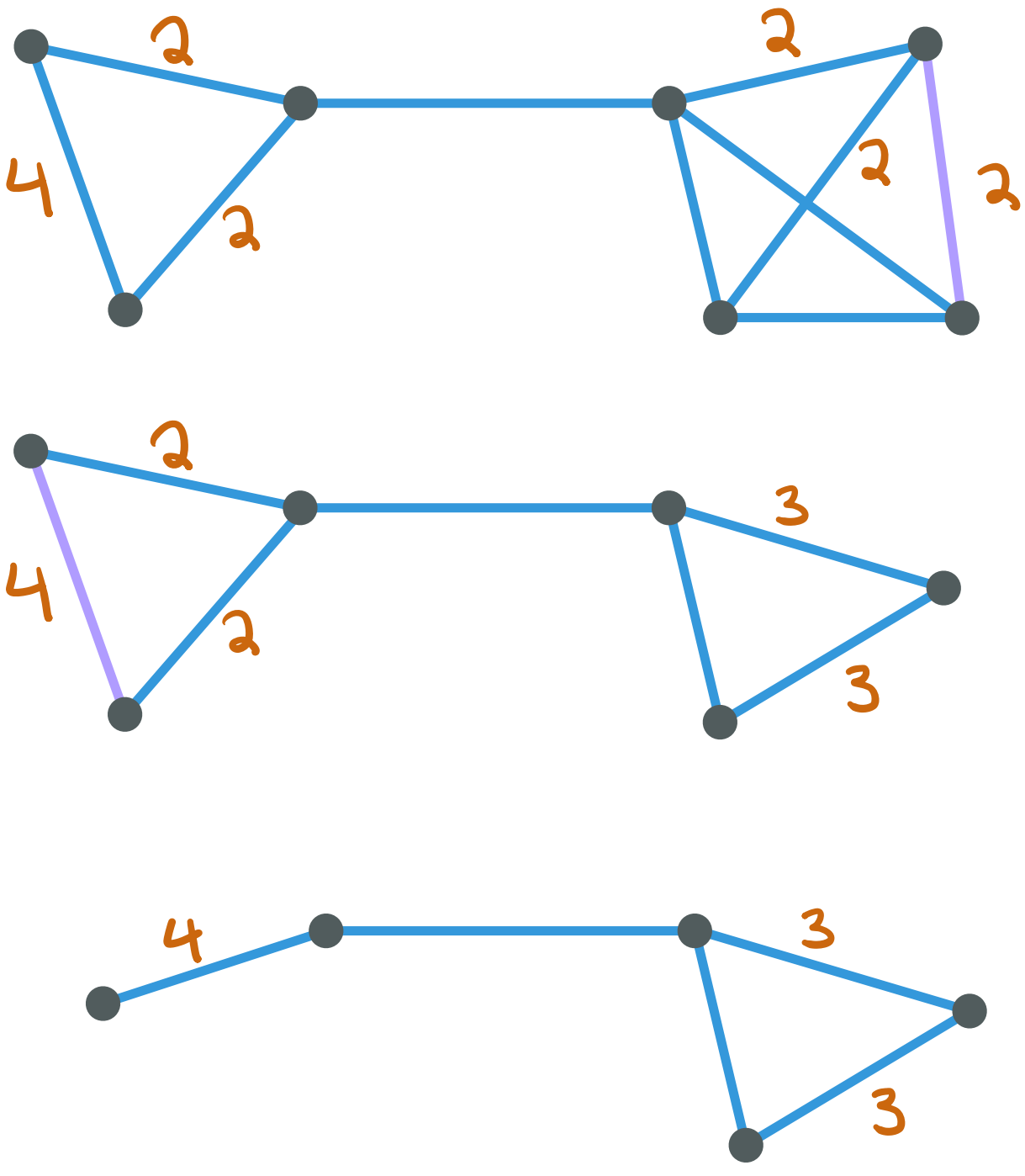
"contracting an edge"



combine endpoints into a single vertex

sum capacities of parallel edges


2
2
2
2
2
2
2
2
2


2
4
2
2
2
2
2
2
4
2
3
3
4
3
3

Randomized contractions

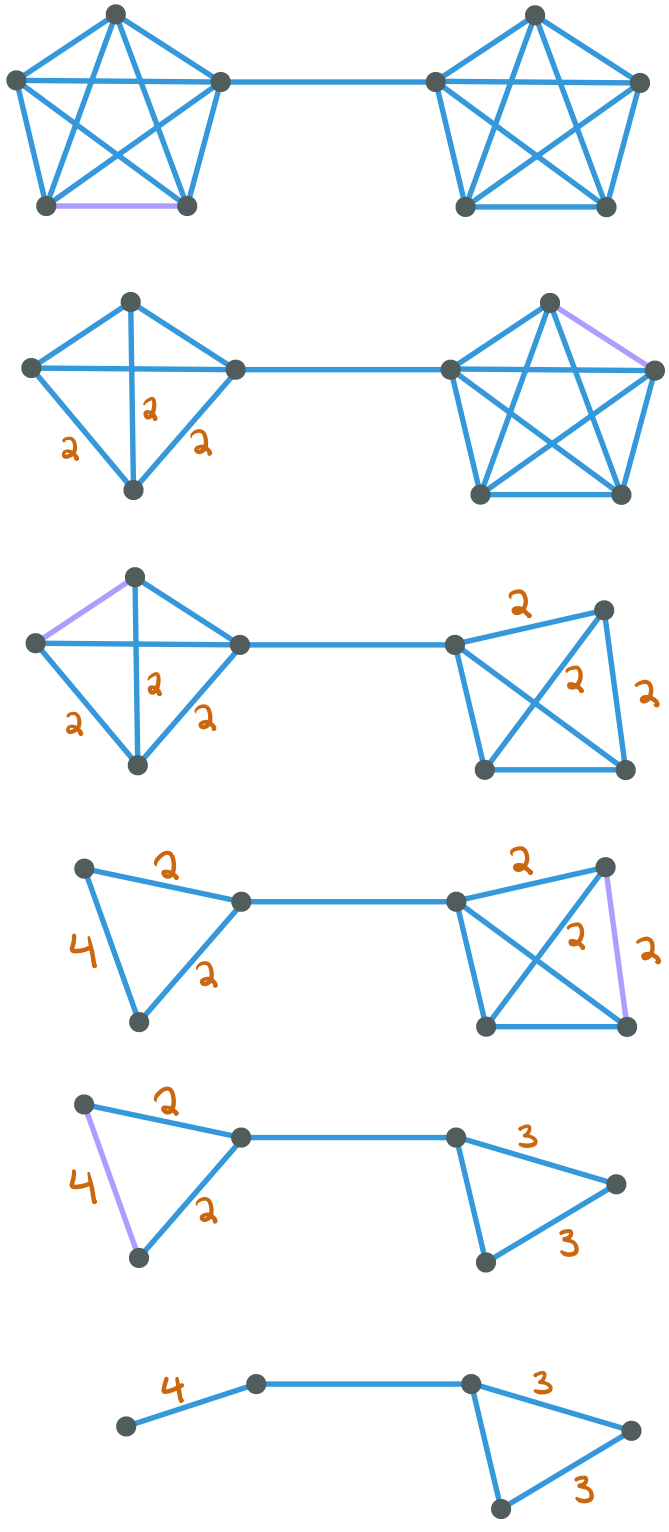
Input: $G=(V,E)$, weights $c \in \mathbb{R}^E_{>0}$ (capacities)

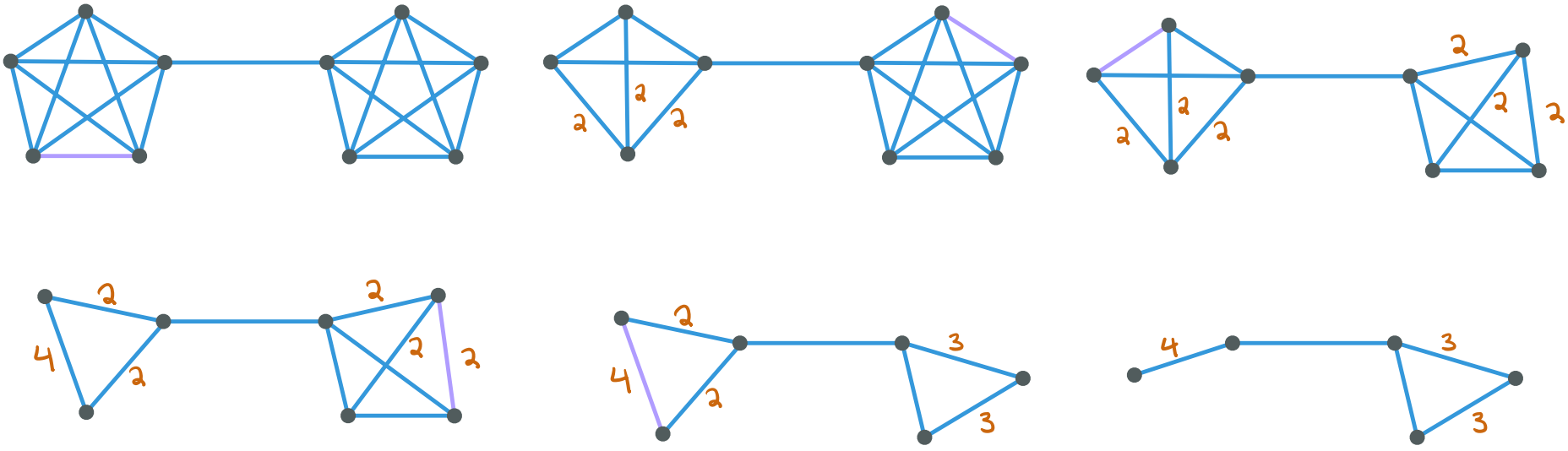
① while $|E| > 1$

  Ⓐ sample $e \in E$ in proportion to c

  Ⓑ "contract" e

② return set of edges (in original graph) contracted to remaining edge



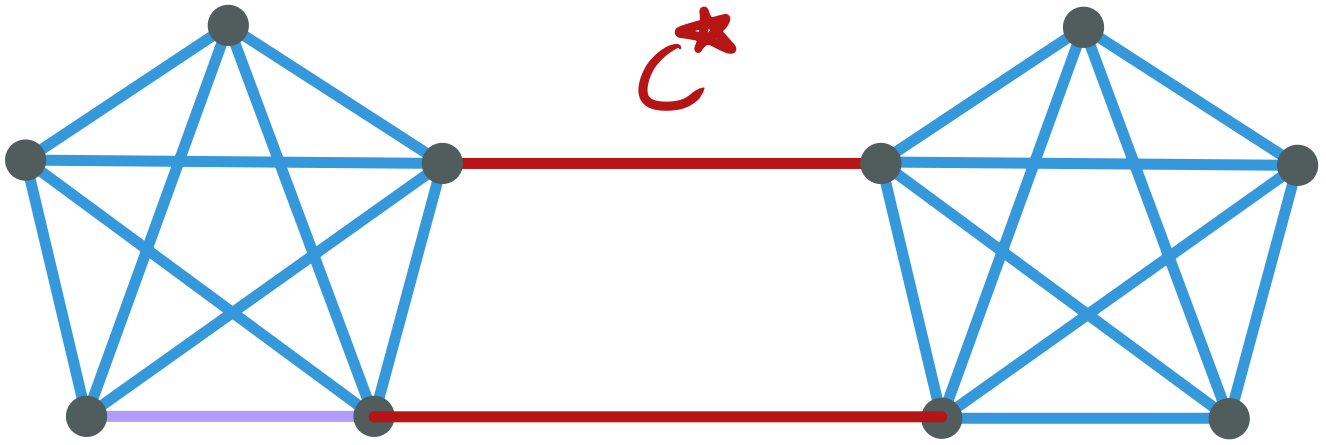


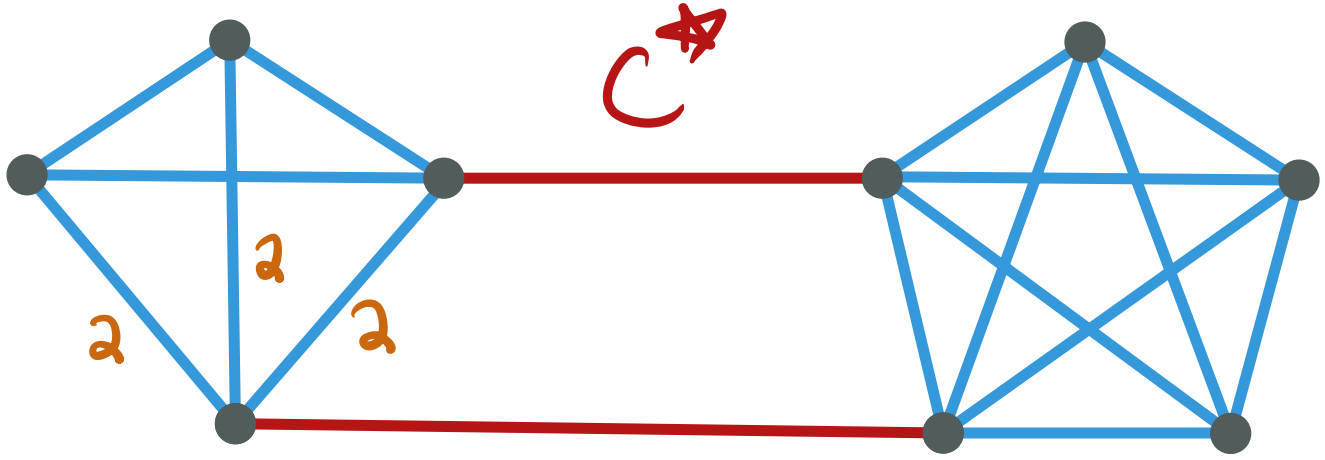
Analysis

Fix a min-cut C^*

P[output C^*]?

Obs. Contracting a non-min-cut edge preserves the min-cut.





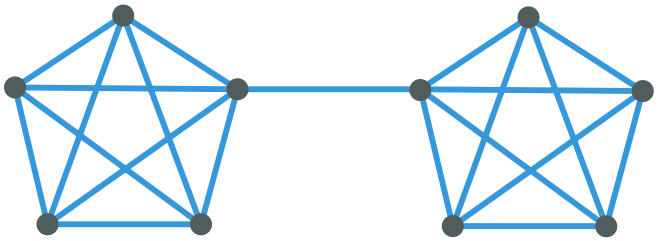
We output C^* iff every sampled edge is not in* C^*

* in the contracted image of C^*

Goal: analyze probability that each sampled edge avoids C^*

Let $\lambda = \sum_{e \in C^*} c(e)$ be capacity of min-cut

Lemma: $\sum_{e \in E} c(e) \geq \frac{\lambda n}{2}$



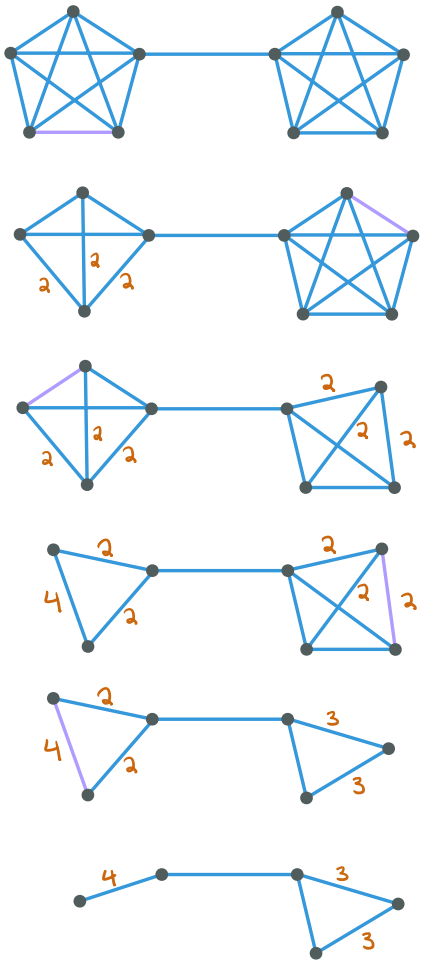
Lemma: $\sum_{e \in E} c(e) \geq \frac{\lambda n}{2}$

Lemma Let $e \in E$ be sampled in proportion to c. Then $\Pr[e \in C^*] \leq \frac{2}{n}$

Theorem $\Pr[C^* \text{ returned}] \geq \frac{1}{\binom{n}{2}}$

E_k = event that C^* not sampled after k iterations

want to show $P[E_{n-2}] \geq \frac{2}{\binom{n}{2}}$



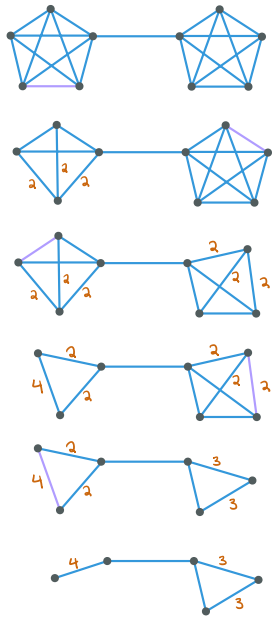
$$\Pr[E_{n-2}] = \Pr[E_1]\Pr[E_2|E_1]\cdots\Pr[E_{n-2}|E_{n-1}]$$

$$\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\cdots\left(1-\frac{2}{3}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\cdots\left(\frac{1}{3}\right)$$

$$= \frac{(n-2)!\,2!}{n!}$$

$$= \frac{1}{\binom{n}{2}}$$



Recall:

Lemma 2. let $e \in E$ be sampled in proportion to c.

Then $\Pr[e \in C^*] \leq \frac{2}{n}$

Question:

Max # min-cuts in a graph?

A

$O(1)$

constant

B

$n^{O(1)}$

polynomial

C

$n^{\log^{O(1)} n}$

subexponential

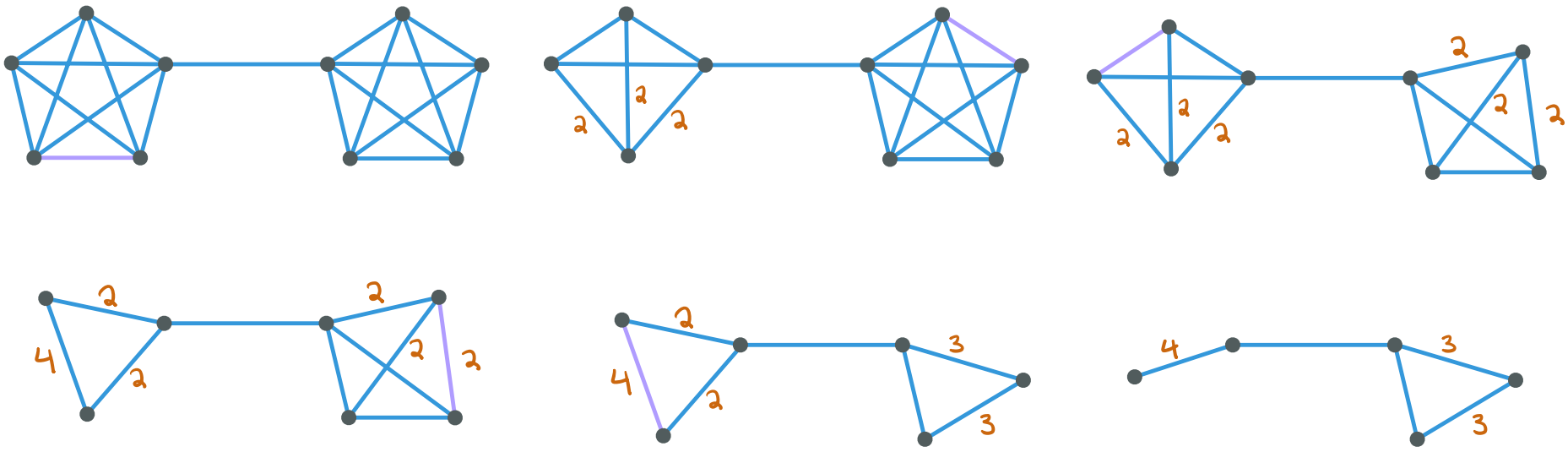
D

2^n

exponential

why?

any min-cut is output w/ prob $\geq \frac{1}{\binom{n}{2}}$



Random contractions algo takes $O(m)$ time

Rerun algo $O(n^2 \log(n))$ times to ensure success w/h/p

$\Rightarrow O(m n^2 \log(n))$ running time

Better?


PLAYER Red
BADGES 1
POKéDEX 10
TIME 1:24
YES
NO
Would you like to
SAVE the game?

Amplification by branching

Instead of restarting at the beginning, restart partway through

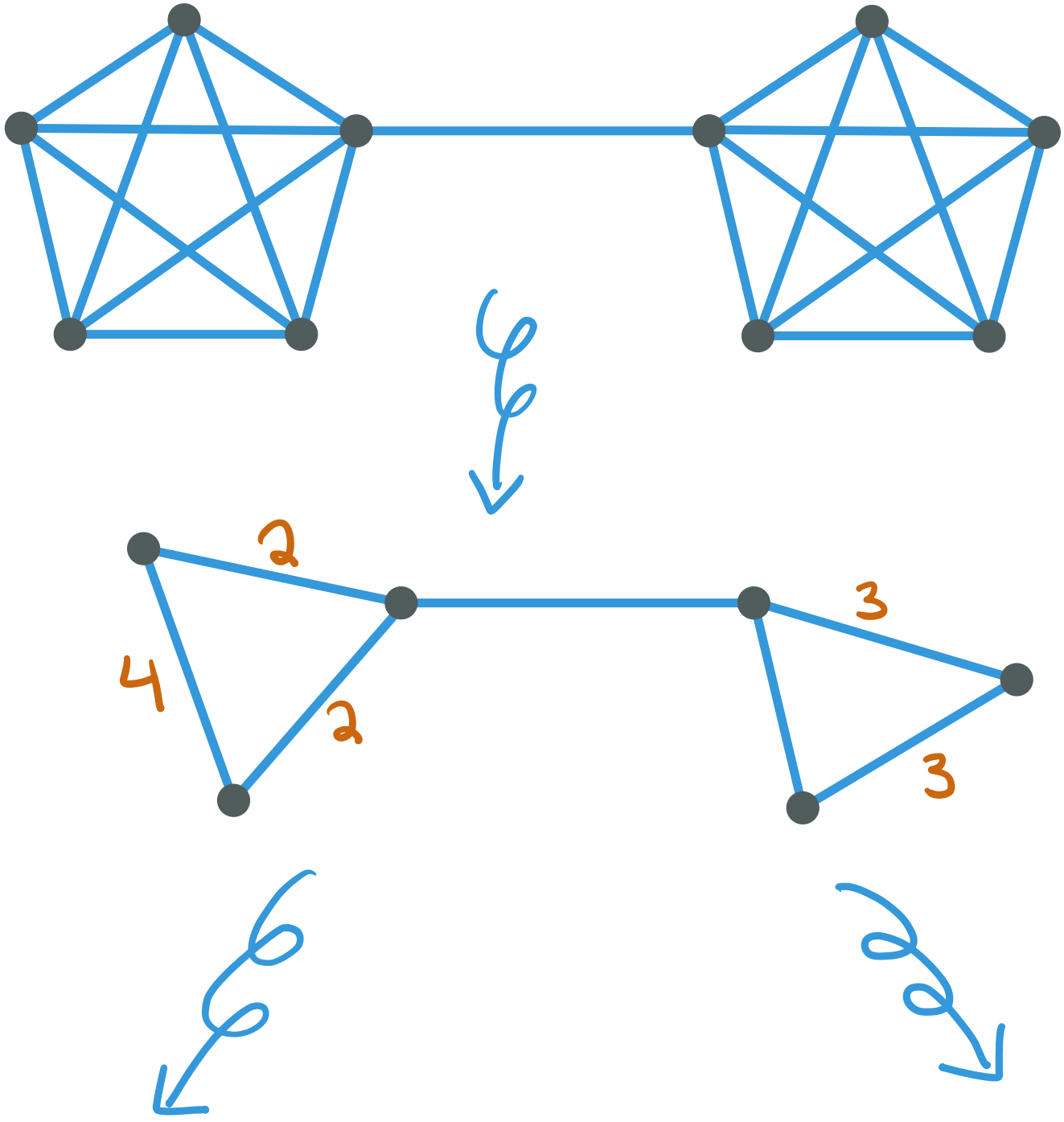
High-level algorithm

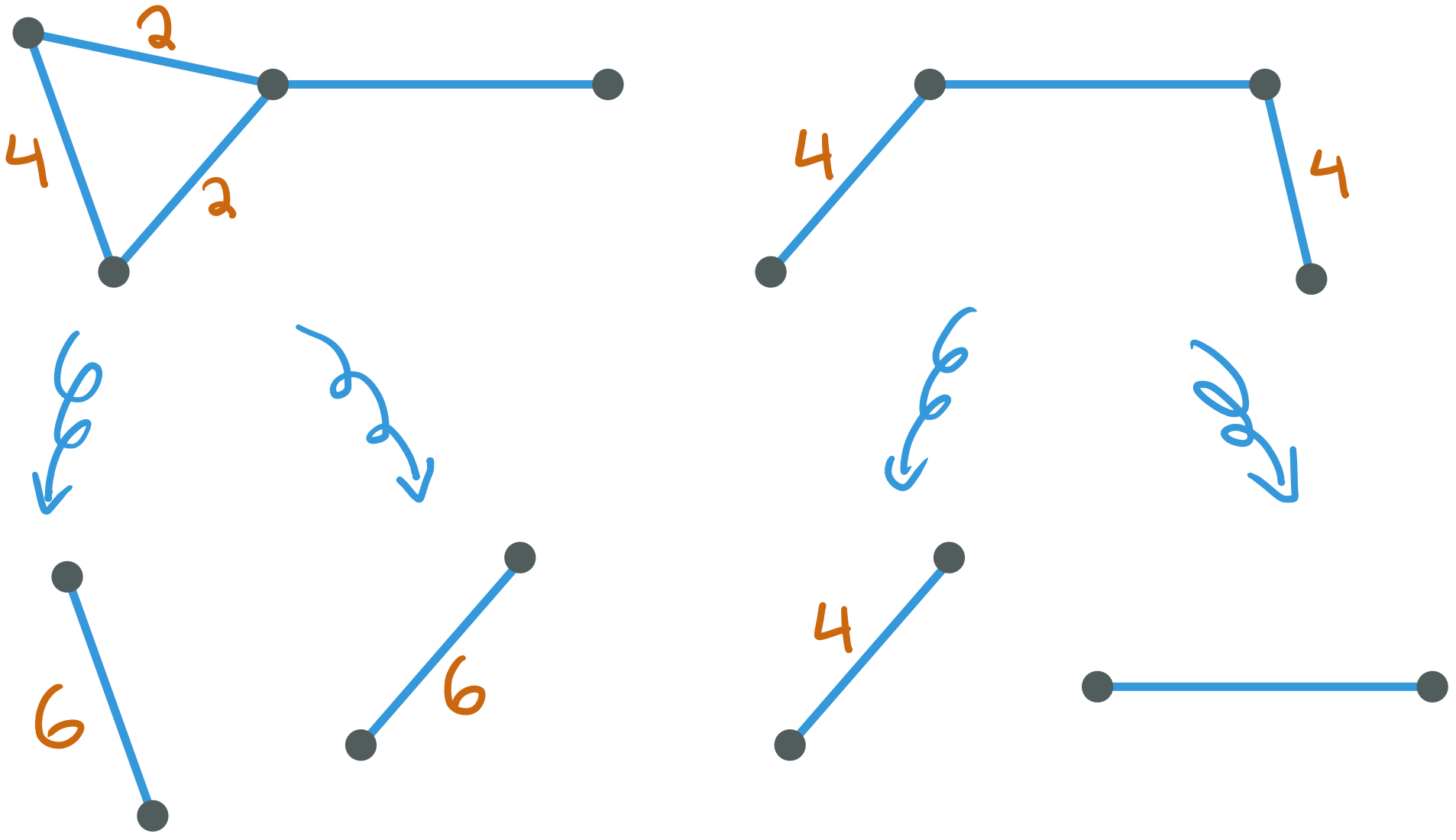
① randomly contract until prob. of ruining the min-cut is $\sim \frac{1}{2}$

② *Branch:*

make two copies of current graph

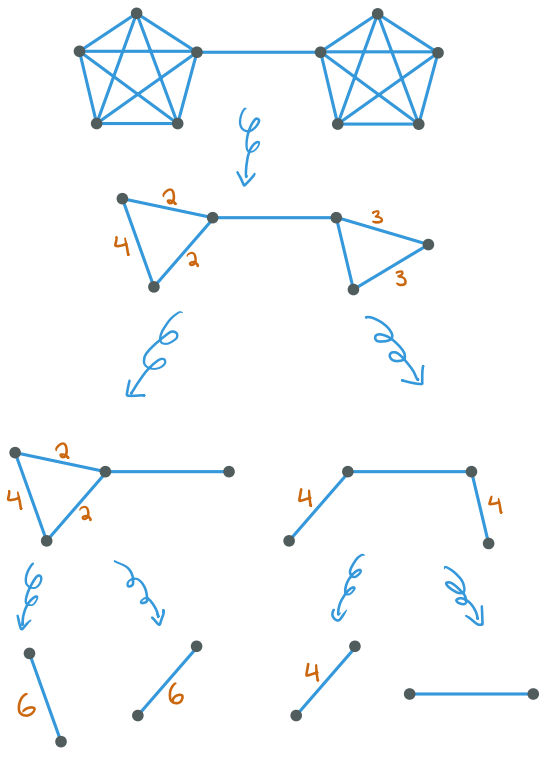
recurse on both copies


2
4
2
3
3


2
4
2
4
4
6
6
4

Amplification by branching

Instead of restarting at the beginning, restart partway through



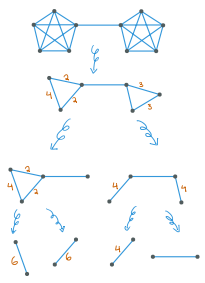
High-level algorithm

① randomly contract until prob. of ruining the min-cut is $\sim \frac{1}{2}$

② *Branch:*

make two copies of current graph

recurse on both copies



High-level algorithm

① randomly contract until prob. of ruining the min-cut is $\sim 1/2$

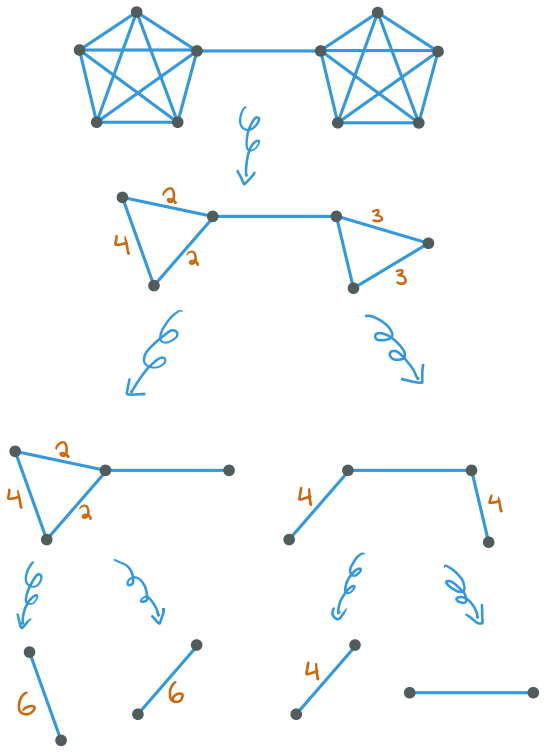
② *Branch:*

make two copies of current graph

recurse on both copies

branching-contractions($G = (V, E)$, c)

1. Let $n = |V|$. If $n \leq 3$ then compute the minimum cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e.
3. $C_1 \leftarrow$ branching-contractions(G, c)
4. $C_2 \leftarrow$ branching-contractions(G, c)
5. Uncontract and return the minimum of C_1 and C_2.



High-level algorithm

① randomly contract until prob. of ruining the min-cut is $\sim \frac{1}{2}$

② *Branch:*

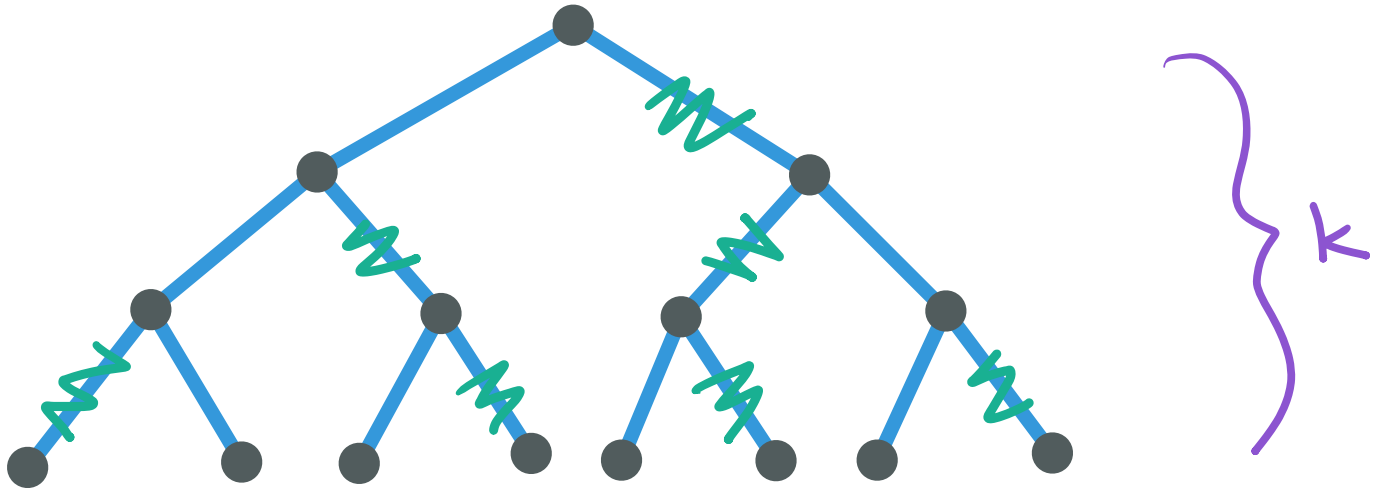
make two copies of current graph

recurse on both copies

Claim: branching-contractions returns min-cut w/ probability $\geq \frac{1}{O(\log n)}$

Claim: branching-contractions returns min-cut w/ prob. $\geq \frac{1}{\alpha(\log n)}$

Need the following fact (proved later)



let T = complete binary tree of height k,
w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

"Galton-Watson process"

Claim:

returns min-cut w/ prob. $\geq \frac{1}{O(\log n)}$

Proof:

High-level idea:

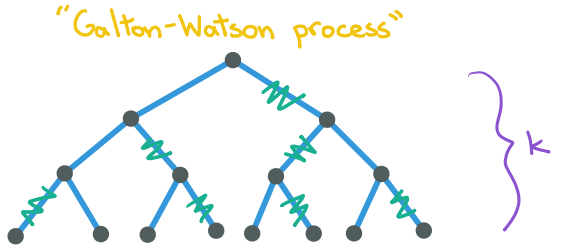
model recursive calls as tree

- nodes = subproblems
- edges = recursive subcalls
- delete parent edge of nodes/subproblems that fail before recursive call

Q: what is probability of edge being deleted

branching-contractions$(G=(V,E),\ c)$

1. Let $n = |V|$. If $n \leq 3$ then compute the min-cut by brute force.
2. Until $|V| = \lceil \frac{n}{\sqrt{2}} \rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ branching-contractions(G, c).
4. $C_2 \leftarrow$ branching-contractions(G, c).
5. Uncontract and return the minimum of C_1 and C_2.



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

Fix $C^* = $ min-cut.

suppose we contract $n-k$ edges

Recall

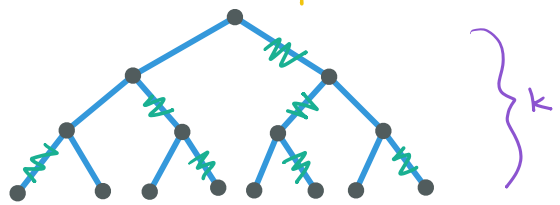
E_k = event that C^* not sampled after k iterations

$\Pr[E_{i+1} | E_i] = 1 - \frac{2}{n-i}$

branching-contractions$(G = (V, E),\ c)$

1. Let $n = |V|$. If $n \leq 3$ then compute the min-cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ branching-contractions(G, c).
4. $C_2 \leftarrow$ branching-contractions(G, c).
5. Uncontract and return the minimum of C_1 and C_2.

"Galton-Watson process"



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

$$\Pr[E_{n-k}] = \Pr[E_1]\Pr[E_2|E_1] \cdots \Pr[E_{n-k}|E_{n-k-1}]$$

$$= \left(\frac{n-2}{n}\right)\cdot\left(\frac{n-3}{n-1}\right) \cdots \left(\frac{k-1}{k+1}\right)$$

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

$$k = \lceil n/\sqrt{2} \rceil + 1 \Rightarrow P[E_{n-k}] \geq \frac{1}{2}$$

Claim:

returns min-cut w/ prob. $\geq \frac{1}{O(\log n)}$

Proof:

High-level idea:

model recursive calls as tree

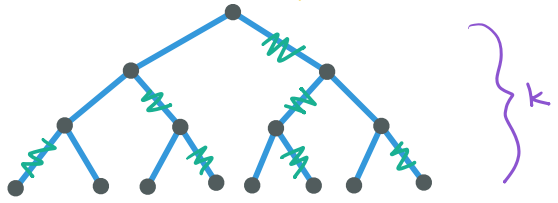
- nodes = subproblems
- edges = recursive subcalls
- delete parent edge of nodes/subproblems that fail before recursive call

} Prob. $\leq 1/2$

branching-contractions$(G = (V, E),\ c)$

1. Let $n = |V|$. If $n \leq 3$ then compute the min-cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ branching-contractions(G, c).
4. $C_2 \leftarrow$ branching-contractions(G, c).
5. Uncontract and return the minimum of C_1 and C_2.

"Galton-Watson process"



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\leq 1/2$

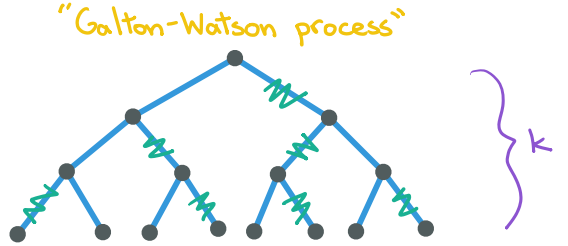
Fact: P[some leaf connected to root] $\geq \frac{1}{k}$

model recursive calls as tree

- nodes = subproblems
- edges = recursive subcalls
- delete parent edge of nodes/subproblems that fail before recursive call } Prob. $\leq 1/2$

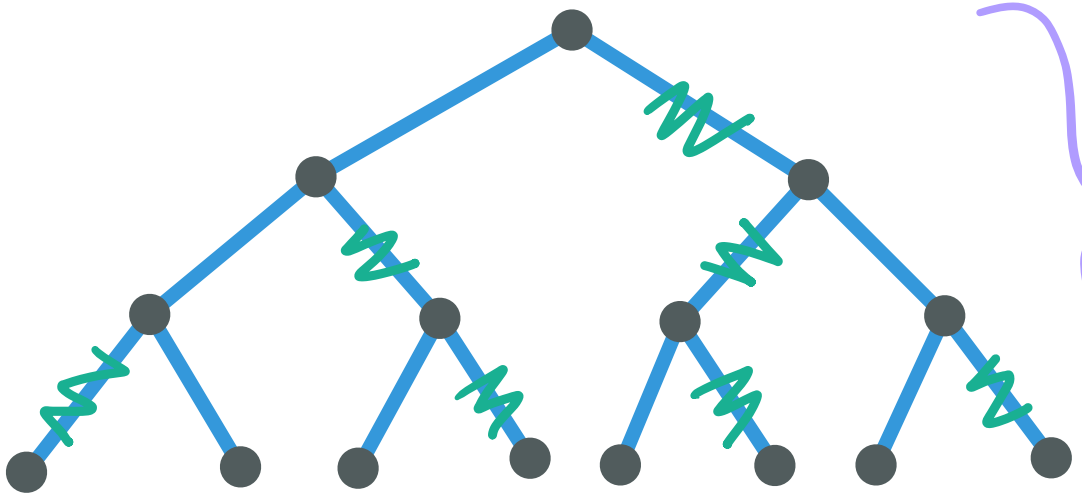
branching-contractions($G = (V, E)$, c)

1. Let $n = |V|$. If $n \leq 3$ then compute the min-cut by brute force.
2. Until $|V| = \lceil \frac{n}{\sqrt{2}} \rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ branching-contractions(G, c).
4. $C_2 \leftarrow$ branching-contractions(G, c).
5. Uncontract and return the minimum of C_1 and C_2.



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\leq 1/2$

Fact: P[some leaf connected to root] $\geq \frac{1}{k}$



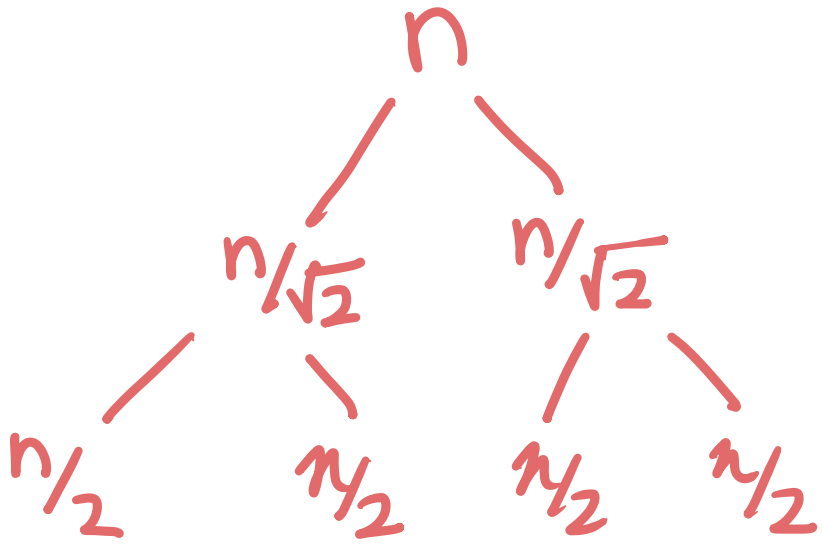
height = $O(\log n)$

algo correct <=> root-to-leaf path

Fact => prob success $\geq \Omega(1/\log n)$

Running time

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



n^2

$2(n/\sqrt{2})^2$

$4(n/2)^2 = O(n^2)$

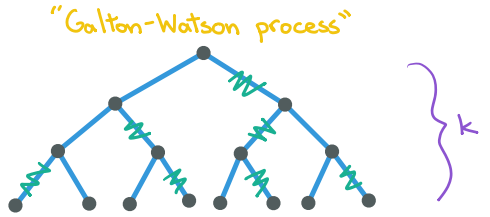
$\vdots$

$+ \; 2^i(n/2^{i/2})^2 = O(n^2)$

$O(n^2 \log n)$

`branching-contractions`$(G=(V,E),\ c)$

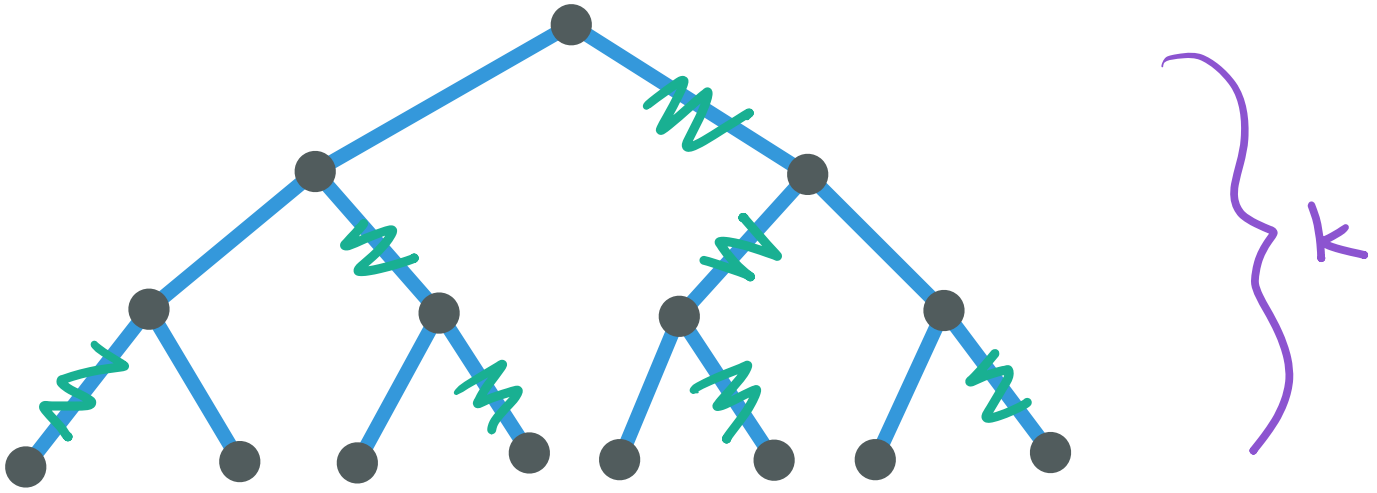
1. Let $n = |V|$. If $n \le 3$ then compute the min-cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ `branching-contractions`(G, c).
4. $C_2 \leftarrow$ `branching-contractions`(G, c).
5. Uncontract and return the minimum of C_1 and C_2.



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\le 1/2$

Fact: $P[\text{some leaf connected to root}] \ge \frac{1}{k}$

Now we prove the fact:



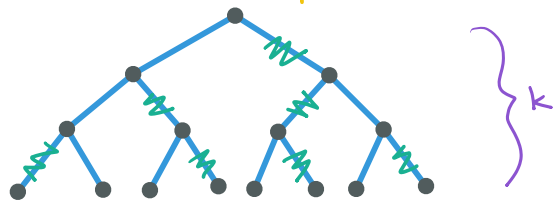
let T = complete binary tree of height k,
w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

"Galton-Watson process"

$p_i \stackrel{\text{def}}{=}$ prob that a particular node at level i has path to a subleaf

"Galton-Watson process"



let T = complete binary tree of height k, w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

$$p_0 = 1$$

$$p_{i+1} = 1 - \left(\frac{1}{2} + \frac{1}{2}(1-p_i)\right)^2$$

$$= 1 - \left(1 - \frac{p_i}{2}\right)^2 = p_i\left(1 - p_i/4\right)$$

$p_0 = 1$

$p_1 = \frac{3}{4}$

$p_2 = \frac{39}{64} \geq \frac{1}{2}$

$$p_{i+1} = p_i(1 - p_i/4)$$

claim: $p_k \geq 1/k \quad \forall k$

$p_0 = 1, \quad p_1 = 3/4, \quad p_2 \geq \frac{39}{64}$

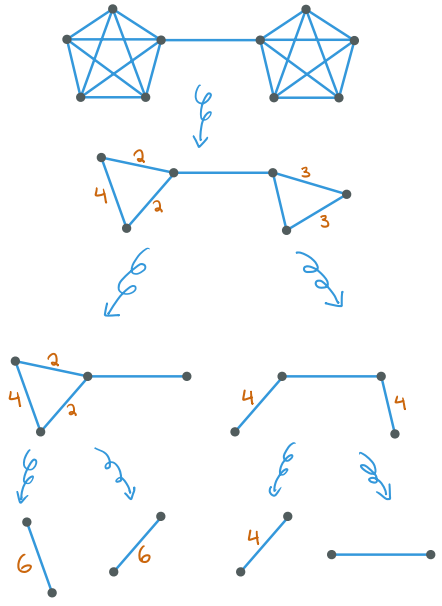
let $f(x) = x(1 - x/4)$

f is increasing for $x \leq 2$

$$p_{k+1} \leq \frac{1}{k}\left(1 - \frac{1}{4k}\right) = \frac{1}{k} - \frac{1}{4k^2} \overset{*}{\geq} \frac{1}{k+1}$$

$$(*)\ \frac{1}{k} - \frac{1}{k+1} = \frac{1}{k^2+1} \geq \frac{1}{4k^2}$$

All put together:



High-level algorithm

① randomly contract until prob. of ruining the min-cut is $\sim 1/2$

② *Branch:*

make two copies of current graph

recurse on both copies

$O(n^2 \log n)$ time

succeeds w/ probability $\Omega(\frac{1}{\log n})$

repeat $O(\log n)$ times to get constant prob.

Chapter 4

Sampling edges

This chapter is about applying random sampling while trying to preserve *combinatorial* structures, like graphs.

Consider, for example, the (s,t)-flow problem. We have as input a graph $G = (V,E)$, positive edge capacities $c : E \to \mathbb{R}_{>0}$, and two vertices s and t. We want to route the maximum amount of flow from s to t. Can we sample a small subgraph of G while preserving the value of the maximum s to t flow? If so, then we could run a max flow algorithm on the sampled subgraph and get faster overall running times. Try to imagine sampling edges from a graph in the interest of flow. Flow has a combinatorial aspect that would appear much more delicate than, say, heavy hitter estimates. For example, missing a single important edge when sampling can disrupt a polynomial number of paths used by the maximum flow. Nonetheless, we will see that for *undirected* graphs, the surprising answer is yes: one can indeed subsample the important parts of an undirected graph and preserve the maximum flow.

4.1 Minimum cut

Recall the *minimum cut* problem in undirected graphs. The input consists of a connected, undirected graph $G = (V,E)$ with positive edge capacities $c : E \to \mathbb{R}_{>0}$. A *cut* is a set of edges $C \subseteq E$ whose removal disconnects the graph. The goal is to

$$\text{minimize } \sum_{e \in C} c(e) \text{ over all cuts } C \subset E. \tag{4.1}$$

This problem is polynomial time solvable. Whatever the optimum cut is, it must be a minimum (s,t)-cut for some pair of vertices s and t. Thus to find the global minimum, one can guess s and t by looping over V, and compute the minimum $\{s,t\}$-cut for each choice of s and t. (Better yet: fix s, and loop over all t.) Previously, we also studied an algorithm due to [NI92b] that runs in $O(mn)$ time.

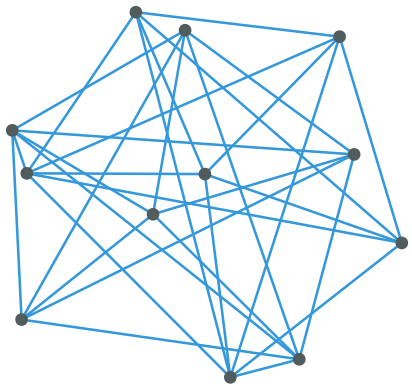


Figure 4.1: What is the minimum cut in this graph?

For a set of vertices S, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. $\delta(S)$ is called the cut *induced by* S. The induced cuts are also the inclusionwise minimal cuts, and it suffices to consider only the induced cuts when solving (4.1).

We will study a subtle algorithm discovered by Karger [Kar93], that has been influential beyond the minimum cut problem. Consider the following description of Karger's algorithm.

> *Repeatedly sample edges in proportion to their capacities until there is only one cut from which we have not yet sampled any edges. Return this cut.*

This algorithm is clearly ridiculous. For the unweighted setting, the above algorithm is equivalent to the following, equally absurd approach (see exercise 4.2).

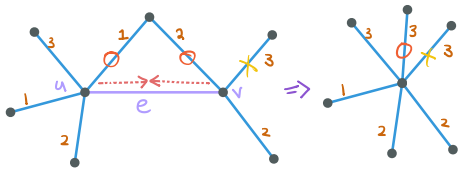
> *Independently assign every edge $e \in E$ a weight $w_e \in [0,1]$ uniformly at random. Build the minimum weight spanning tree T w/r/t w. Let e be the heaviest edge in T. Return the cut induced by the two components of $T - e$.*

Compare the two approaches above. Of course we know how to compute the minimum spanning tree; among other approaches, we can repeatedly add the smallest weight edge to T that does not create a cycle. On the other hand, in the first approach, it might appear difficult to keep track of which cuts we have and have not sampled from, being that there are so many cuts. This can be addressed by *contracting the graph*.

`random-contractions`$(G = (V, E), c)$

1. While $|E| > 1$
   A. Sample $e \sim c$
   B. $G \leftarrow G/e$, $c \leftarrow c/e$
2. Let $E = \{e\}$
3. Return the edges in the original graph that contracted to e

Figure 4.2: A randomized minimum cut algorithm due to Karger [Kar93].



Suppose we sample an edge $e = \{s, t\}$. Then we know that any cut $\delta(S)$, where $s \in S$ and $t \notin S$, has now been sampled from. Thus we can safely *contract* e; replacing s and t with a single vertex u that has the sum[1] of edges incident to s and t. Note that contracting e will only effect cuts that contain e.

Now imagine we contract edges as we sample them. Eventually there are only two vertices left in the contracted graph, which represent two connected components in the input graph. These components induce the only cut we have not yet sampled from, and this is the cut that we return.

Pseudocode for the contraction algorithm is given in fig. 4.2. Here, for an edge $e \in G$, we let $G/e = (V/e, E/e)$ denote the graph obtained by contracting e, and we let $c/e : E/e \to \mathbb{R}_{>0}$ denote the corresponding capacities. Figure 4.3 sketches a few iterations of the algorithm applied to a barbell graph.

The intuition behind `random-contractions` is as follows. Here we describe the intuition for unweighted graphs for simplicity. (The intuition is the same for weighted graphs, except replacing "many edges" with "large capacity", etc.) Suppose we have an unweighted graph $G = (V, E)$, and let $C \subset E$ be the minimum cut. Since C is the minimum cut – keyword minimum – there are presumably very few edges in C. If we

[1] More precisely, for every edge f of the form $\{s, z\}$ or $\{t, z\}$, we create a new edge $\{u, z\}$ with the same capacity. If s and t both have edges to the same vertex z, we can either create two edges from u to z with the appropriate capacities, or make a single edge from u to v with the same capacity. We remove s and t and its incident edges from the graph, replacing them with u and the newly created edges incident to u.

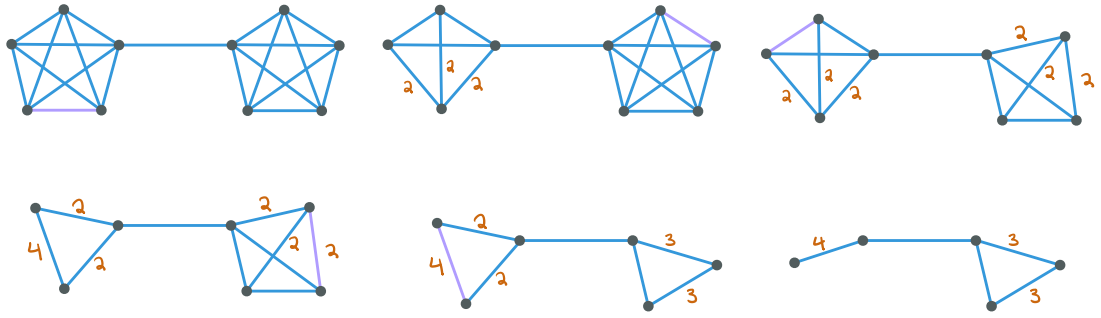


Figure 4.3: A few iterations (from left to right, top to bottom) of the `random-contractions` algorithm applied to a barbell graph.

randomly sample an edge $e \in E$, then hopefully $e \notin C$. If C "survives" this round, then we have all made some progress because there is one less vertex in the graph after contracting e. In the next round, C is still the minimum cut, so the high-level logic from the first round still holds. Thus we can repeatedly sample edges and preserve the hope that we avoid C.

The above argument hinges on *how much smaller* C is than all of E. If we can argue that C is always a small fraction of E, then that gives hope that C survives to the end. On the other hand, if C is even a small constant fraction of G, we will probably sample from C after a constant number of rounds. Observe also that over time, C becomes a larger and larger fraction of E, as we contract and remove edges outside of C.

The key observation is that *every vertex v induces a cut $\delta(v)$, which must have at least as many edges as C. Thus the minimum cut is at most the minimum degree in the graph.* In turn, since the number of edges in E is the sum of degrees (divided by 2), *the minimum cut C is at most a $2/n$ fraction of the total number of edges*! This observation holds initially in the input graph and thereafter in the contracted graphs, although n decreases by 1 in each iteration.

On the first iteration, C has at most a $2/n$ chance of being hit. On the second interation, assuming C surivived the first iteration, C has (at most) a $2/(n-1)$ chance of being hit. Continuing in this fashion, assuming C surivived the first $i-1$ iterations, C has a $2/(n-i+1)$ change of being hit in the ith iteration. If one combines these problems, one discovers that C has a $\geq 1/\binom{n}{2}$ chance of surviving all $n-1$ rounds We can repeat the experiment $\binom{n}{2} = O(n^2)$ (a polynomial!) number of times to find the

minimum cut with constant probability, and $O(n^2 \log n)$ times to find the minimum cut with high probability.

In the sequel, we formalize the the above argument, as well as extend it to positive capacities. For ease of notation, for a set of edges $C \subset E$, we denote the sum of capacities over C by

$$\sum_{e \in C} c(e) \stackrel{\text{def}}{=} \sum_{e \in C} c(e).$$

Lemma 4.1. *Let $C^\star$ be the minimum cut in (G, c), and suppose $e \notin C$. Then $C^\star$ is (or maps to) to the minimum cut in the contracted graph $(G/e, c/e)$.*

Proof sketch. Direct inspection. □

Lemma 4.2. $\sum_{e \in E} c(e) \geq \frac{\lambda n}{2}$.

Proof. Every vertex v has weighted degree $\sum_{e \in \delta(v)} c(e) \geq \lambda$ since $\delta(v)$ is a cut. Thus

$$\sum_{e \in E} c(e) = \frac{\sum_v \sum_{e \in \delta(v)} c(e)}{2} \geq \frac{\lambda n}{2}.$$

□

Lemma 4.3. *Let $e \sim c$. Then $\mathbf{P}[e \in C^\star] \leq \frac{2}{n}$.*

Proof. We have $\mathbf{P}[e \in C^\star] = \frac{\sum_{e \in C^\star} c(e)}{\sum_{e \in E} c(e)} \stackrel{\text{(a)}}{\leq} \frac{2}{n}$ by (a) lemma 4.2. □

Lemma 4.4. *Let $C^\star$ be a minimum cut. With probability $\geq 1/\binom{n}{2}$,* `random-contractions` *returns $C^\star$.*

Proof. For $k \in \mathbb{Z}_{\geq 0}$, let E_k be the event that we have not sampled $C^\star$ after k iterations. Initially, $\mathbf{P}[E_0] = 1$, and we want to show that $\mathbf{P}[E_{n-2}] \geq 1/\binom{n}{2}$. By lemma 4.3, we have

$$\mathbf{P}[E_k \mid E_{k-1}] \geq 1 - \tfrac{2}{n-(k-1)} \text{ for each } k \in [n].$$

The probability of succeeding (event E_{n-2}) is at least

$$\begin{aligned}\mathbf{P}[E_{n-2}] &= \prod_{k=1}^{n-2} \mathbf{P}[E_k \mid E_{k-1}] \geq \prod_{i=3}^{n} \left(1 - \frac{2}{i}\right) \\ &= \prod_{i=3}^{n} \frac{i-2}{i} = \frac{(n-2)!2}{n!} = \frac{1}{\binom{n}{2}}.\end{aligned}$$

□

Thus with probability about $1/n^2$, the random contraction algorithm returns the minimum cut. To find the minimum cut with constant probability, we rerun the algorithm $O(n^2)$ time and return the best cut. To find the minimum cut with *high probability*, we rerun the algorithm $O(n^2 \log n)$ times.

The nice thing about repetition is that we can run the randomized trials *in parallel*. Moreover, a single instance of the contraction algorithm (via its connection to minimum spanning trees) can be made to run in $\text{polylog}(n)$ time with polynomially many processors. Thus one obtains a randomized parallel algorithm for minimum cut.

Corollary 4.5. *A randomized minimum cut can be computed in parallel in polylogarithmic time with a polynomial number of processors.*

That said, `random-contractions` is not just an algorithm. It is also a surprising structural observation about the number of minimum cuts in an undirected graph. In the above algorithm, any fixed minimum cut is returned with probability $1/\binom{n}{2}$. This implies that there are at most $\binom{n}{2}$ minimum cuts in the graph!

Corollary 4.6. *There are at most $\binom{n}{2}$ minimum cuts in a graph.*

4.2 Amplification by branching

The `randomized-contraction` algorithm preserves a fixed minimum cut with probability at least $1/\binom{n}{2}$. One can amplify this algorithm directly by running it independently $O(n^2 \log n)$ and outputting the minimum over all the trials. With high probability, the minimum cut lies in one of these $O(n^2 \log n)$ cuts.

Karger and Stein [KS96] reduced the probability of failure more efficiently by an amplification process called *branching*. To motivate the branching technique, recall from **??** that the probability of failure increases as the number of remaining vertices decreases. Instead of restarting the entire algorithm from the beginning again and again, one might restart the algorithm from some relatively confident point partway through. One might further apply this strategy recursively.

Karger and Stein [KS96] proposed randomly contract edges for a fixed number of iterations so that the probability of avoiding the min-cut is still at least $1/2$, and then "branching"; i.e., running two independent processes that continue from this point. The branching is recursive: each of the two independent trials also continue for a relatively safe number of iterations before branching again. This refined amplification process, it is shown below, is much more efficient than repeated independent trials

`branching-contractions`$(G = (V, E),\ c)$

1. Let $n = |V|$. If $n \leq 3$ then compute the minimum cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e.
3. $C_1 \leftarrow$ `branching-contractions`(G, c)
4. $C_2 \leftarrow$ `branching-contractions`(G, c)
5. Uncontract and return the minimum of C_1 and C_2.

Figure 4.4: A randomized minimum cut algorithm that amplifies the `random-contractions` algorithm by branching, due to Karger and Stein [KS96].

of `random-contractions`. The amplification technique is interesting in its own right and extends past this particular problem.

A sketch of the algorithm, which we call `branching-contractions`, is given in fig. 4.4.

Lemma 4.7. *Let $C^\star$, and suppose we contract $n - k$ edges sequentially at random. The probability that none of theses edges samples the minimum cut is at least $\frac{k(k-1)}{n(n-1)}$.*

Proof. For $i \in \mathbb{N}$, let E_i be the probability that we have not sampled $C^\star$ after i iterations. We ware interested in $\mathbf{P}[E_{n-k}]$. We have

$$\mathbf{P}[E_{n-k}] = \prod_{i=1}^{n-k} \mathbf{P}[E_i \mid E_{i-1}] \overset{\text{(a)}}{=} \prod_{i=k+1}^{n} \left(1 - \frac{2}{i}\right)$$
$$= \frac{\prod_{i=3}^{n} \frac{i-2}{i}}{\prod_{i=3}^{k} \frac{i-2}{i}} = \frac{\binom{k}{2}}{\binom{n}{2}} = \frac{k(k-1)}{n(n-1)}.$$

by (a) lemma 4.3. □

Theorem 4.8. `branching-contractions` *runs in $O(n^2 \log n)$, and returns a minimum cut with probability $\Omega(1/\log n)$.*

Proof. We first prove the running time, and then discuss the correctness. The running time is dominated by the recurrence

$$f(n) \leq 2f(\varphi(n)) + n^2 \text{ for } \varphi(n) = n/\sqrt{2} + c$$

for some constant $c > 0$. The recursion tree is drawn in fig. 4.5. For $i \in \mathbb{N}$, let $\varphi^i = \varphi \circ \varphi^{i-1}$ recursively apply φ i times (e.g., $\varphi^1 = \varphi$). Each problem at depth i has size at most

$$\varphi^i(n) = \left(\frac{1}{\sqrt{2}}\right)^i n + \sum_{j=0}^{i} \left(\frac{1}{\sqrt{2}}\right)^i c \leq \left(\frac{1}{\sqrt{2}}\right)^i n + \frac{c}{\sqrt{2}-1}.$$

Since there are 2^i problems at depth i, the total amount of work at level i is

$$2^i \Big(\varphi^i(n)\Big)^2 = 2^i \left(\left(\frac{1}{\sqrt{2}}\right)^i n + O(1)\right)^2 = O\Big(n^2\Big).$$

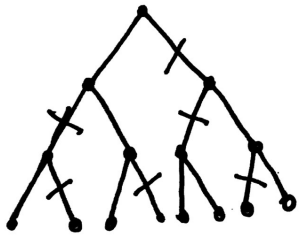
Over $O(\log n)$ levels, then, the total work is $O(n^2 \log n)$.

The proof of correctness is based on a more general phenomena, called the *Galton-Watson process*. We study the Galton-Watson process in greater generality in the following section, and here we only give the reduction from `branching-contractions` to the Galton-Watson process.

We can arrange the recursive calls in a binary tree. Each node consists of a subproblem, with two children consisting of the two subproblems. The leaves are the constant-size subgraphs that can be computed by brute force. The height is $O(\log n)$ because every level decreases n by a constant factor.

The process at a subtree succeeds iff the node succeeds and one of the two subtrees succeeds, and each node succeeds with probability $1/2$. The overall algorithm succeeds iff there is a root to leaf path where every node succeeds. This is the Galton-Watson process over $O(\log n)$ generations, which (by theorem 4.9 below), has a $\Omega(1/\log n)$ probability of success. □

4.3 Randomized branching



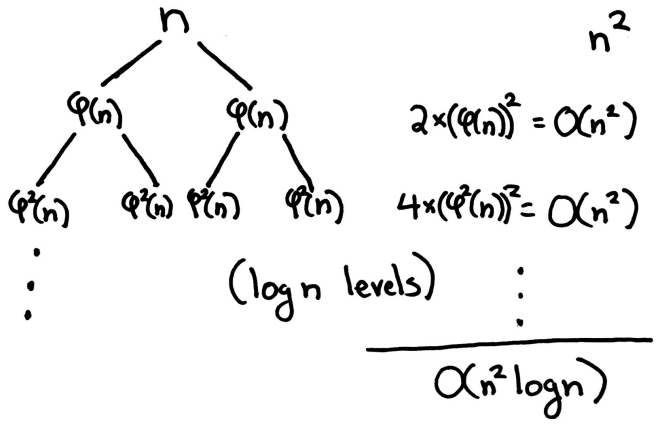


Figure 4.5: A recursion tree analysis for `branching-contractions`.

Theorem 4.9. *Let T be a complete binary tree of height $k \geq 2$, and suppose every edge is deleted independently with probability $1/2$. The probability that there is a leaf connected to the root is $\geq 1/k$.*

Proof. For $i \in \mathbb{N}$. Let p_i be the probability that a particular node at height i is connected to a subleaf. We have $p_0 = 1$. For a node at height $i+1$, the probability that there is no path to a leaf via a particular child is

$$\frac{1}{2} + \frac{1}{2}(1 - p_i) = 1 - \frac{p_i}{2}.$$

p_{i+1} is 1 one minus the probabilities there is no path to a leaf via either child, which by independence to the two subtrees is

$$p_{i+1} = 1 - \left(1 - \frac{p_i}{2}\right)^2 = p_i\left(1 - \frac{p_i}{4}\right). \tag{4.2}$$

The first three values are

$$p_0 = 1,\ p_1 = 3/4, \text{ and } p_2 = \tfrac{39}{64} \geq 1/2.$$

We claim by induction on k that $p_k \geq \frac{1}{k}$ for all $k \geq 2$. Looking at the RHS of (2), we first observe that function

$$f(x) = x\left(1 - \tfrac{x}{4}\right) \text{ is increasing for } x \leq 2,$$

which can be seen from its derivative $f'(x) = 1 - \frac{x}{2} \geq 0$. In particular, to lower bound p_{k+1} via (2), we can replace p_k by any lower bound for p_k. By induction, $p_k \geq 1/k$,

hence

$$p_{k+1} \geq \frac{1}{k}\left(1 - \frac{1}{4k}\right) = \frac{1}{k} - \frac{1}{4k^2} \overset{\text{(a)}}{\geq} \frac{1}{k+1}.$$

The last inequality (a) is obtained by

$$\frac{1}{k} - \frac{1}{k+1} = \frac{1}{k^2+1} \geq \frac{1}{4k^2} \text{ for } k \geq 1. \tag{4.3}$$

□

4.4 Sparsification

Sparsification refers to the general method of taking a large, dense graph and producing a smaller, sparse graph (over the same vertex set) that preserves some desired structure of the original graph.

We start with a very familiar example. Recall that two vertices s and t are *connected* in an undirected graph if there is a path from s to t. Given an undirected graph $G = (V, E)$, suppose we wanted a sparse subgraph $G' = (V, E')$ such that any two vertices $s, t \in V$ are connected in G iff they are connected in G'. Here there is a solution G' with (slightly less than) n edges. The reader probably knows the answer and should pause to realize it.

4.4.1 Preserving small connectivities

Definition 4.10. *Let $G = (V, E)$ be an undirected graph. The* connectivity between two vertices $s, t \in V$ *is the size of the minimum $\{s, t\}$-cut. When G is unweighted, this is also the maximum number of disjoint paths from s to t. The* connectivity of an edge $e \in E$ *is defined as the connectivity of its endpoints.*

Above, we discussed sparsifiers that maintain all pairwise connectivities in an undirected graph up to connectivity 1. Here we will analyze a generalization by Nagamochi and Ibaraki [NI92a] for maintaining connectivities up to a fixed cardinality $k \in \mathbb{N}$. That is, given an undirected graph $G = (V, E)$ and a parameter k, we will compute a set $F \subseteq E$ with less than kn edges with the following guarantees.

1. If $s, t \in V$ have connectivity $\ell \leq k$ in G, then s and t have connectivity ℓ in F.
2. If $s, t \in V$ have connectivity $\ell \geq k$ in G, then s and t have connectivity $\geq k$ in G.

`branching-contractions`$(G = (V, E),\ c)$

1. Let $n = |V|$. If $n \leq 3$ then compute the min-cut by brute force.
2. Until $|V| = \left\lceil \frac{n}{\sqrt{2}} \right\rceil + 1$:
   A. Sample $e \sim c$ and contract e in G.
3. $C_1 \leftarrow$ `branching-contractions`(G, c).
4. $C_2 \leftarrow$ `branching-contractions`(G, c).
5. Uncontract and return the minimum of C_1 and C_2.

Figure 4.4: A randomized minimum cut algorithm that amplifies the `random-contractions` algorithm by branching, due to Karger and Stein [KS96].

of `random-contractions`. The amplification technique is interesting in its own right and extends past this particular problem.

A sketch of the algorithm, which we call `branching-contractions`, is given in fig. 4.4.

Lemma 4.7. *Let $C^\star$, and suppose we contract $n - k$ edges sequentially at random. The probability that none of theses edges samples the minimum cut is at least $\frac{k(k-1)}{n(n-1)}$.*

Proof. For $i \in \mathbb{N}$, let E_i be the probability that we have not sampled $C^\star$ after i iterations. We ware interested in $\mathbf{P}[E_{n-k}]$. We have

$$\mathbf{P}[E_{n-k}] = \prod_{i=1}^{n-k} \mathbf{P}[E_i \mid E_{i-1}] \overset{\text{(a)}}{=} \prod_{i=k+1}^{n} \left(1 - \frac{2}{i}\right)$$
$$= \frac{\prod_{i=3}^{n} \frac{i-2}{i}}{\prod_{i=3}^{k} \frac{i-2}{i}} = \frac{\binom{k}{2}}{\binom{n}{2}} = \frac{k(k-1)}{n(n-1)}.$$

by (a) lemma 4.3. □

Theorem 4.8. `branching-contractions` *runs in $O(n^2 \log n)$, and returns a minimum cut with probability $\Omega(1/\log n)$.*

Proof. We first prove the running time, and then discuss the correctness. The running time is dominated by the recurrence