

طراحی الگوریتم ها (CE221)

جلسه هفدهم:
برش کمینه

سجاد شیرعلی شمرضا

بهار، 1401

دوشنبه، 19 اردیبهشت 1401

اطلاع رسانی

- بخش مرتبط کتاب برای این جلسه: 26
- میان ترم
 - شنبه هفته آینده، 24 اردیبهشت 1401، در ساعت کلاس
 - به صورت حضوری، تشریحی، و جزوه بسته

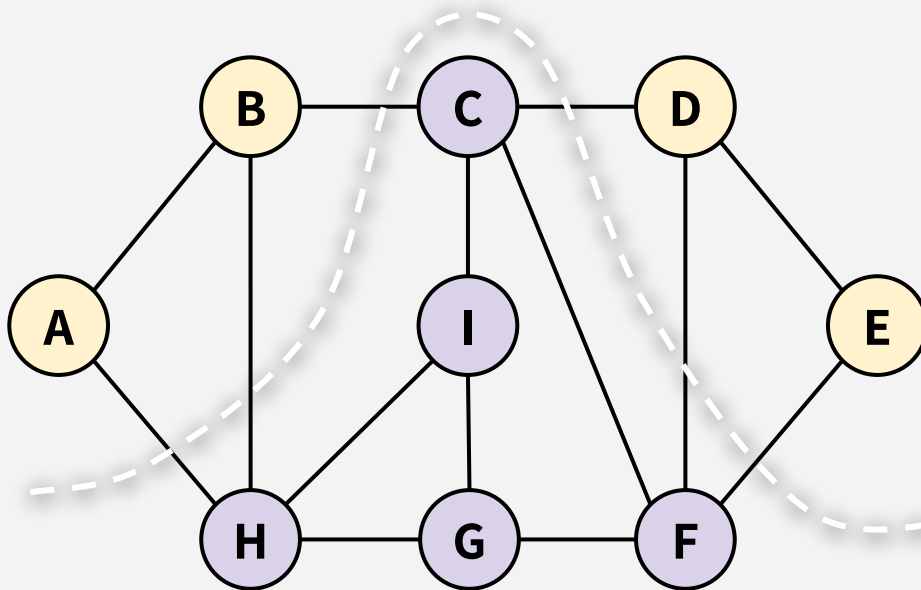
برش کمینه

یک برش کمینه در گراف چیست؟

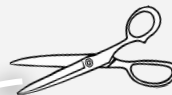
RECALL: CUTS IN GRAPHS

A **cut** is a partition of the vertices into two nonempty parts.

e.g. this is the cut
“{A,B,D,E}
and
{C,I,F,G,H}”



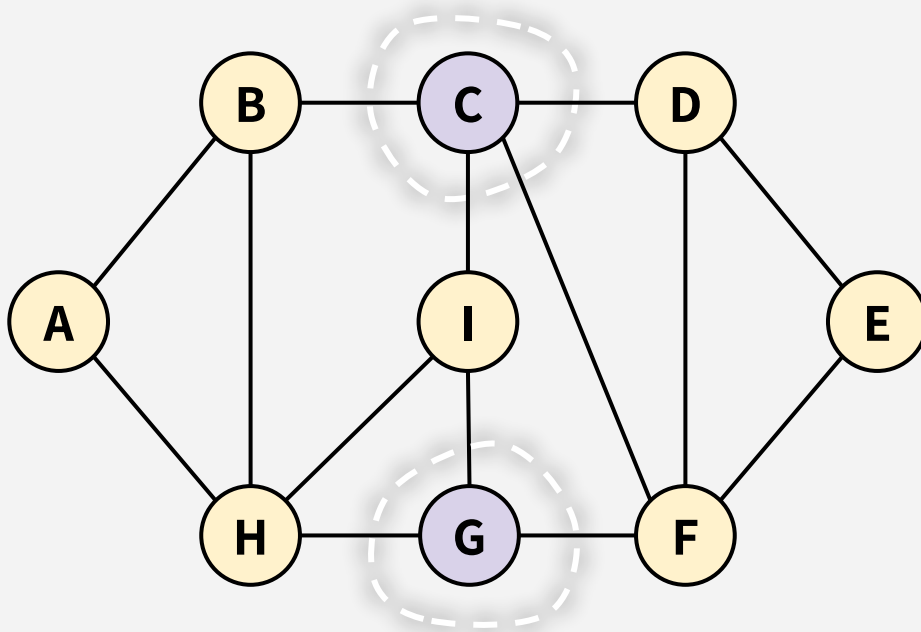
For the first half of class,
we'll stick with talking
about **undirected &
unweighted** graphs.



RECALL: CUTS IN GRAPHS

A **cut** is a partition of the vertices into two nonempty parts.

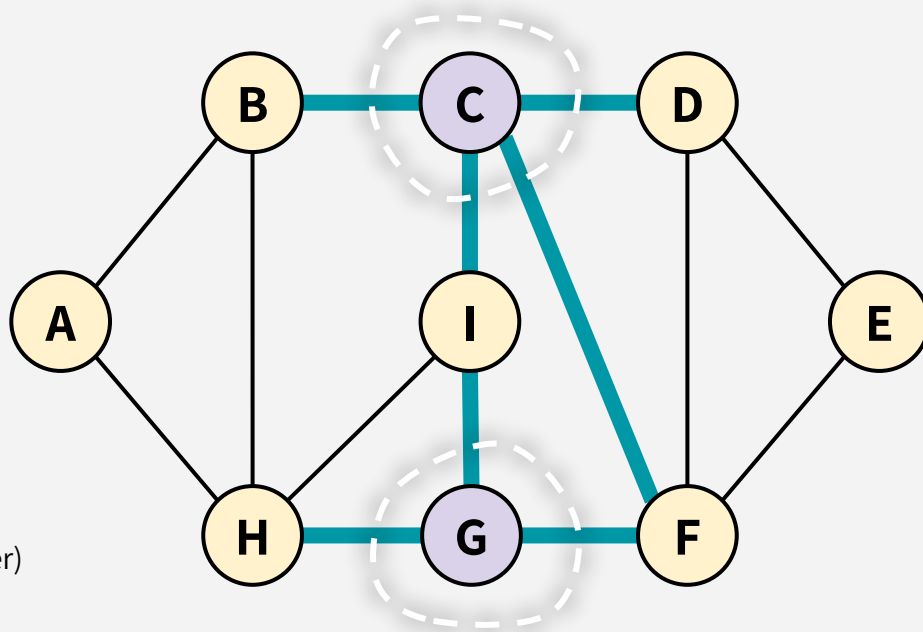
This is also a cut!



For the first half of class, we'll stick with talking about **undirected & unweighted** graphs.

EDGES THAT “CROSS” A CUT

A **cut** is a partition of the vertices into two nonempty parts.



This is also a cut!

And the **teal edges**
are the ones
“**crossing the cut**”

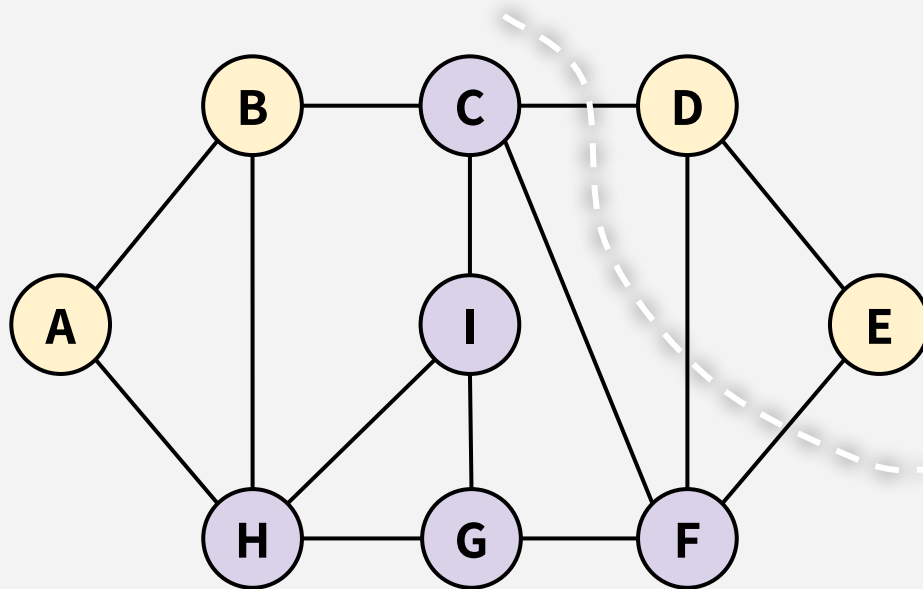
(they go from one part to the other)

For the first half of class,
we'll stick with talking
about **undirected &
unweighted** graphs.

EDGES THAT “CROSS” A CUT

A **cut** is a partition of the vertices into two nonempty parts.

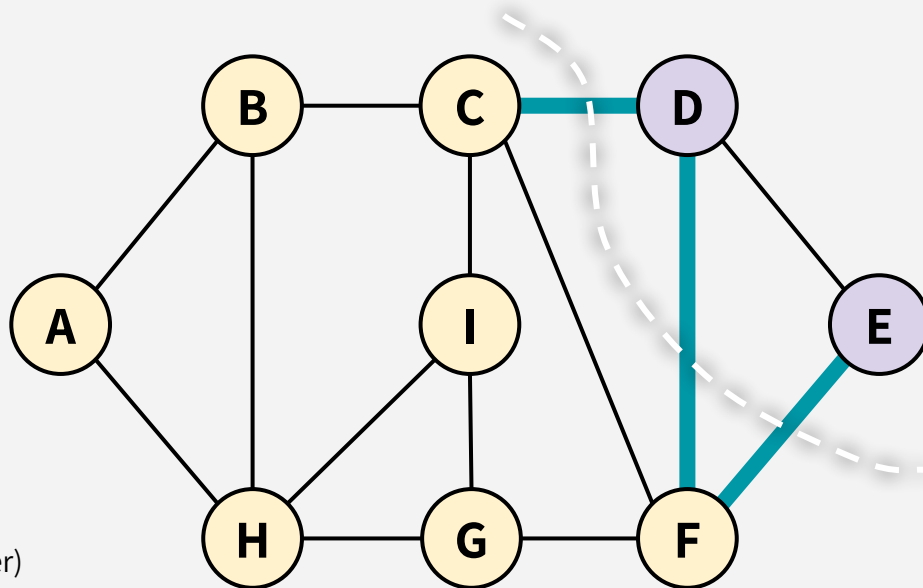
This is also a cut!



For the first half of class, we'll stick with talking about **undirected & unweighted** graphs.

EDGES THAT “CROSS” A CUT

A **cut** is a partition of the vertices into two nonempty parts.



This is also a cut!

And the **teal edges**
are the ones
“**crossing the cut**”

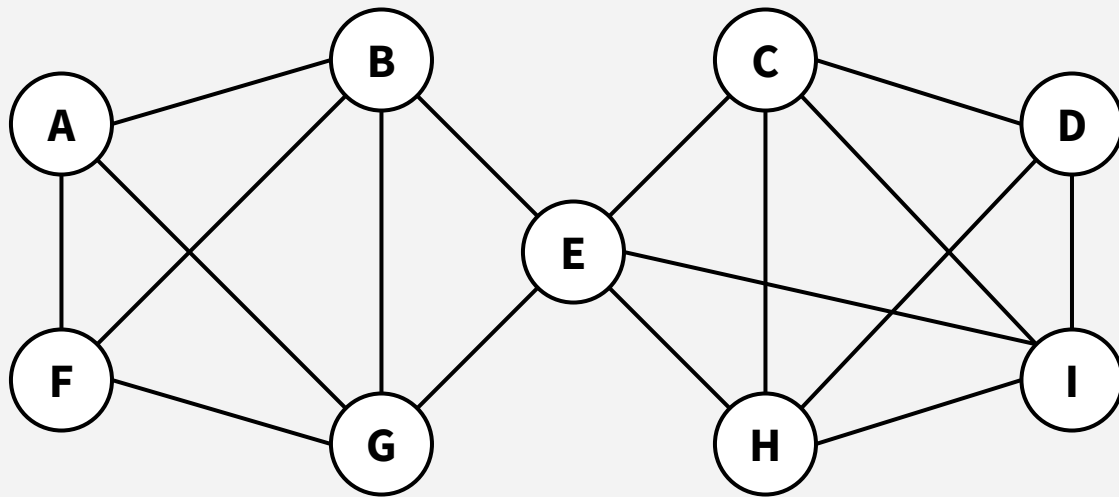
(they go from one part to the other)

For the first half of class,
we'll stick with talking
about **undirected &
unweighted** graphs.

(GLOBAL) MINIMUM CUT

A (global) **minimum cut** is a cut that has the fewest edges possible crossing it.

What is the (global) minimum cut in this graph?

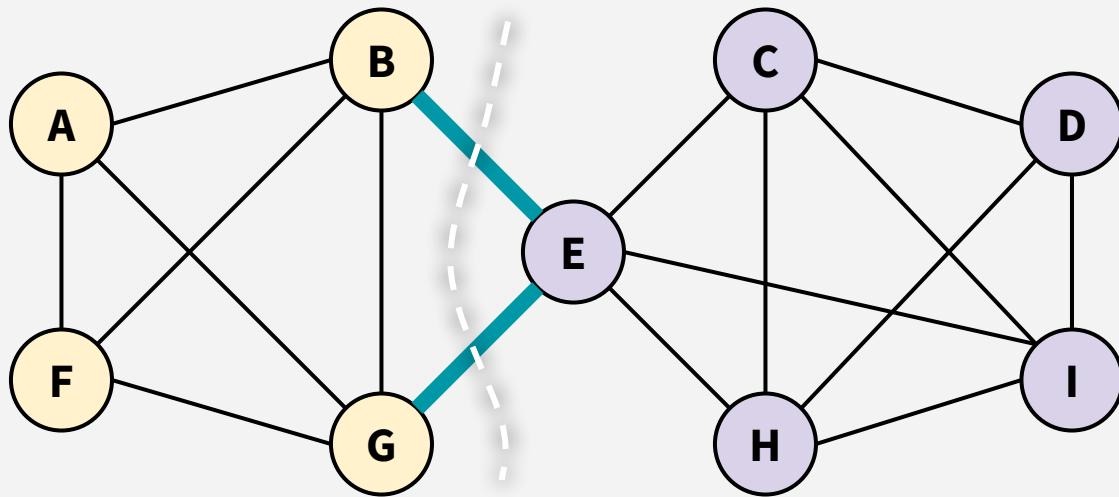


Note: we emphasize “global” because later in class, we’ll talk about *s-t minimum cuts*, which is a cut that separates specific nodes *s* and *t* (a slightly different concept).

(GLOBAL) MINIMUM CUT

A (global) **minimum cut** is a cut that has the fewest edges possible crossing it.

What is the (global) minimum cut in this graph?

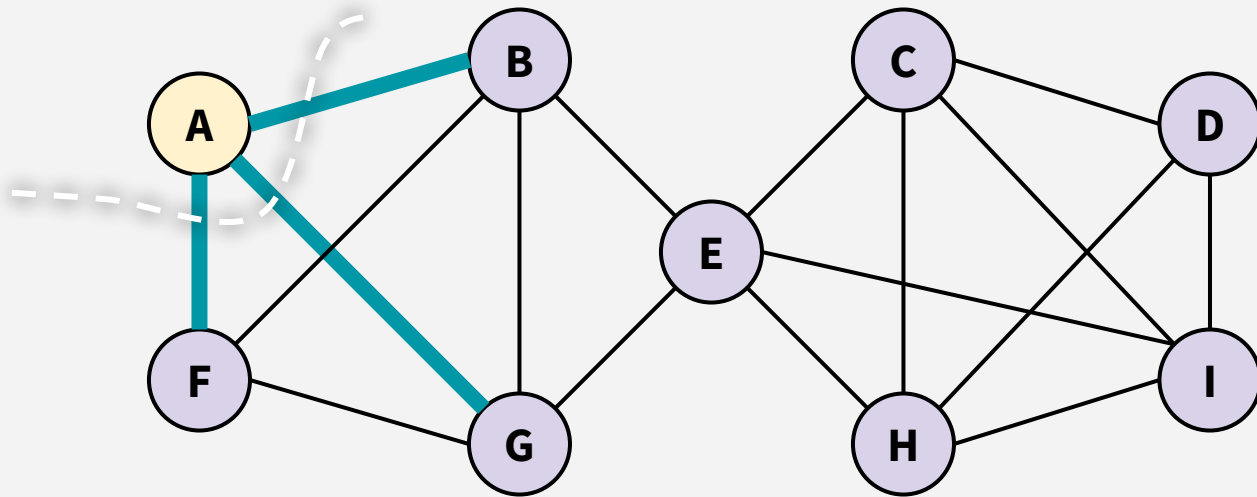


This cut is the (global) minimum cut (it has only 2 edges crossing it)!

(GLOBAL) MINIMUM CUT

A (global) **minimum cut** is a cut that has the fewest edges possible crossing it.

What is the (global) minimum cut in this graph?

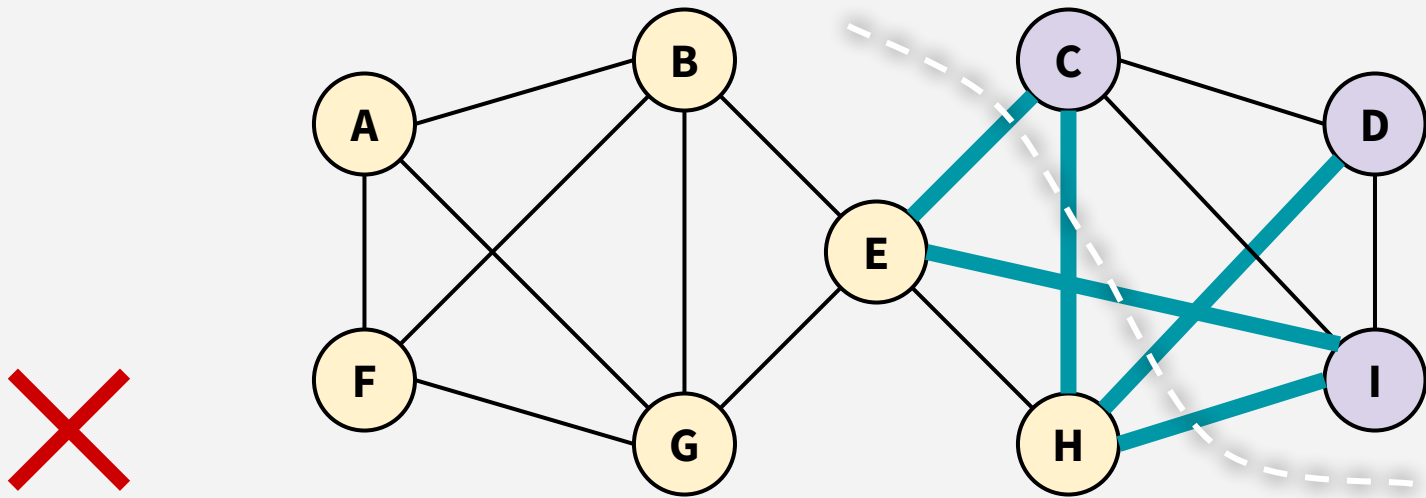


This is **NOT** a global min-cut (3 edges cross it, so it's not better than the one on the last slide)

(GLOBAL) MINIMUM CUT

A (global) **minimum cut** is a cut that has the fewest edges possible crossing it.

What is the (global) minimum cut in this graph?



This is also **NOT** a global min-cut (5 edges cross it, so it's not better than the cut we already found) 12

APPLICATIONS?

There are several immediate use cases for finding global minimum cuts!

Clustering

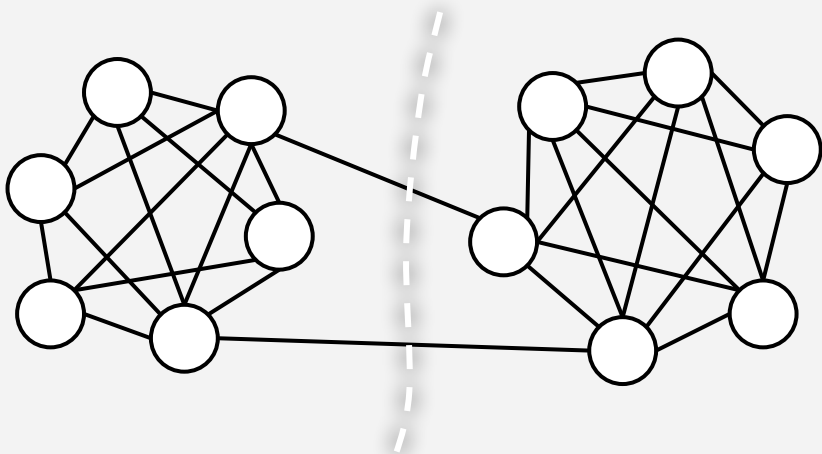
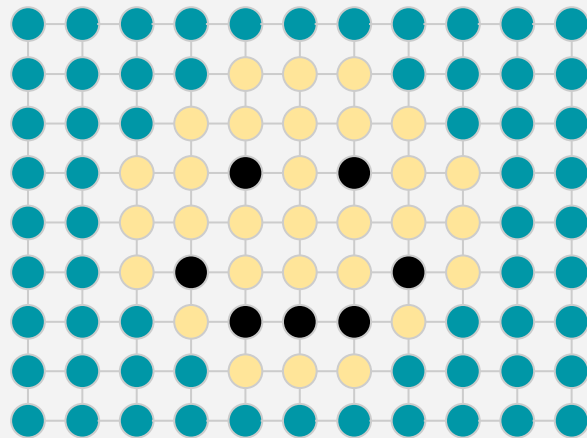


Image Segmentation*



*Although we're only working with unweighted graphs for now, we could represent an image using large edge weights between similar pixels



سوال؟

الگوریتم کارگر

یک الگوریتم تصادفی برای یافتن برش کمینه سراسری در گراف بدون جهت

LAS VEGAS vs. MONTE CARLO

LAS VEGAS ALGORITHMS

Guarantees correctness!

But the runtime is a random variable.
(i.e. there's a chance the runtime could take awhile)

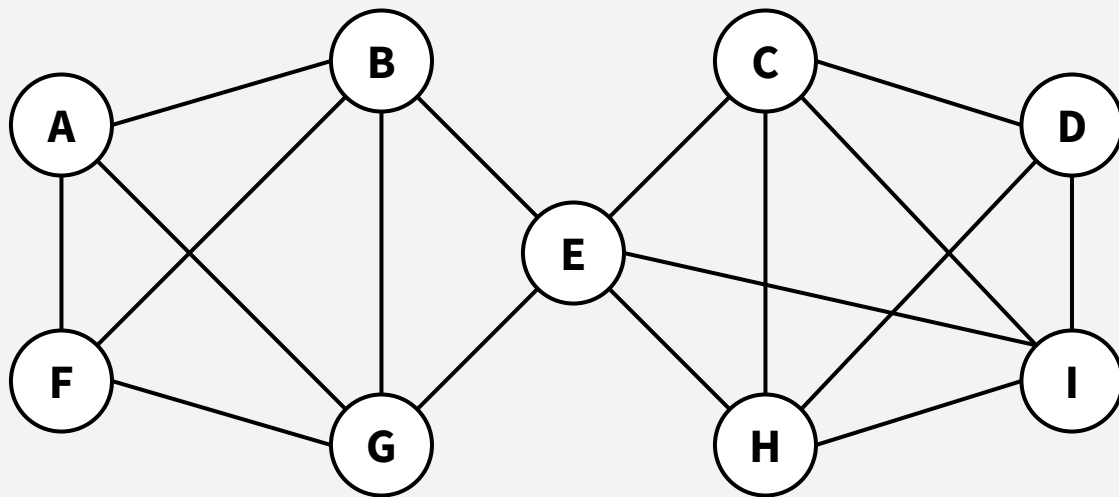
MONTE CARLO ALGORITHMS

Correctness is a random variable.
(i.e. there's a chance the output is wrong)

But the runtime is guaranteed!

KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

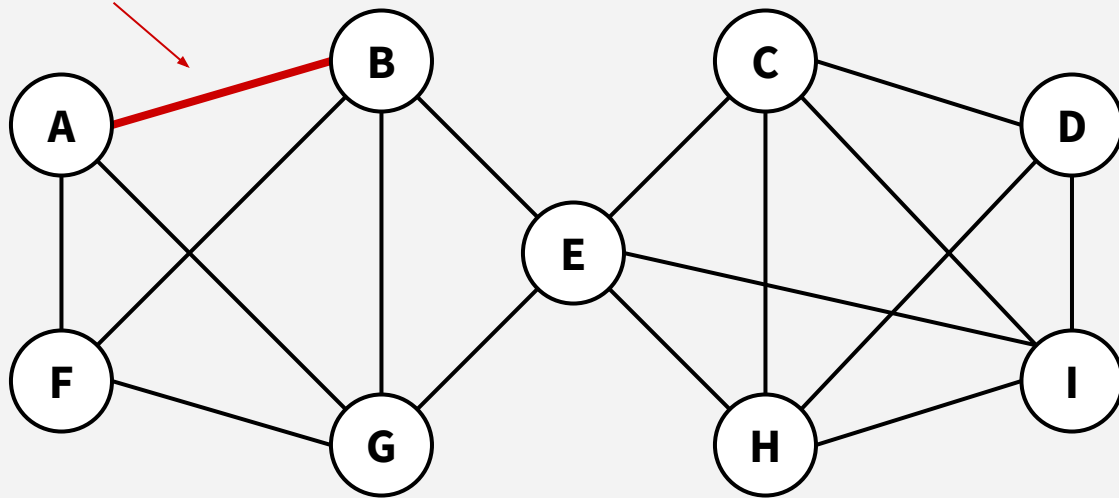


As always, let's see an example of the algorithm in action!

KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

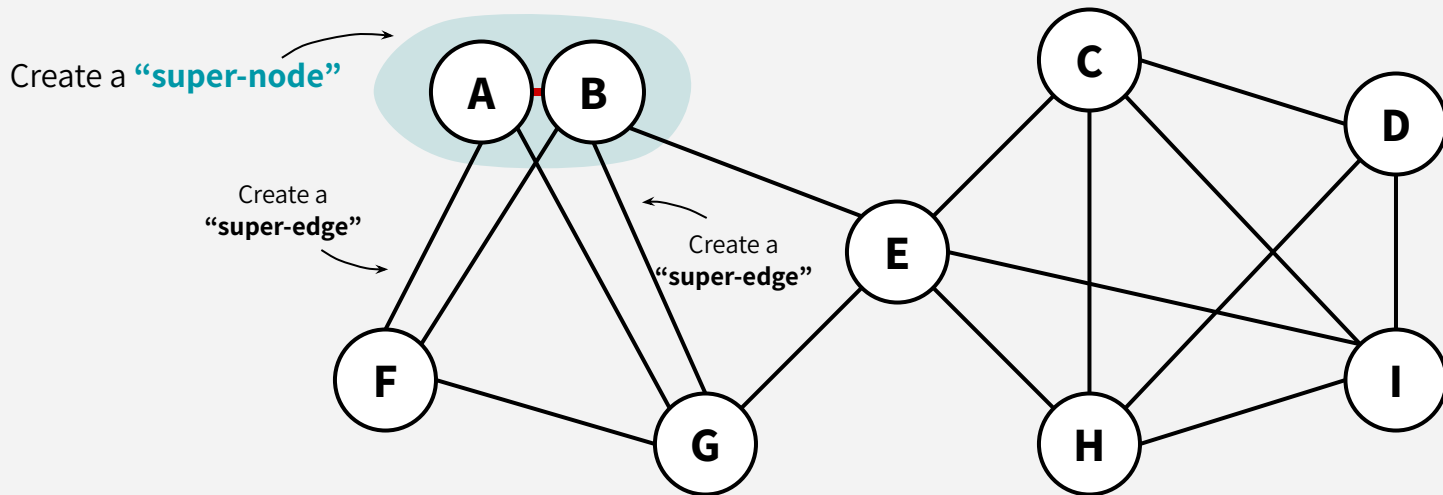
Pick a random edge!



KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

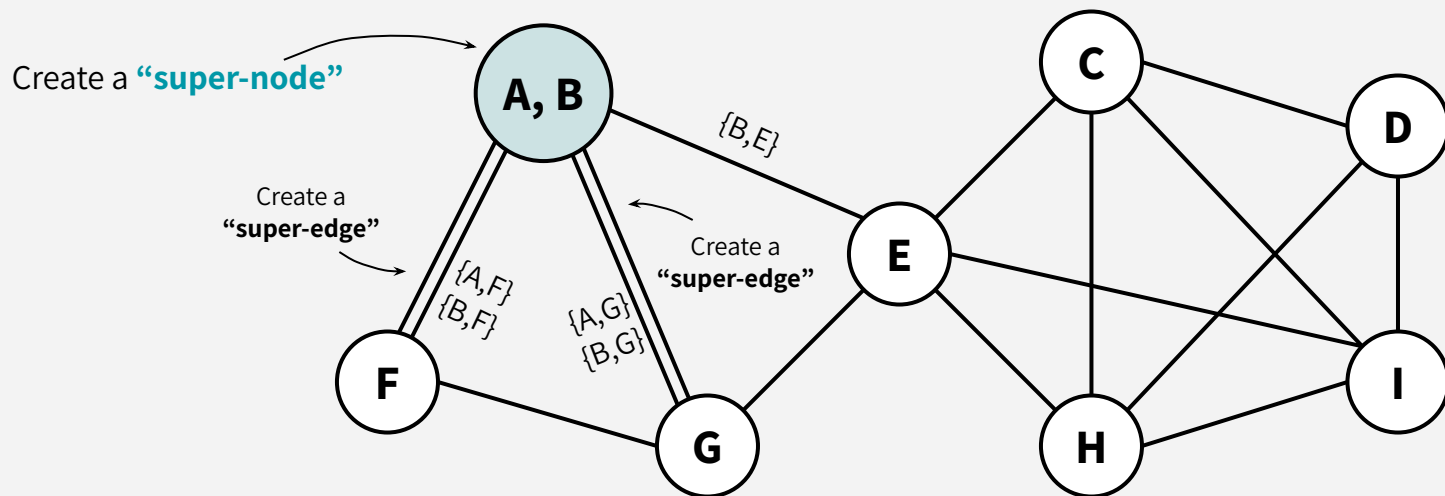
Now perform an “edge contraction”!



KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

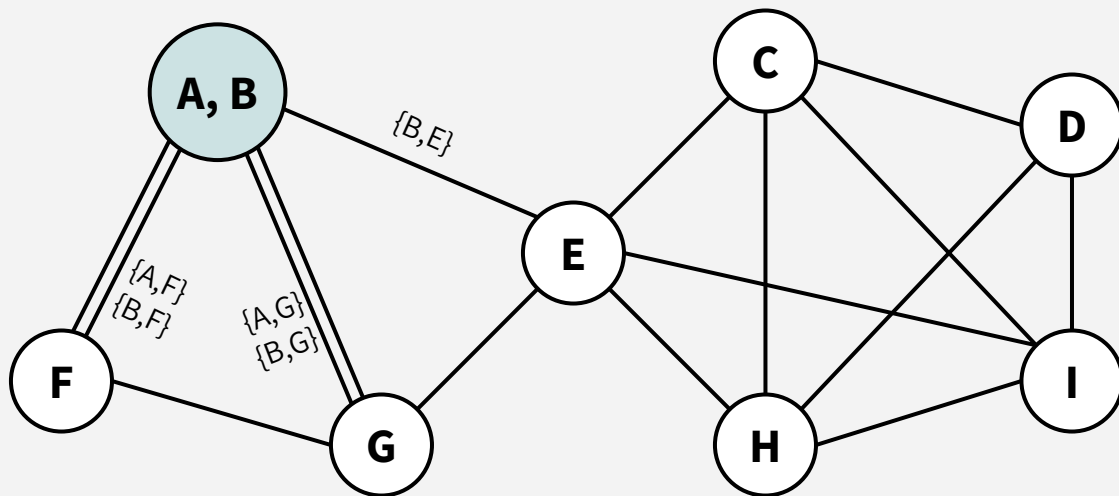
Now perform an “edge contraction”!



KARGER'S ALGORITHM

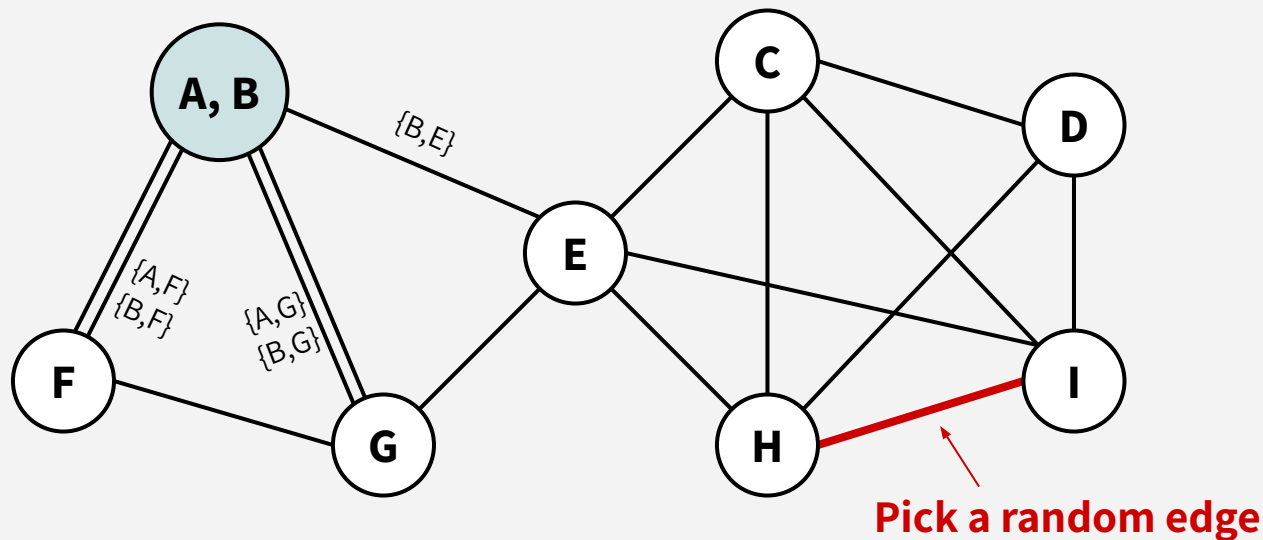
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Repeat!



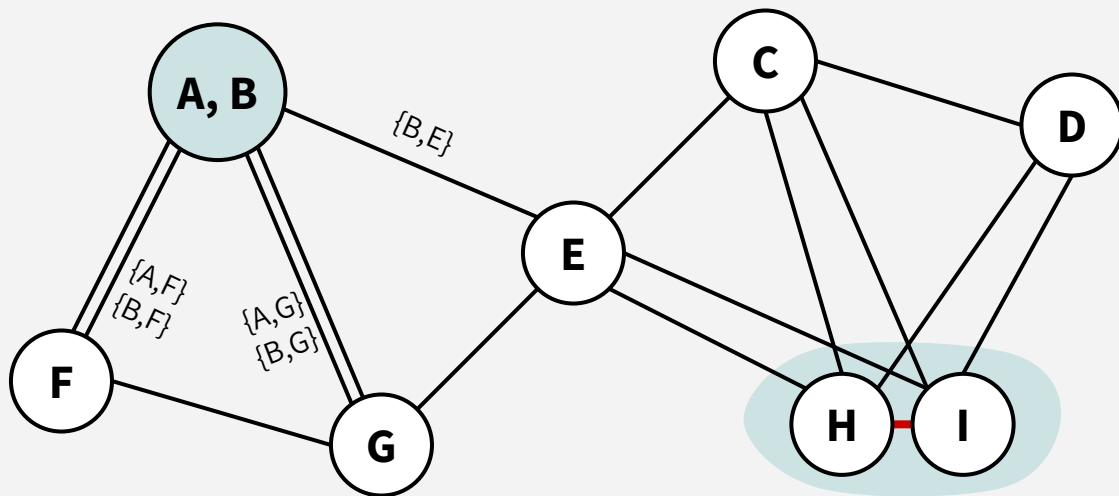
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

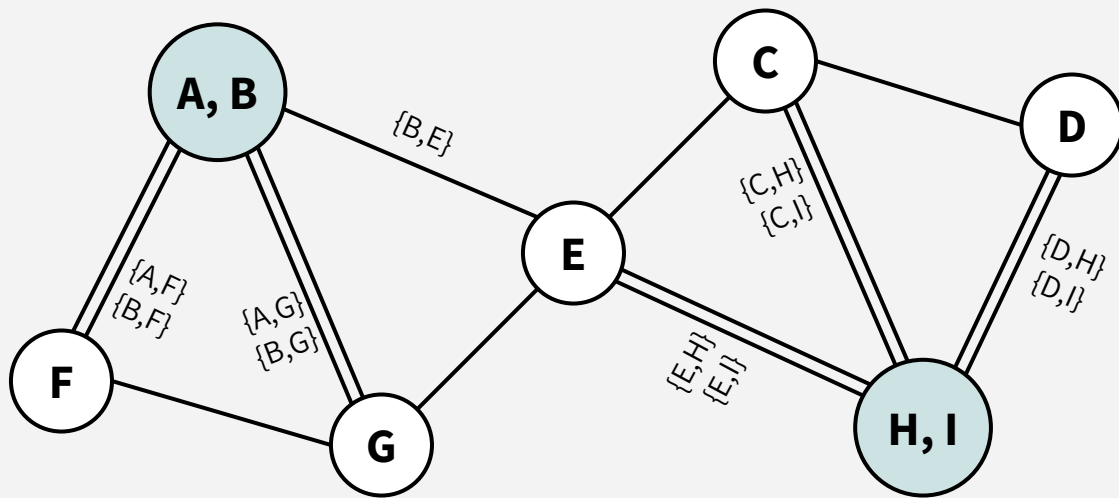
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Now perform an “edge contraction”!

KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

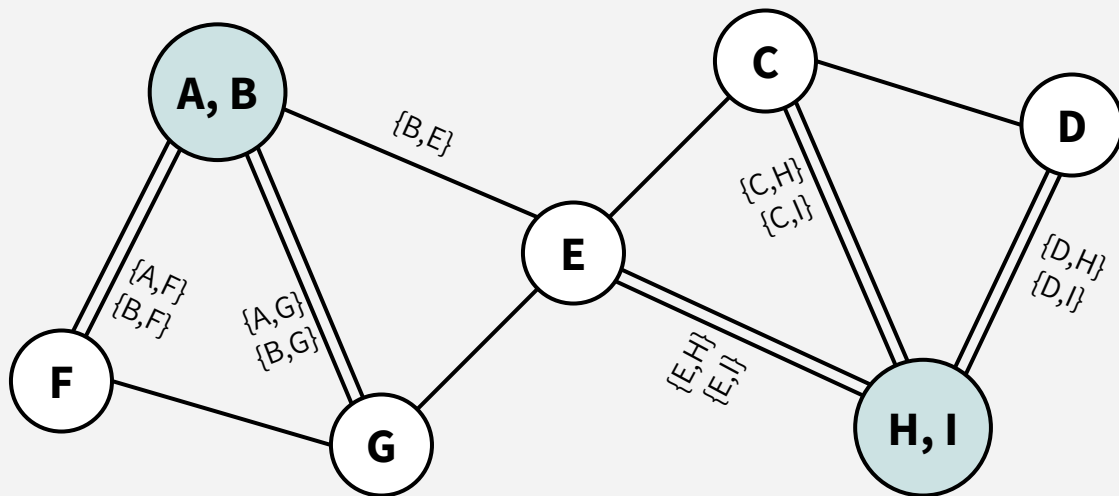


Now perform an “edge contraction”!

KARGER'S ALGORITHM

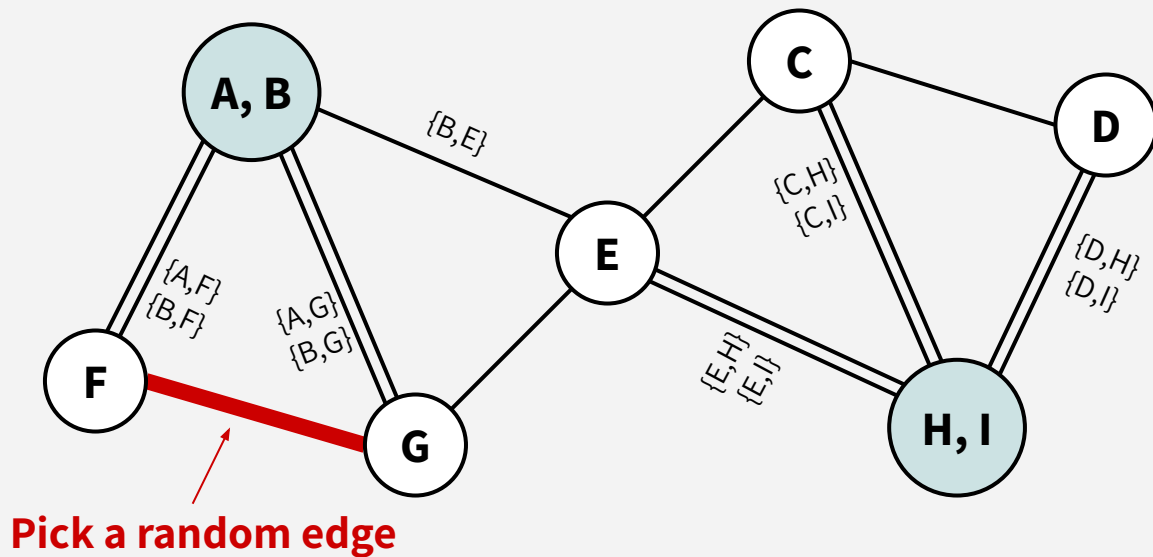
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Repeat!



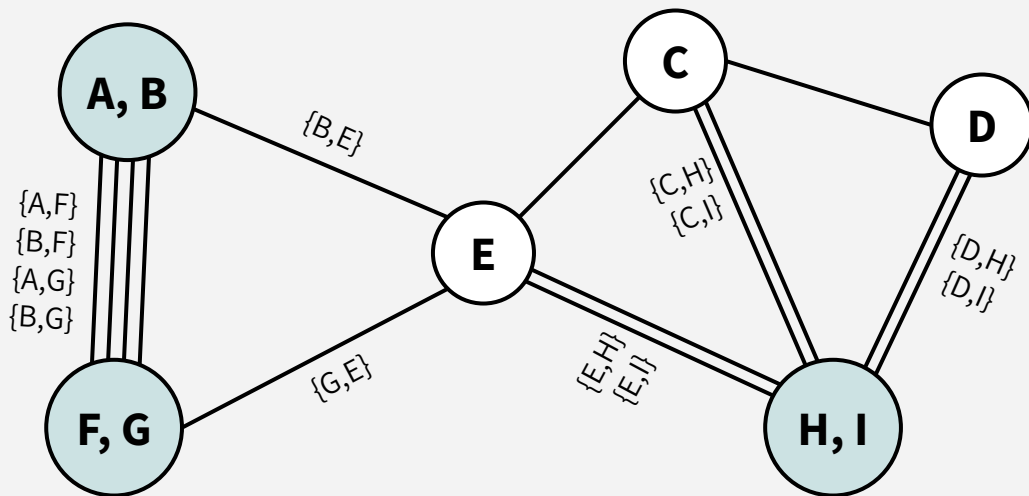
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

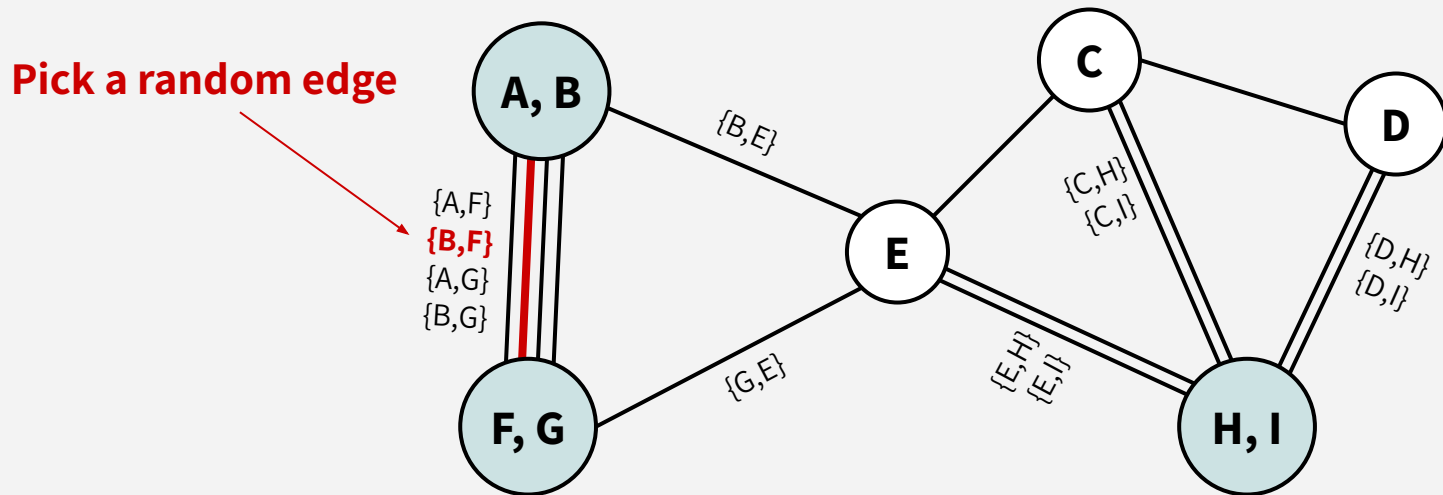
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Perform an “edge contraction”

KARGER'S ALGORITHM

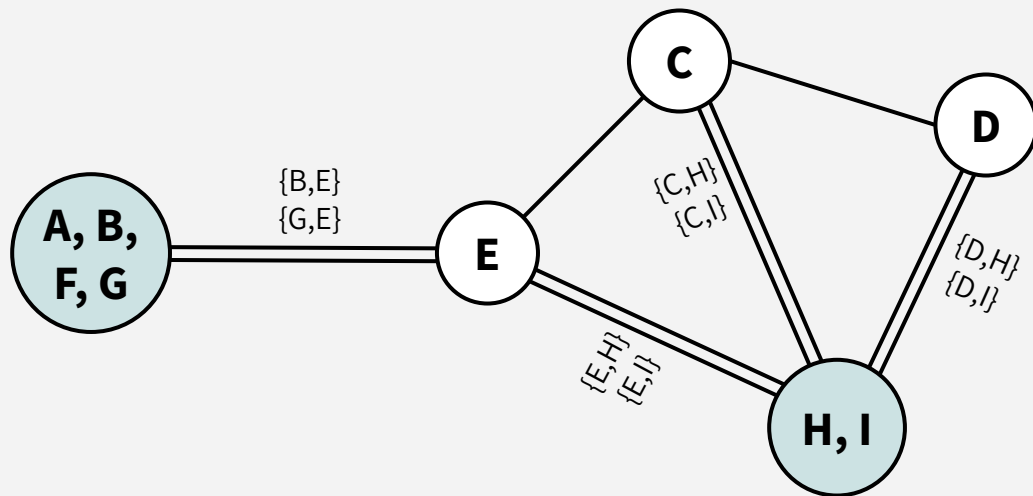
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

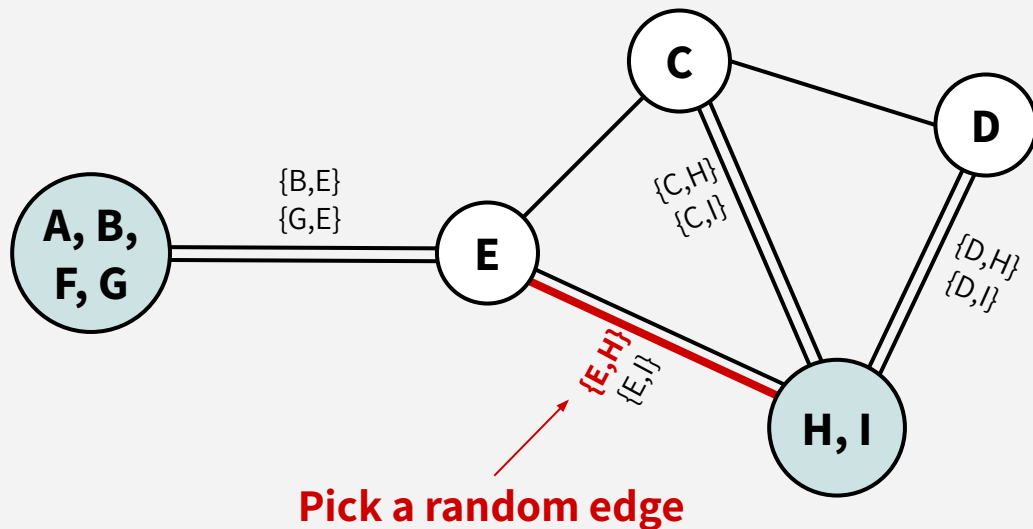
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Perform an
“edge contraction”



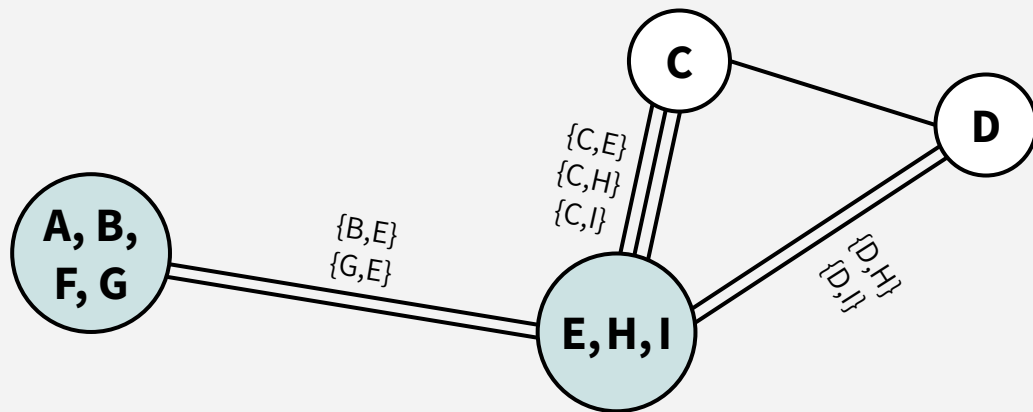
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

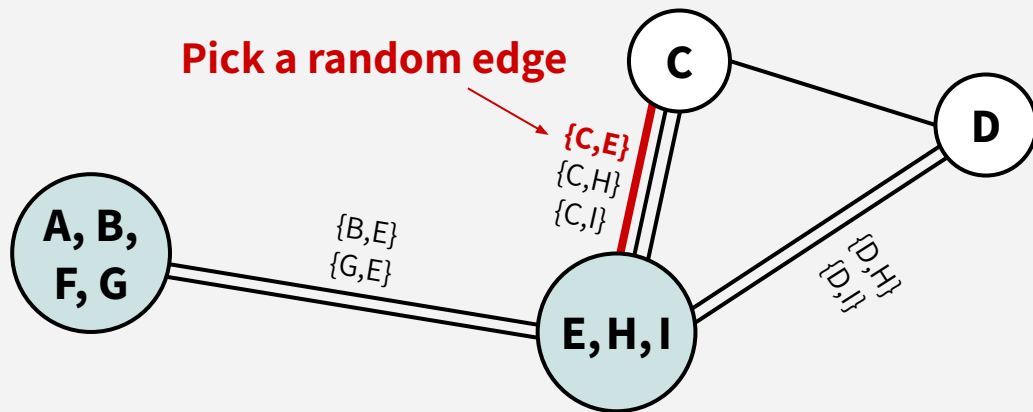
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Perform an “edge contraction”

KARGER'S ALGORITHM

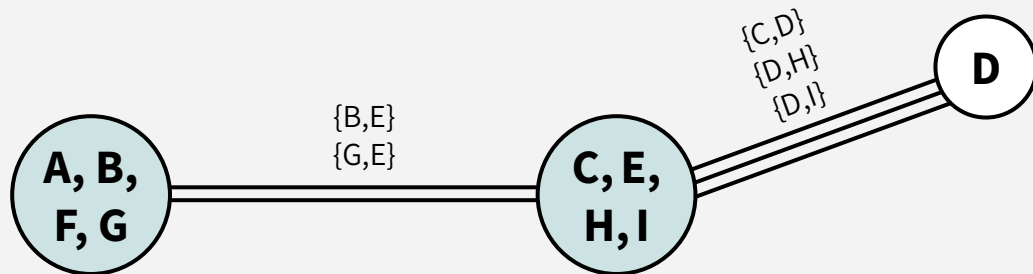
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

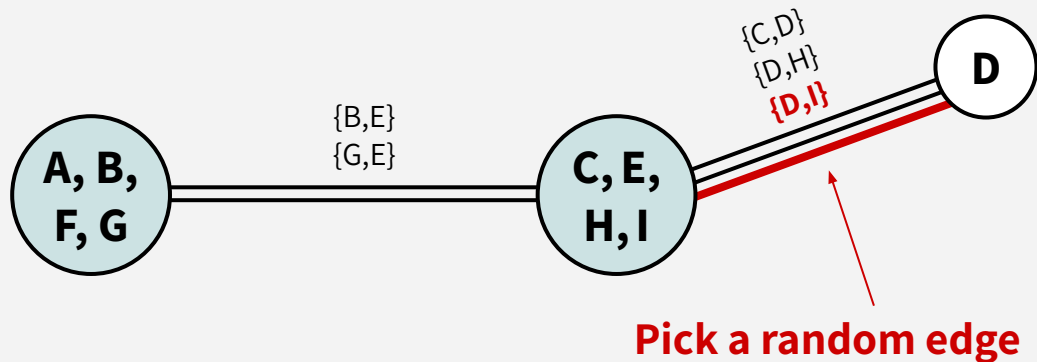
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Perform an “edge contraction”



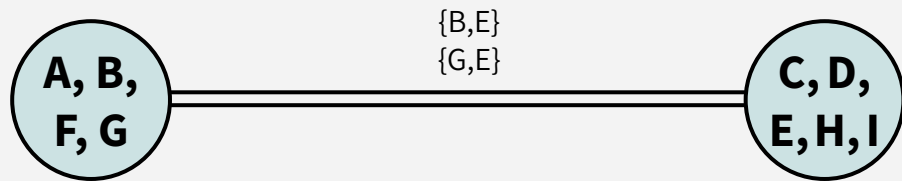
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

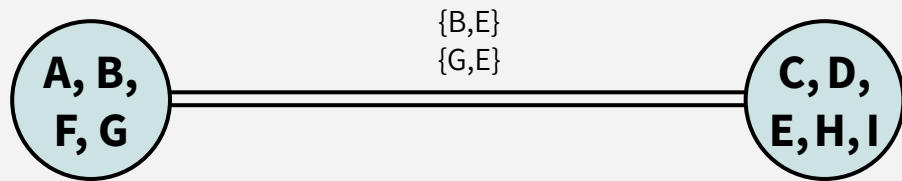


Perform an “edge contraction”

KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

We return the cut given by the remaining 2 super-vertices!

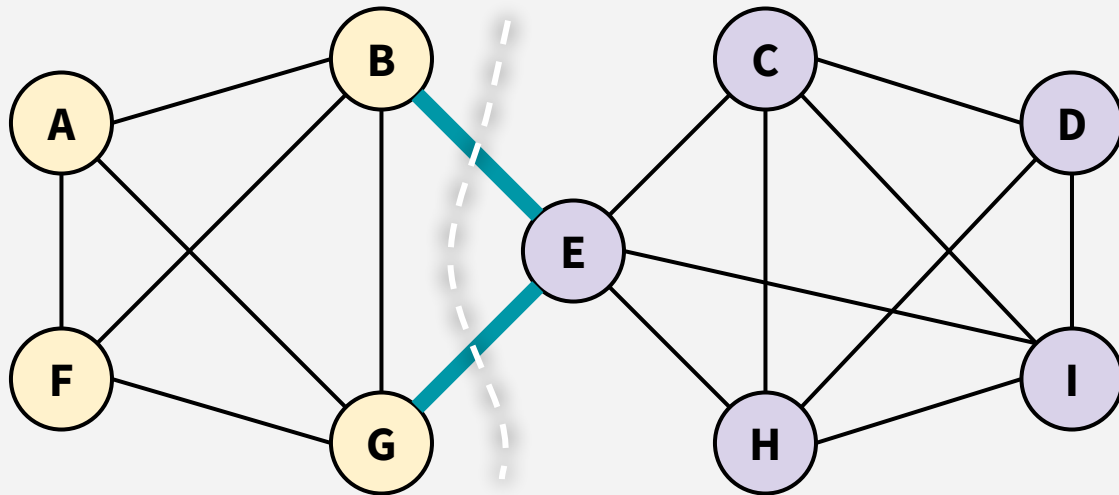


KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

We return the cut given by the remaining 2 super-vertices!

(Just bringing back
our original graph
to see what this
corresponds to)





سوال؟

زمان اجرای و درستی الگوریتم کارگر

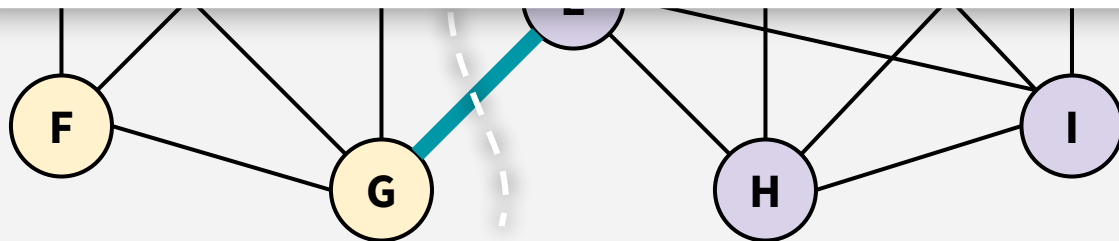
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

We return the cut given by the remaining 2 super-vertices!

(Just bringing back
our original graph
to see what this
corresponds to)

How fast is this? Does it always work?



KARGER'S ALGORITHM: PSEUDOCODE

KARGER($G=(V,E)$):

vertices of $G' = \{\text{supernode}(v) \text{ for } v \text{ in } V\}$ ← one supernode for each vertex

$E'_{u',v'} = \{(u,v)\}$ for $(u,v) \text{ in } E$ ← one superedge for each edge

$E'_{u',v'} = \{\}$ for $(u,v) \text{ not in } E$

$F = \text{a copy of } E$ ← we'll choose randomly from F

while $|G'| > 2$:

$\{(u,v)\} = \text{uniformly random edge in } F$

MERGE_SUPERNODES(u,v)

$F = F \setminus E'_{u',v'}$

return the cut given by the remaining two supernodes!

now that u' and v' (supernodes of u and v , respectively) are merged, remove edges in the superedge between u' and v' (those edges are no longer "candidate" edges)

MERGE_SUPERNODES(u,v):

$x' = \text{supernode}(\text{nodes in } u' \cup \text{nodes in } v')$

for each $w' \text{ in } G' \setminus \{u', v'\}$:

$E_{x',w'} = E_{u',w'} \cup E_{v',w'}$

 remove u' and v' from G' and add x' ← x' is our merged result!

assume that **MERGE_SUPERNODES** knows about u' and v' (the supernodes u and v belong to)

KARGER'S ALGORITHM: PSEUDOCODE

The while loop
executes $n - 2$ times...

MERGE_SUPERNODES
takes $O(n)$ time naively
(we can be more clever
with a union-find data
structure but don't worry
about that for today!)

KARGER($G=(V,E)$):

vertices of $G' = \{\text{supernode}(v) \text{ for } v \text{ in } V\}$ ← one supernode for each vertex

$E'_{u,v} = \{(u,v)\}$ for $(u,v) \text{ in } E$ ← one superedge for each edge

$E'_{u,v} = \{\}$ for $(u,v) \text{ not in } E$

$F = \text{a copy of } E$ ← we'll choose randomly from F

while $|G'| > 2$:

$\{(u,v)\} = \text{uniformly random edge in } F$

MERGE_SUPERNODES(u,v)

$F = F \setminus E'_{u,v}$

return the cut given by the remaining two supernodes!

now that u' and v' (supernodes of u and v , respectively)
are merged, remove edges in the superedge between u'
and v' (those edges are no longer "candidate" edges)

MERGE_SUPERNODES(u,v):

$x' = \text{supernode}(\text{nodes in } u' \cup \text{nodes in } v')$

for each $w' \text{ in } G' \setminus \{u', v'\}$:

$E_{x',w'} = E_{u',w'} \cup E_{v',w'}$

remove u' and v' from G' and add x' ← x' is our merged result!

assume that **MERGE_SUPERNODES** knows about
 u' and v' (the supernodes u and v belong to)

Runtime (w/o fancy data structures): $O(n^2)$

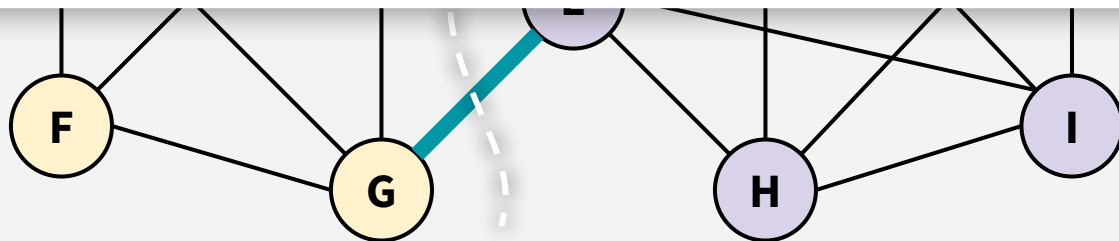
KARGER'S ALGORITHM

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

We return the cut given by the remaining 2 super-vertices!

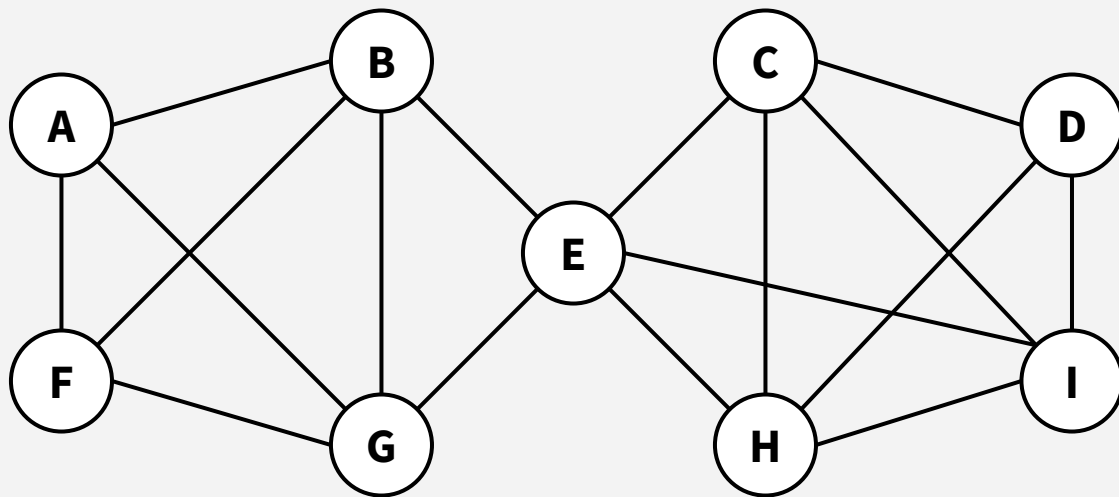
(Just bringing back
our original graph
to see what this
corresponds to)

How fast is this? **Does it always work?**



WHAT IS AN “UNLUCKY” CHOICE?

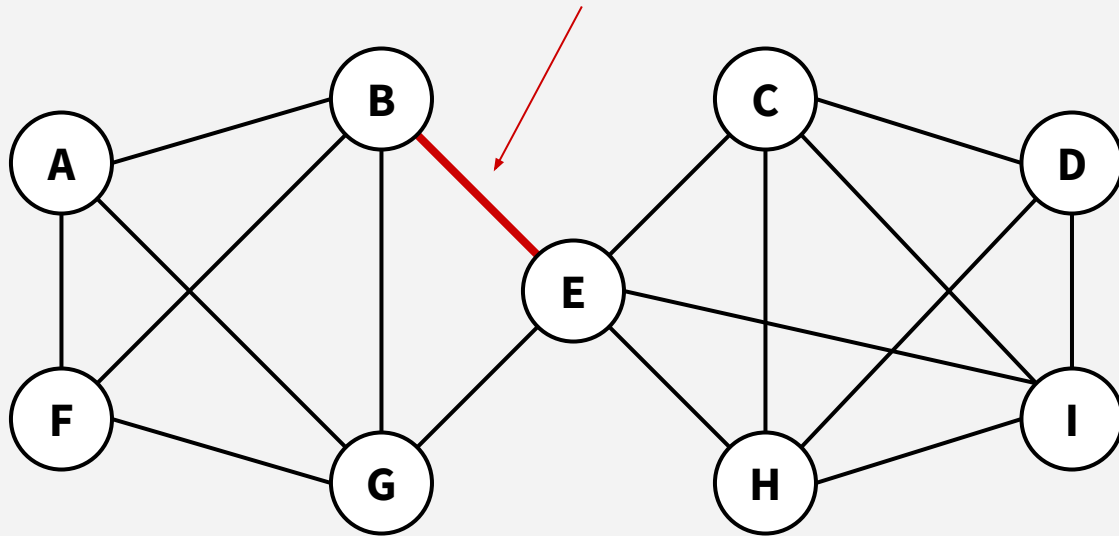
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Let's see what happens if we're not so lucky all the time...

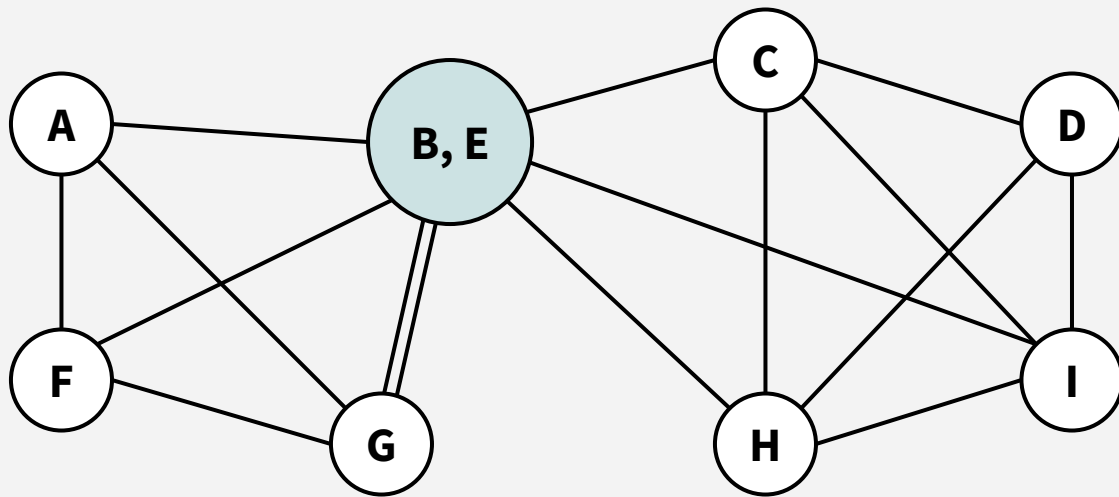
WHAT IS AN “UNLUCKY” CHOICE?

Pick a random edge! (this is a more “unlucky” choice)



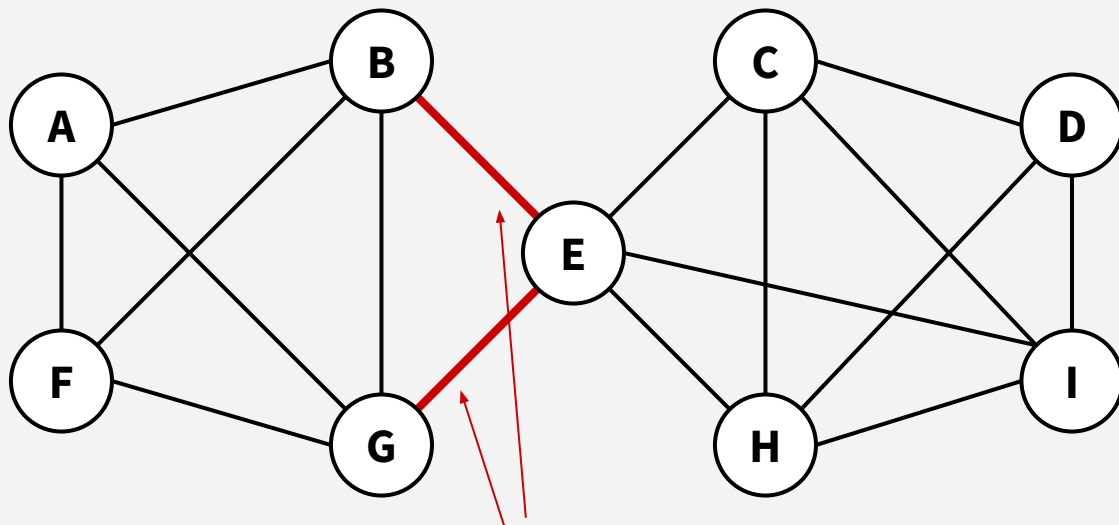
WHAT IS AN “UNLUCKY” CHOICE?

After we do the “edge contraction”, there’s no way to return a cut that separates B and E.



WHAT IS AN “UNLUCKY” CHOICE?

In fact, if Karger’s algorithm ever randomly selects **edges in the min-cut**,
then it won’t return that min-cut.



These would be very unlucky edges to pick...

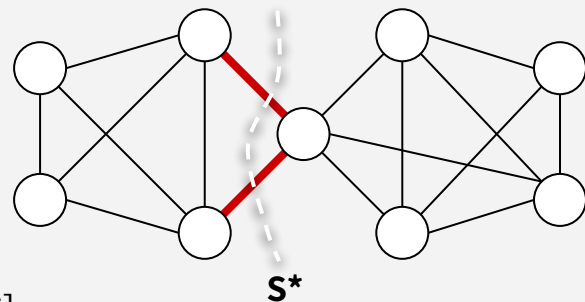
KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF: Suppose S^* is a min-cut, and suppose we select edges e_1, e_2, \dots, e_{n-2} . Then:

$$\begin{aligned} \text{Pr}[\text{return } S^*] &= \text{Pr}[\text{none of the } e_i \text{ cross } S^*] \\ &= \text{Pr}[e_1 \text{ doesn't cross } S^*] \\ &\quad \times \text{Pr}[e_2 \text{ doesn't cross } S^* \mid e_1 \text{ doesn't cross } S^*] \\ &\quad \times \text{Pr}[e_3 \text{ doesn't cross } S^* \mid e_1, e_2 \text{ don't cross } S^*] \\ &\quad \dots \\ &\quad \times \text{Pr}[e_{n-2} \text{ doesn't cross } S^* \mid e_1, \dots, e_{n-3} \text{ don't cross } S^*] \end{aligned}$$



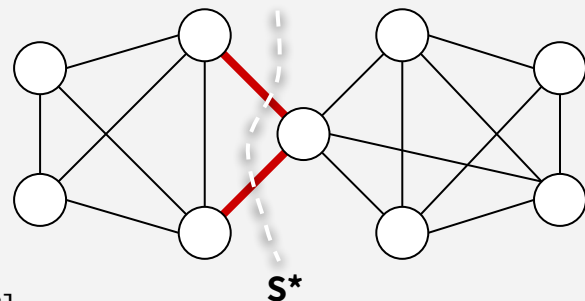
KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1/\binom{n}{2}$

PROOF: Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned}\Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*]\end{aligned}$$

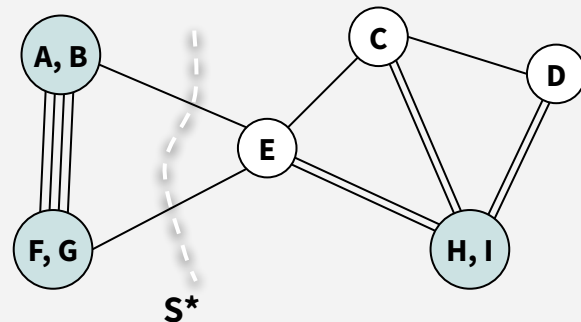


$$\geq 1/\binom{n}{2}$$

We need to show this! We can find an expression for:
 $\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$

KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?



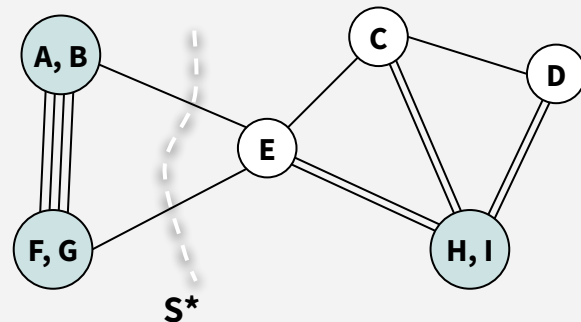
KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?



KARGER'S PROBABILITY OF SUCCESS

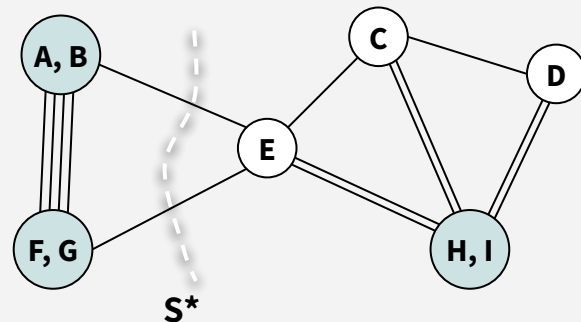
PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

Say there are k edges that cross S^* .



KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

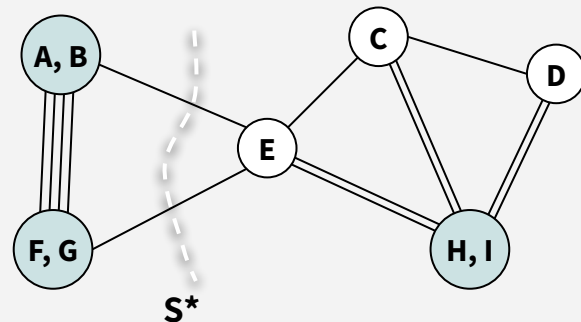
This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.



KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

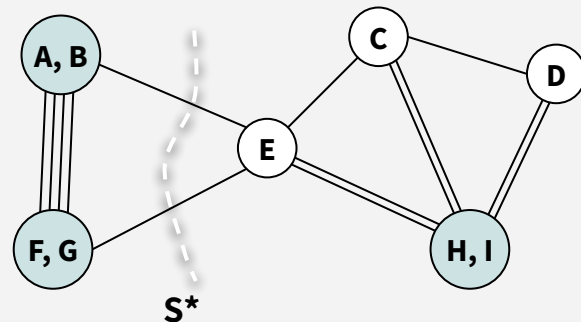
Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.

(Otherwise, that would mean we actually have a smaller cut!)



KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

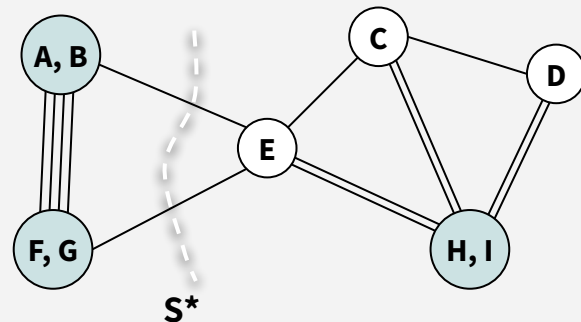
What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.

(Otherwise, that would mean we actually have a smaller cut!)

Thus, there are at least $k(n-j+1)/2$ remaining edges to choose from!



KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

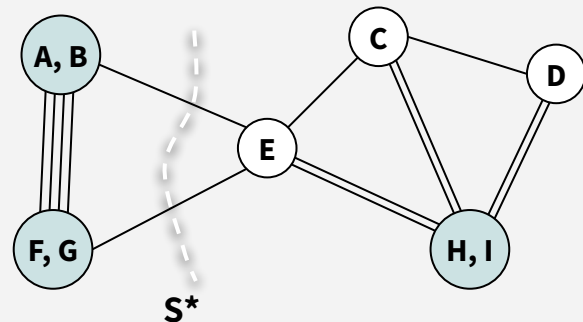
Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.

(Otherwise, that would mean we actually have a smaller cut!)

Thus, there are at least $k(n-j+1)/2$ remaining edges to choose from!

Since the number of nodes decreases by 1 each time we choose an edge, there are $n - (j - 1) = n - j + 1$ nodes left, *each with at least k edges*.



KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

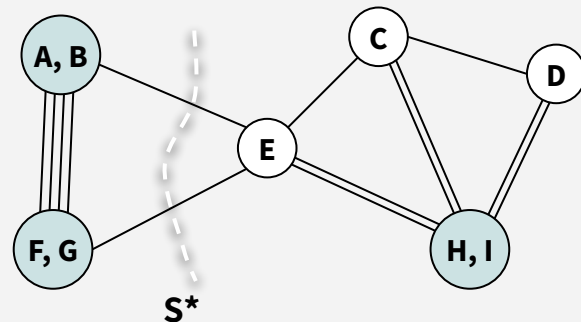
Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.

(Otherwise, that would mean we actually have a smaller cut!)

Thus, there are at least $k(n-j+1)/2$ remaining edges to choose from!

Since the number of nodes decreases by 1 each time we choose an edge, there are $n - (j - 1) = n - j + 1$ nodes left, *each with at least k edges*.



We basically computed
 $1 - \Pr[\text{failure}]$
 $= 1 - (k / \text{total \# remaining edges})$

So, the probability that our j^{th} edge is also *safe* is: $\geq 1 - \frac{k}{\frac{k(n-j+1)}{2}}$

KARGER'S PROBABILITY OF SUCCESS

PROOF (cont'd): What is $\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$?

This is equivalent to answering this:

Suppose after $j-1$ edge choices, Karger hasn't messed up yet!

What's the probability that our j^{th} edge, e_j , also doesn't cross S^* ?

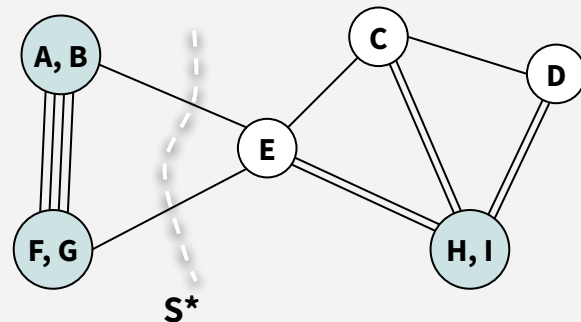
Say there are k edges that cross S^* .

All nodes must currently have at least k (original) edges coming out.

(Otherwise, that would mean we actually have a smaller cut!)

Thus, there are at least $k(n-j+1)/2$ remaining edges to choose from!

Since the number of nodes decreases by 1 each time we choose an edge, there are $n - (j - 1) = n - j + 1$ nodes left, *each with at least k edges*.



We basically computed

$1 - \Pr[\text{failure}]$

$= 1 - (k / \text{total \# remaining edges})$

So, the probability that our j^{th} edge is also safe is:

$$\geq 1 - \frac{k}{\frac{k(n-j+1)}{2}} = 1 - \frac{2}{n-j+1} = \frac{n-j-1}{n-j+1}$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\Pr[\text{return } \mathbf{S}^*] = \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*]$$

$$= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*]$$

...

$$\times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*]$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \left(\frac{n-5}{n-3}\right) \cdots \left(\frac{4}{6}\right) \left(\frac{3}{5}\right) \left(\frac{2}{4}\right) \left(\frac{1}{3}\right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{n-3}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{n-5}{n-3} \right) \cdots \left(\frac{4}{6} \right) \left(\frac{3}{5} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{n-4}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \cdots \left(\frac{4}{6} \right) \left(\frac{3}{5} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \cdots \left(\frac{4}{6} \right) \left(\frac{3}{5} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \cdots \left(\frac{4}{6} \right) \left(\frac{3}{5} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\Pr[\text{return } \mathbf{S}^*] = \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*]$$

$$= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*]$$

...

$$\times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*]$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \cdots \left(\frac{\cancel{4}}{\cancel{6}} \right) \left(\frac{3}{5} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \dots \left(\frac{\cancel{4}}{\cancel{6}} \right) \left(\frac{\cancel{3}}{\cancel{5}} \right) \left(\frac{2}{4} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{n-1} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \dots \left(\frac{\cancel{4}}{\cancel{6}} \right) \left(\frac{3}{\cancel{5}} \right) \left(\frac{2}{\cancel{4}} \right) \left(\frac{1}{3} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\Pr[\text{return } \mathbf{S}^*] = \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*]$$

$$= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*]$$

$$\times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*]$$

...

$$\times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*]$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{\cancel{n-2}}{n} \right) \left(\frac{\cancel{n-3}}{\cancel{n-1}} \right) \left(\frac{\cancel{n-4}}{\cancel{n-2}} \right) \left(\frac{\cancel{n-5}}{\cancel{n-3}} \right) \cdots \left(\frac{\cancel{4}}{\cancel{6}} \right) \left(\frac{\cancel{3}}{\cancel{5}} \right) \left(\frac{\cancel{2}}{\cancel{4}} \right) \left(\frac{\cancel{1}}{\cancel{3}} \right)$$

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

PROOF (cont'd): Suppose \mathbf{S}^* is a min-cut, and suppose we select edges $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n-2}$. Then:

$$\begin{aligned} \Pr[\text{return } \mathbf{S}^*] &= \Pr[\text{none of the } \mathbf{e}_i \text{ cross } \mathbf{S}^*] \\ &= \Pr[\mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_2 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1 \text{ doesn't cross } \mathbf{S}^*] \\ &\quad \times \Pr[\mathbf{e}_3 \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \mathbf{e}_2 \text{ don't cross } \mathbf{S}^*] \\ &\quad \dots \\ &\quad \times \Pr[\mathbf{e}_{n-2} \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{n-3} \text{ don't cross } \mathbf{S}^*] \\ &\geq \left(\frac{2}{n(n-1)} \right) = 1 / \binom{n}{2} \end{aligned}$$

From previous slide:

$$\Pr[\mathbf{e}_j \text{ doesn't cross } \mathbf{S}^* \mid \mathbf{e}_1, \dots, \mathbf{e}_{j-1} \text{ don't cross } \mathbf{S}^*]$$

$$= \frac{n-j-1}{n-j+1}$$



سوال؟

KARGER'S PROBABILITY OF SUCCESS

CLAIM:

The probability that Karger's algorithm returns a minimum cut is $\geq 1 / \binom{n}{2}$

We proved the claim!

But $1 / \binom{n}{2}$ doesn't seem like a very high probability.

For example, with our example of $n = 9$, Karger would return the $\{A, B, C, D\} - \{E, F, G, H, I\}$ min cut with probability $1 / \binom{9}{2} = \mathbf{0.028\dots}$

To boost our chances of success, let's employ some repetition!

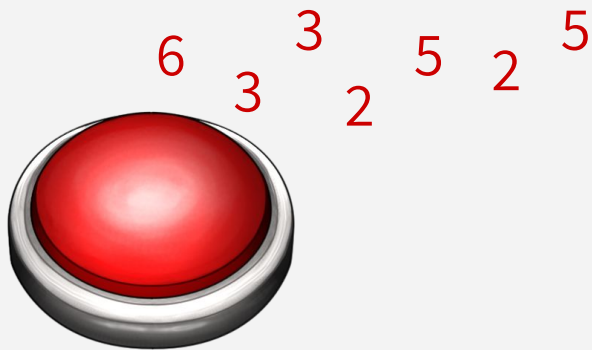
$$\geq \left(\frac{2}{n(n-1)} \right) = 1 / \binom{n}{2}$$

A QUICK PROBABILITY EXERCISE

Suppose we have a magic button that produces one of 5 numbers $\{a,b,c,d,e\}$, uniformly at random when we push it. (We don't know what $\{a,b,c,d,e\}$ are).

Q1: How many times do we need to push the button, in expectation, to see the minimum value?

Q2: What is the probability that you have to push it more than 5 times? 10 times?



A QUICK PROBABILITY EXERCISE

Suppose we have a magic button that produces one of 5 numbers {a,b,c,d,e}, uniformly at random when we push it. (We don't know what {a,b,c,d,e} are).

Q1: How many times do we need to push the button, in expectation, to see the minimum value?

Q2: What is the probability that you have to push it more than 5 times? 10 times?

1. We've done this same calculation a bunch of times:

$$E[\text{\# times we push until min}] = 1/(0.2) = 5$$



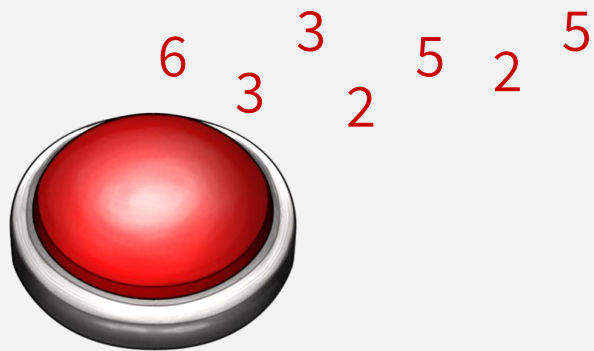
6 3 3 5 2 5
3 2

A QUICK PROBABILITY EXERCISE

Suppose we have a magic button that produces one of 5 numbers {a,b,c,d,e}, uniformly at random when we push it. (We don't know what {a,b,c,d,e} are).

Q1: How many times do we need to push the button, in expectation, to see the minimum value?

Q2: What is the probability that you have to push it more than 5 times? 10 times?



1. We've done this same calculation a bunch of times:

$$E[\text{\# times we push until min}] = 1/(0.2) = 5$$

2. We've done this less frequently:

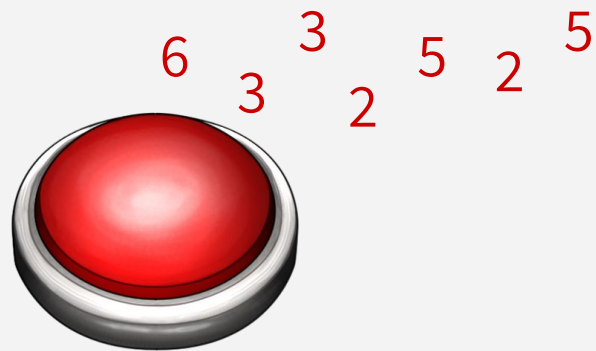
$$\Pr[\text{we don't see min after } T \text{ times}] = (1 - 0.2)^T$$

A QUICK PROBABILITY EXERCISE

Suppose we have a magic button that produces one of 5 numbers {a,b,c,d,e}, uniformly at random when we push it. (We don't know what {a,b,c,d,e} are).

Q1: How many times do we need to push the button, in expectation, to see the minimum value?

Q2: What is the probability that you have to push it more than 5 times? 10 times?



1. We've done this same calculation a bunch of times:

$$E[\text{\# times we push until min}] = \mathbf{1/(0.2) = 5}$$

2. We've done this less frequently:

$$\Pr[\text{we don't see min after } T \text{ times}] = \mathbf{(1 - 0.2)^T}$$

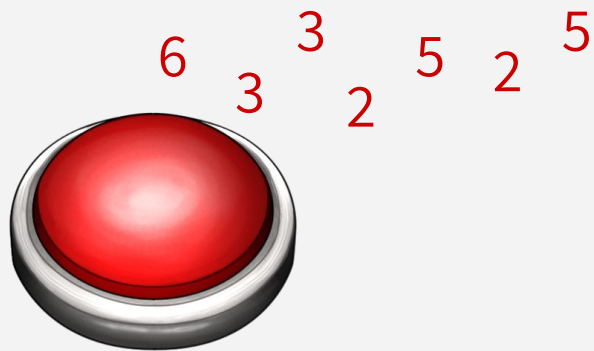
$$\Pr[\text{we don't see min after 5 times}] = \mathbf{(1 - 0.2)^5 \approx 0.33}$$

A QUICK PROBABILITY EXERCISE

Suppose we have a magic button that produces one of 5 numbers {a,b,c,d,e}, uniformly at random when we push it. (We don't know what {a,b,c,d,e} are).

Q1: How many times do we need to push the button, in expectation, to see the minimum value?

Q2: What is the probability that you have to push it more than 5 times? 10 times?



1. We've done this same calculation a bunch of times:

$$E[\# \text{ times we push until min}] = \mathbf{1/(0.2) = 5}$$

2. We've done this less frequently:

$$\Pr[\text{we don't see min after } T \text{ times}] = \mathbf{(1 - 0.2)^T}$$

$$\Pr[\text{we don't see min after 5 times}] = \mathbf{(1 - 0.2)^5 \approx 0.33}$$

$$\Pr[\text{we don't see min after 10 times}] = \mathbf{(1 - 0.2)^{10} \approx 0.1}$$

A QUICK PROBABILITY EXERCISE

**To boost our chances of success,
let's repeat Karger multiple times and return the smallest cut found!**

- Run Karger's! The cut size is 6!
- Run Karger's! The cut size is 3!
- Run Karger's! The cut size is 3!
- Run Karger's! The cut size is 2!
- Run Karger's! The cut size is 5!

$$(1 - 0.2)^T$$

is the probability that
our repetition fails,
i.e. that the minimum value
out of the T tries is *not* the
true min cut value!

$$\Pr[\text{we don't see min after 10 times}] = (1 - 0.2)^{10} \approx 0.1$$

REPEATING KARGER FOR A BOOST!

THE QUESTION:

How many times should we run Karger's to succeed with probability $\geq 1-\delta$?

Here, δ represents an upper bound on the probability of *failure* we're willing to tolerate
(it's a parameter we can choose, e.g. it can be some small constant like 0.01)

REPEATING KARGER FOR A BOOST!

THE QUESTION:

How many times should we run Karger's to succeed with probability $\geq 1-\delta$?

Here, δ represents an upper bound on the probability of *failure* we're willing to tolerate
(it's a parameter we can choose, e.g. it can be some small constant like 0.01)

$\Pr[\text{we fail to return a min-cut after } T \text{ trials}] = (1-p)^T$, where $p = \text{probability a trial succeeds} = 1/\binom{n}{2}$

We want to choose T so that we end up with $(1-p)^T \leq \delta$

REPEATING KARGER FOR A BOOST!

THE QUESTION:

How many times should we run Karger's to succeed with probability $\geq 1-\delta$?

Here, δ represents an upper bound on the probability of *failure* we're willing to tolerate
(it's a parameter we can choose, e.g. it can be some small constant like 0.01)

$\Pr[\text{we fail to return a min-cut after } T \text{ trials}] = (1-p)^T$, where $p = \text{probability a trial succeeds} = 1/\binom{n}{2}$

We want to choose T so that we end up with $(1-p)^T \leq \delta$

- First, we'll make a ~clever~ observation that $1-p \leq e^{-p}$ (if you plot the lines you'll see why!)
- Thus, we have $(1-p)^T \leq e^{-pT}$. We want $e^{-pT} \leq \delta \Rightarrow$ we want $-pT \leq \ln \delta \Rightarrow$ we want $T \geq -(1/p)(\ln \delta)$

REPEATING KARGER FOR A BOOST!

THE QUESTION:

How many times should we run Karger's to succeed with probability $\geq 1-\delta$?

Here, δ represents an upper bound on the probability of *failure* we're willing to tolerate
(it's a parameter we can choose, e.g. it can be some small constant like 0.01)

$\Pr[\text{we fail to return a min-cut after } T \text{ trials}] = (1-p)^T$, where $p = \text{probability a trial succeeds} = 1/\binom{n}{2}$

We want to choose T so that we end up with $(1-p)^T \leq \delta$

- First, we'll make a ~clever~ observation that $1-p \leq e^{-p}$ (if you plot the lines you'll see why!)
- Thus, we have $(1-p)^T \leq e^{-pT}$. We want $e^{-pT} \leq \delta \Rightarrow$ we want $-pT \leq \ln \delta \Rightarrow$ we want $T \geq -(1/p)(\ln \delta)$

$$\binom{n}{2} \ln\left(\frac{1}{\delta}\right)$$

- Choose T to be at least $(1/p)\ln(1/\delta)$, i.e. run Karger's at least _____ times!

KARGER'S: PUTTING IT ALL TOGETHER

Runtime of Karger's (without fancy data structures): **$O(n^2)$**

To successfully find a minimum cut with probability $\geq 1-\delta$,
repeat Karger's at least $\binom{n}{2} \ln(\frac{1}{\delta})$ times.

Treating δ as a constant (e.g. 0.01), the total runtime to succeed w/ prob. $\geq 1-\delta =$

$$O\left(n^2 \cdot \binom{n}{2} \ln\left(\frac{1}{\delta}\right)\right) = \mathbf{O(n^4)}$$

If you want different success probabilities, this overall runtime would be adjusted as well (by adjusting the number of trials needed). Keep in mind that this can be improved by using a union-find data structure as well.

KARGER'S: PUTTING IT ALL TOGETHER

Runtime of Karger's (without fancy data structures): **$O(n^2)$**

This is pretty good!

In practice, it's often good enough to use Monte-Carlo algorithms that'll succeed with probability 0.99. Repetition was the key!!!

Treating δ as a constant (e.g. 0.01), the total runtime to succeed w/ prob. $\geq 1-\delta =$

$$O\left(n^2 \cdot \binom{n}{2} \ln\left(\frac{1}{\delta}\right)\right) = \mathbf{O(n^4)}$$

If you want different success probabilities, this overall runtime would be adjusted as well (by adjusting the number of trials needed). Keep in mind that this can be improved by using a union-find data structure as well.

GENERAL TAKEAWAY

- If we have a Monte Carlo algorithm with a small success probability
- And we can check how good a solution is (e.g. cost of cut)
- Then we can **boost** our success probability via repetition! Repeat it a bunch of times and take the best solution

Unfortunately, repeating an $O(n^2)$ time algorithm many times can still be quite expensive...

CAN WE DO BETTER?

Absolutely!!!

The **Karger-Stein** algorithm can achieve a better runtime of **$O(n^2 \log^2 n)$** .

THE KEY: We'll do the repetitions in a clever way so that we save on runtime!

Note: this is a trickier algorithm, so we'll just sketch the high-level approach here.



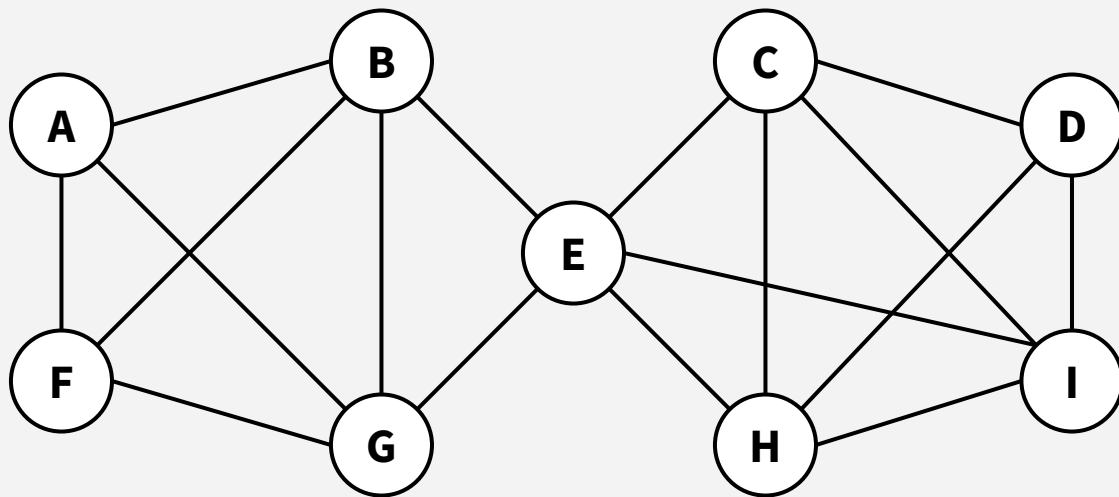
سوال؟

الگوریتم کارگر-استین

راه بهتری برای تکرار الگوریتم کارگر

KARGER'S ALGORITHM (AGAIN)

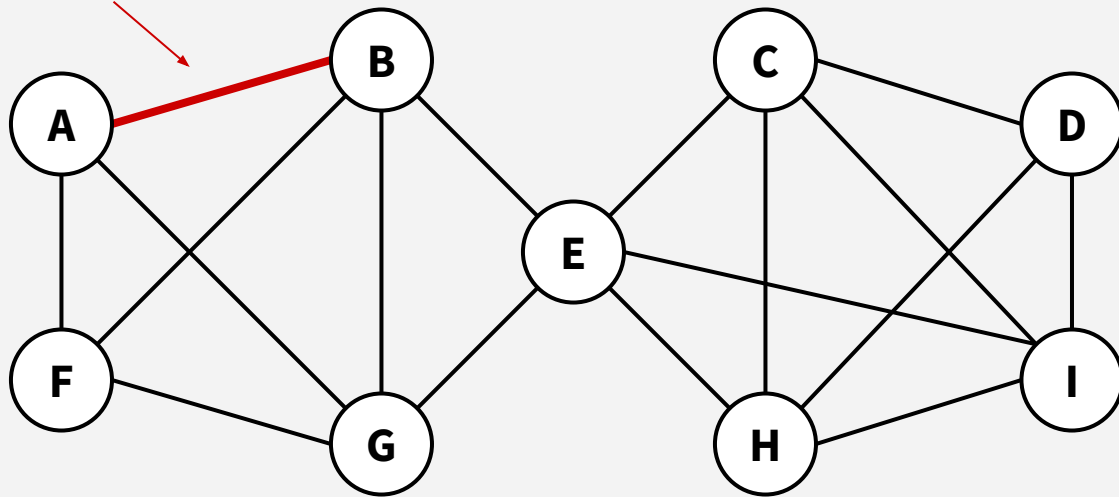
Let's brainstorm how we might save on repetitions...



KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Pick a random edge!



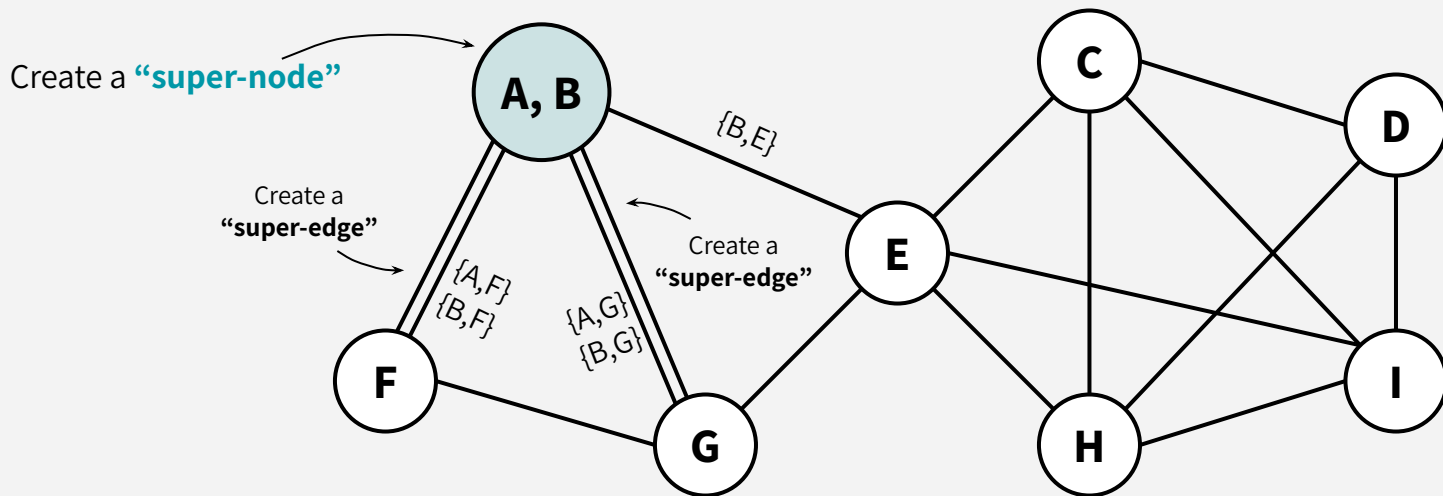
There are 17 edges,
and 15 are good to
contract

Probability that we didn't mess up = **15/17**

KARGER'S ALGORITHM (AGAIN)

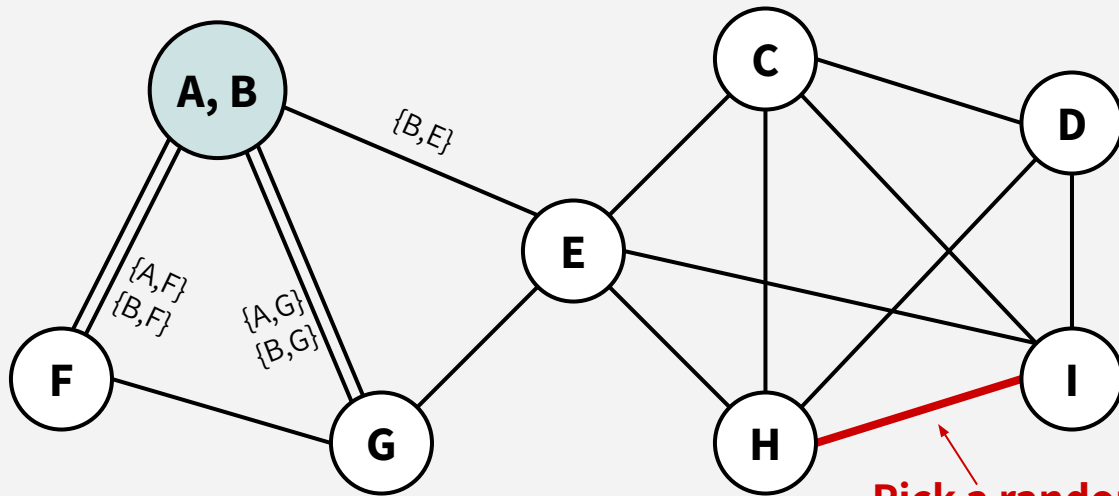
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Now perform an “edge contraction”!



KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

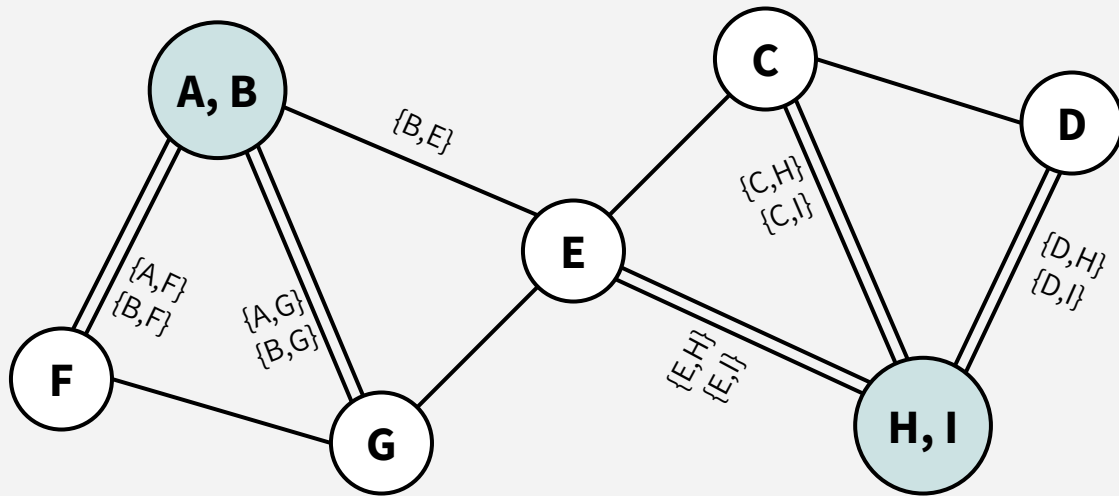


Now there are only 16 edges, since the edge between A and B is gone.

Probability that we didn't mess up = **14/16**

KARGER'S ALGORITHM (AGAIN)

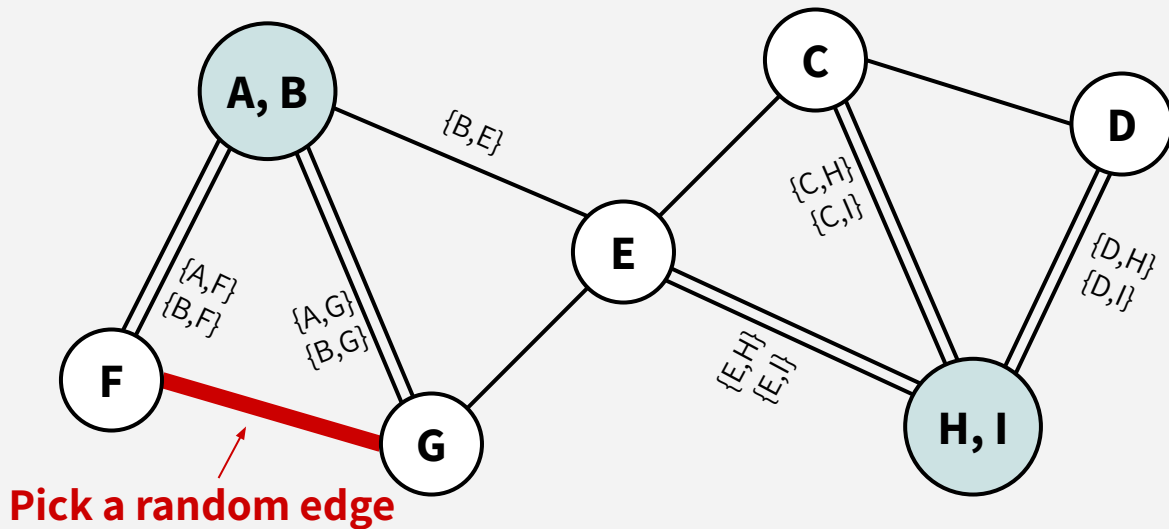
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Now perform an “edge contraction”!

KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



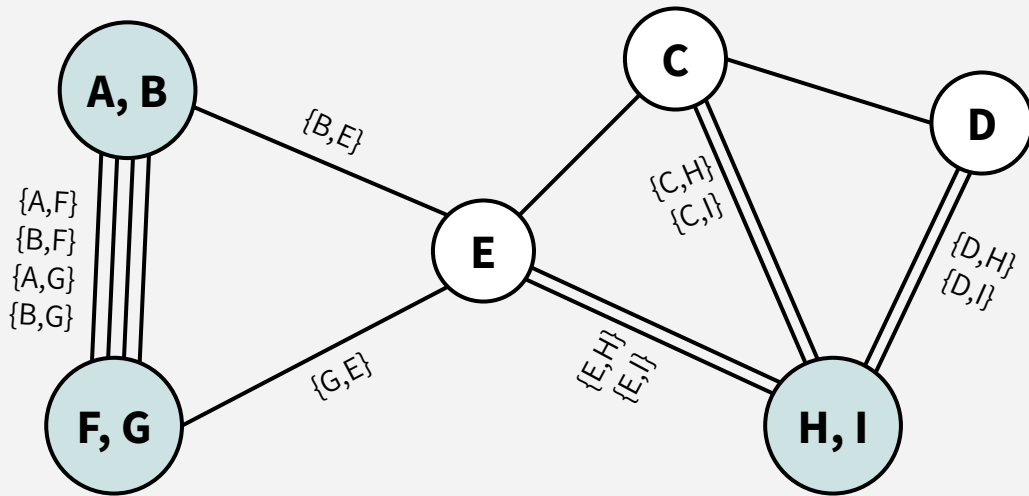
There are 15 edges, since the edge between H and I is gone now.

Again, only 2 edges are bad to pick still.

Probability that we didn't mess up = **13/15**

KARGER'S ALGORITHM (AGAIN)

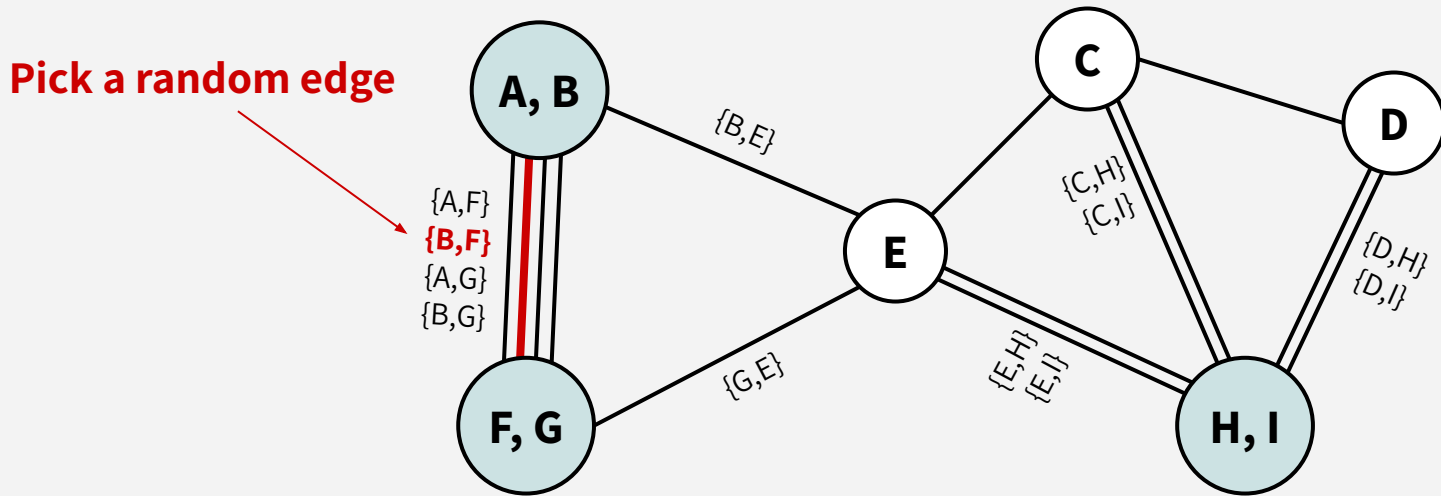
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Perform an “edge contraction”

KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



There are now 14 edges, since the edge between F and G is gone now.

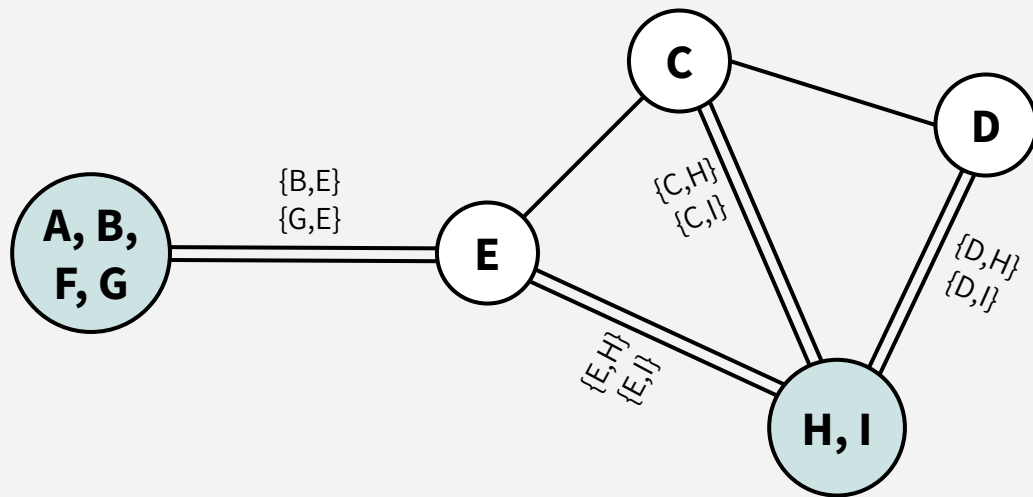
Again, only 2 edges are bad to pick still.

Probability that we didn't mess up = **12/14**

KARGER'S ALGORITHM (AGAIN)

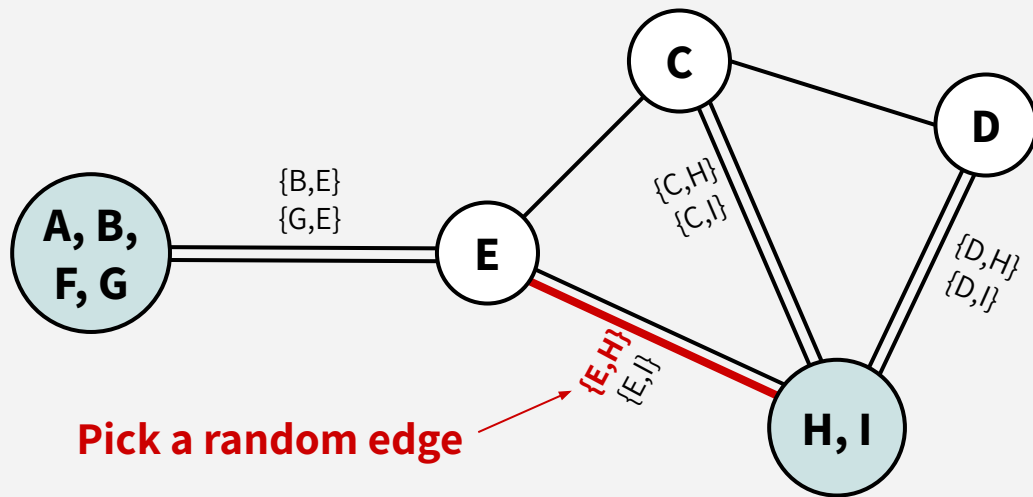
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Perform an
“edge contraction”



KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



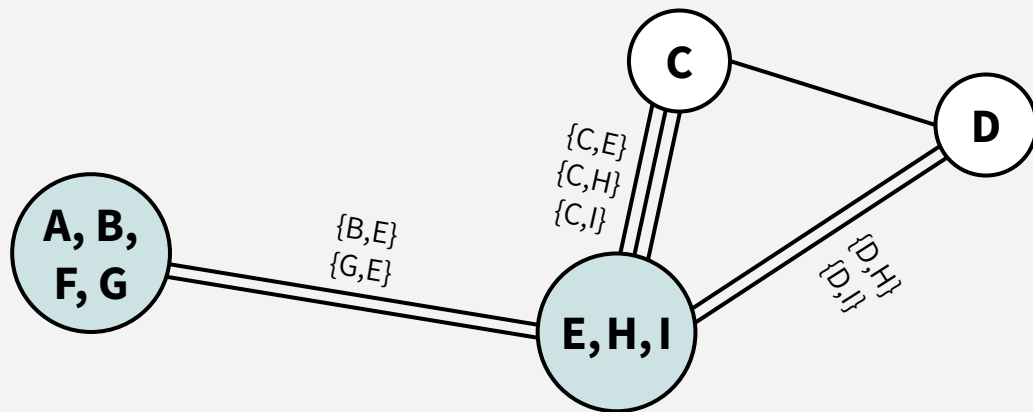
There are now only 10 edges, since the last contraction got rid of 4 edges!

Again, only 2 edges are bad to pick still.

Probability that we didn't mess up = **8/10**

KARGER'S ALGORITHM (AGAIN)

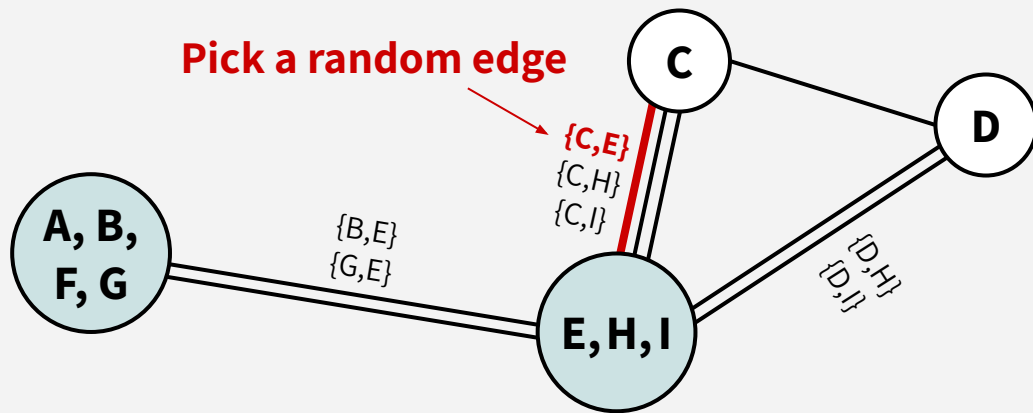
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



Perform an “edge contraction”

KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



There are now only 8 edges, since the last contraction got rid of 2 edges!

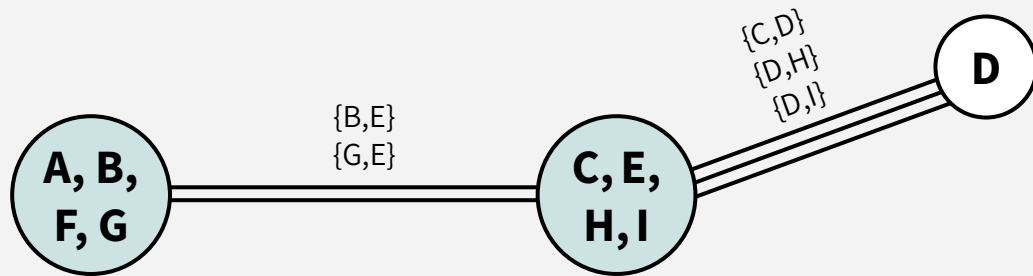
Again, only 2 edges are bad to pick still.

Probability that we didn't mess up = $6/8$

KARGER'S ALGORITHM (AGAIN)

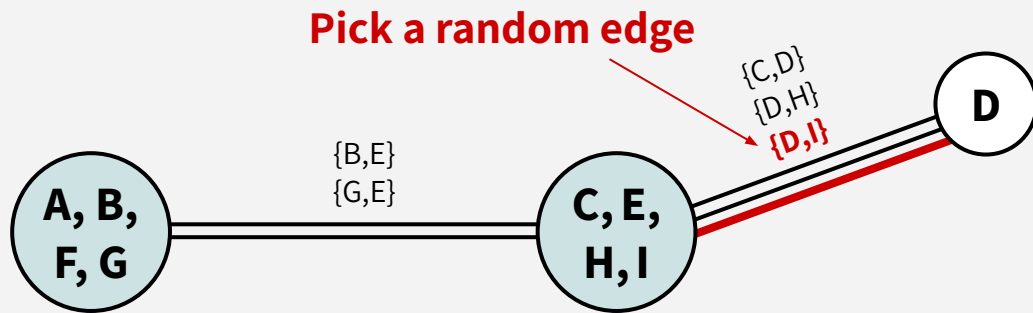
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Perform an “edge contraction”



KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.



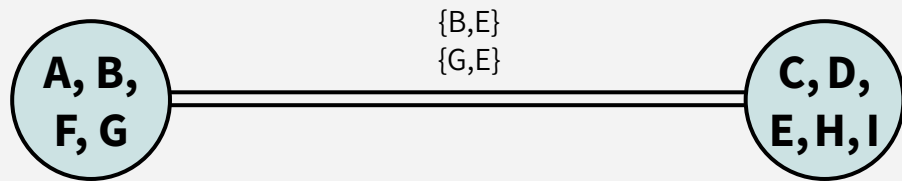
There are now only 5 edges, since the last contraction got rid of 3 edges!

Again, only 2 edges are bad to pick still.

Probability that we didn't mess up = $\frac{3}{5}$

KARGER'S ALGORITHM (AGAIN)

Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

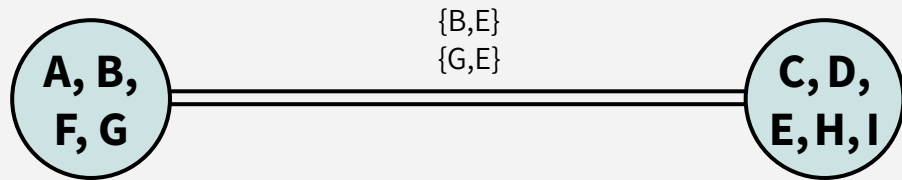


Perform an “edge contraction”

KARGER'S ALGORITHM (AGAIN)

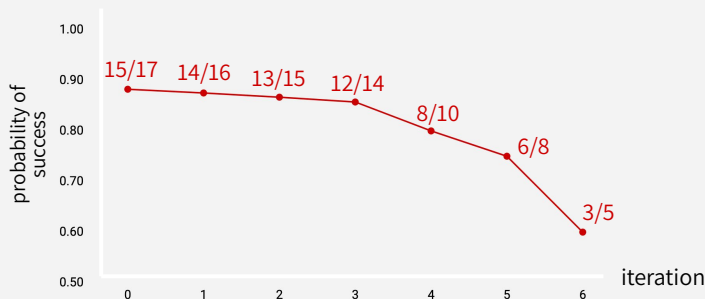
Pick a random edge, **contract** it, and repeat until you only have 2 vertices left.

Here we stop because we only have 2 nodes left!



MORE LIKELY TO MESS UP OVER TIME!

At the beginning of a run of Karger, it's pretty likely our random choice is fine.
The probability that we mess up gets worse and worse over time.



Moral of the walkthrough:

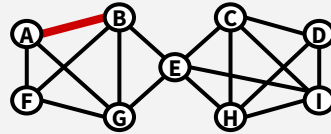
Repeating the beginning of the algorithm is wasteful!!! We weren't really that likely to mess up, so why bother protecting those choices with redundancy?
(i.e. why bother using repetition for our earlier choices?)



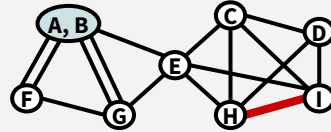
سوال؟

OUR GAMEPLAN (IN PICTURES)

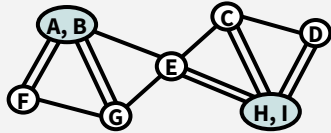
First run Karger's on G for a bit...



Contract!

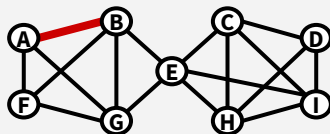


Contract!

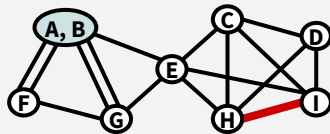


OUR GAMEPLAN (IN PICTURES)

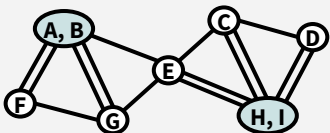
First run Karger's on G for a bit...



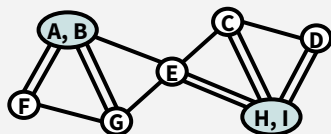
Contract!



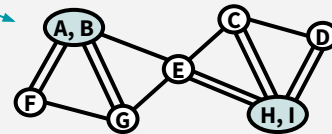
Contract!



ATTEMPT A



ATTEMPT B

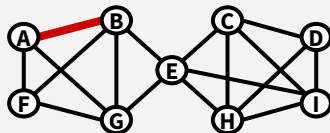


Now FORK!!!

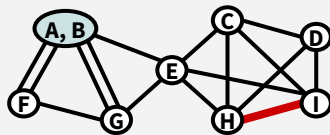


OUR GAMEPLAN (IN PICTURES)

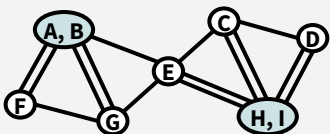
First run Karger's on G for a bit...



Contract!

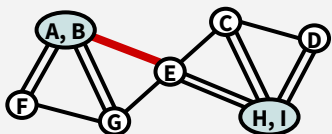


Contract!

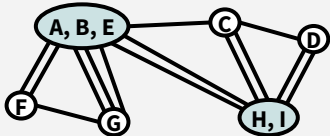


ATTEMPT A

This branch made a bad choice 😞

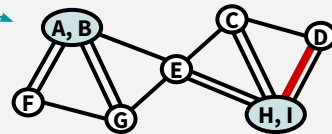


Contract!

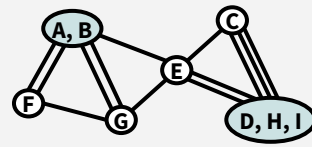


ATTEMPT B

But it's okay since branch B made a good choice 😊



Contract!

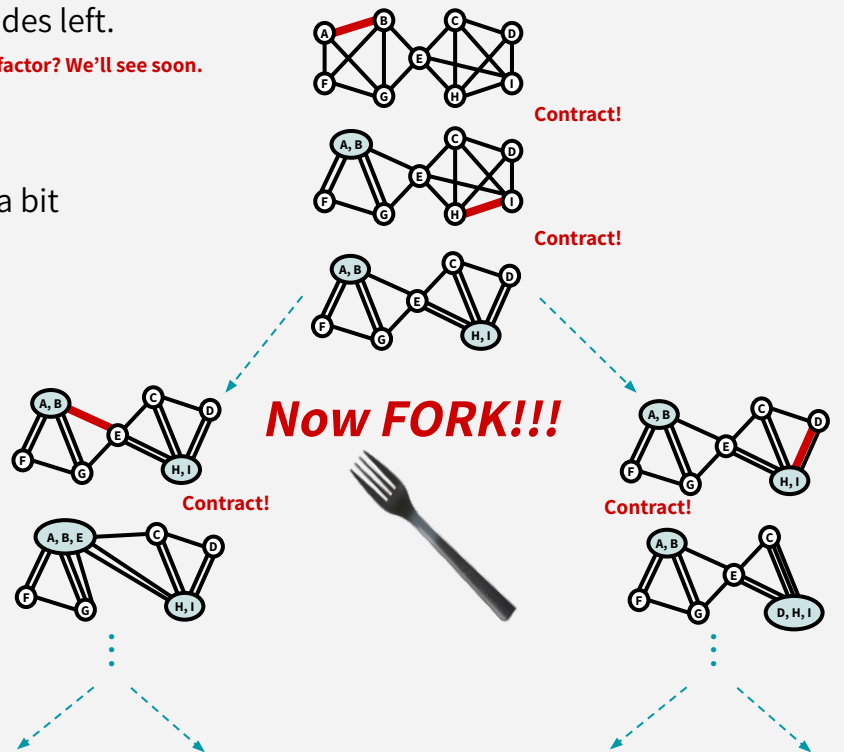


Now FORK!!!



OUR GAMEPLAN (IN WORDS)

- First run Karger's on G for a bit until there are $\frac{n}{\sqrt{2}}$ supernodes left.
Why the $\sqrt{2}$ factor? We'll see soon.
- Then split/fork into two independent copies, G_1 and G_2
- Independently resume Karger's algorithm on G_1 and G_2 for a bit
(until you reduce # of supernodes by another factor of $\sqrt{2}$)
- Then split each of those into two independent copies....
- Etc....

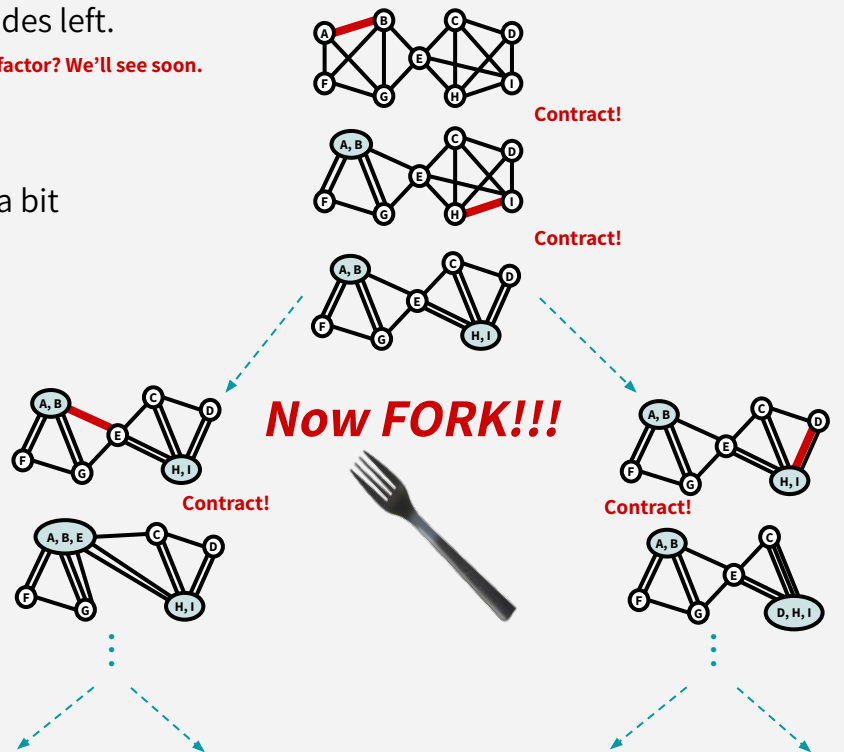


OUR GAMEPLAN (IN WORDS)

- First run Karger's on G for a bit until there are $\frac{n}{\sqrt{2}}$ supernodes left.
Why the $\sqrt{2}$ factor? We'll see soon.
- Then split/fork into two independent copies, G_1 and G_2
- Independently resume Karger's algorithm on G_1 and G_2 for a bit (until you reduce # of supernodes by another factor of $\sqrt{2}$)
- Then split each of those into two independent copies....
- Etc....

Basically:

Run Kargers for a bit, split into two copies, recurse on each, then return whichever copy's answer was better (i.e. the smaller cut)



KARGER-STEIN PSEUDOCODE

There are some details left out, but here's the skeleton:

KARGER-STEIN($G=(V,E)$):

- $n \leftarrow |V|$
- if $n < 4$:
 find a min-cut by brute force $\leftarrow O(1)$ time
- Run Karger's algorithm on G with until $\lfloor \frac{n}{\sqrt{2}} \rfloor$ supernodes remain
- G_1 & $G_2 \leftarrow$ copies of what's left in G
- $S_1 = \text{KARGER-STEIN}(G_1)$
- $S_2 = \text{KARGER-STEIN}(G_2)$
- **return** whichever of S_1, S_2 is the smaller cut

KARGER-STEIN (AS RECURSION TREE)

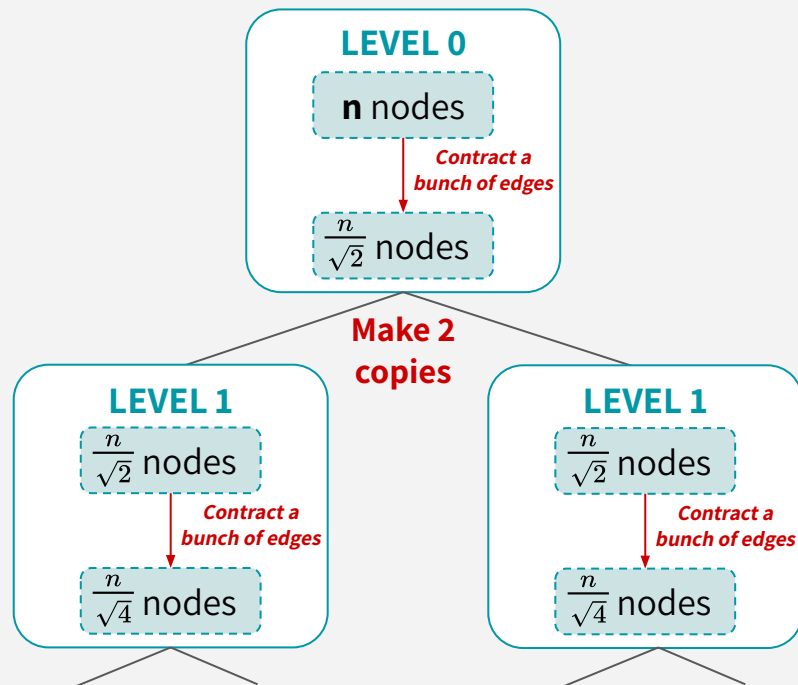
SHRINKING FACTOR: $\sqrt{2}$

DEPTH: $\log_{\sqrt{2}}(n) = \frac{\log n}{\log \sqrt{2}} = 2 \log n$

OF LEAVES: $2^{2 \log n} = n^2$

KARGER-STEIN($G=(V,E)$):

- $n \leftarrow |V|$
- if $n < 4$:
 find a min-cut by brute force
- Run Karger's algorithm on G until $\lfloor \frac{n}{\sqrt{2}} \rfloor$ supernodes remain
- G_1 & $G_2 \leftarrow$ copies of what's left in G
- $S_1 = \text{KARGER-STEIN}(G_1)$
- $S_2 = \text{KARGER-STEIN}(G_2)$
- **return** whichever of S_1, S_2 is the smaller cut

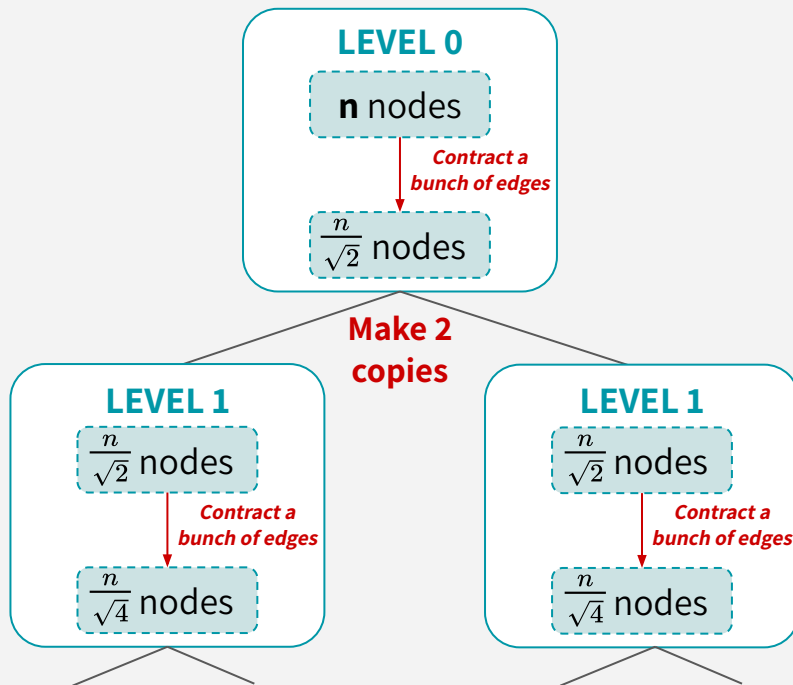


KARGER-STEIN RUNTIME

SHRINKING FACTOR: $\sqrt{2}$

DEPTH: $\log_{\sqrt{2}}(n) = \frac{\log n}{\log \sqrt{2}} = 2 \log n$

OF LEAVES: $2^{2 \log n} = n^2$



- Amount of work per level = amount of work needed to reduce the number of nodes by a factor of $\sqrt{2}$
- That's at most $O(n^2)$ -- that's the time it takes to run Karger's algorithm once in full!
- Our recurrence relation is

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

- The Master Theorem says:

$$T(n) = O(n^2 \log n)$$

WHY $n/\sqrt{2}$?

Suppose we contract $n - \frac{n}{\sqrt{2}}$ edges. CLAIM:

$\Pr[\text{None of those } n - \frac{n}{\sqrt{2}} \text{ edges cross } S^*] \approx \mathbf{1/2}$ (when n is large)

WHY $n/\sqrt{2}$?

Suppose we contract $n - \frac{n}{\sqrt{2}}$ edges. CLAIM:

$\Pr[\text{None of those } n - \frac{n}{\sqrt{2}} \text{ edges cross } S^*] \approx 1/2$ (when n is large)

Suppose S^* is a min-cut, and suppose we select edges e_1, e_2, \dots, e_{n-t} . Then:

$\Pr[\text{None of those } n-t \text{ edges cross } S^*]$

$$\begin{aligned} &= \Pr[e_1 \text{ doesn't cross } S^*] \\ &\times \Pr[e_2 \text{ doesn't cross } S^* \mid e_1 \text{ doesn't cross } S^*] \\ &\times \Pr[e_3 \text{ doesn't cross } S^* \mid e_1, e_2 \text{ don't cross } S^*] \\ &\dots \\ &\times \Pr[e_{n-t} \text{ doesn't cross } S^* \mid e_1, \dots, e_{n-t-1} \text{ don't cross } S^*] \end{aligned}$$

From earlier:

$\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$

$$= \frac{n-j-1}{n-j+1}$$

WHY $n/\sqrt{2}$?

Suppose we contract $n - \frac{n}{\sqrt{2}}$ edges. CLAIM:

$\Pr[\text{None of those } n - \frac{n}{\sqrt{2}} \text{ edges cross } S^*] \approx 1/2$ (when n is large)

Suppose S^* is a min-cut, and suppose we select edges e_1, e_2, \dots, e_{n-t} . Then:

$\Pr[\text{None of those } n-t \text{ edges cross } S^*]$

$$\begin{aligned} &= \Pr[e_1 \text{ doesn't cross } S^*] \\ &\times \Pr[e_2 \text{ doesn't cross } S^* \mid e_1 \text{ doesn't cross } S^*] \\ &\times \Pr[e_3 \text{ doesn't cross } S^* \mid e_1, e_2 \text{ don't cross } S^*] \\ &\dots \\ &\times \Pr[e_{n-t} \text{ doesn't cross } S^* \mid e_1, \dots, e_{n-t-1} \text{ don't cross } S^*] \end{aligned}$$

From earlier:

$\Pr[e_j \text{ doesn't cross } S^* \mid e_1, \dots, e_{j-1} \text{ don't cross } S^*]$

$$= \frac{n-j-1}{n-j+1}$$

$$\geq \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \left(\frac{n-5}{n-3}\right) \dots \left(\frac{t+1}{t+3}\right) \left(\frac{t}{t+2}\right) \left(\frac{t-1}{t+1}\right)$$

WHY $n/\sqrt{2}$?

Suppose we contract $n - \frac{n}{\sqrt{2}}$ edges. CLAIM:

$\Pr[\text{None of those } n - \frac{n}{\sqrt{2}} \text{ edges cross } S^*] \approx \mathbf{1/2}$ (when n is large)

Suppose S^* is a min-cut, and suppose we select edges e_1, e_2, \dots, e_{n-t} . Then:

$\Pr[\text{None of those } n-t \text{ edges cross } S^*]$

$$\begin{aligned} &\geq \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \left(\frac{n-5}{n-3}\right) \cdots \left(\frac{t+1}{t+3}\right) \left(\frac{t}{t+2}\right) \left(\frac{t-1}{t+1}\right) \\ &= \frac{t \cdot (t-1)}{n \cdot (n-1)} \quad \left(\text{now choose } t = \frac{n}{\sqrt{2}}\right) \\ &= \frac{\frac{n}{\sqrt{2}} \cdot \left(\frac{n}{\sqrt{2}} - 1\right)}{n \cdot (n-1)} \approx \frac{1}{2} \quad (\text{when } n \text{ is large}) \end{aligned}$$

WHY $n/\sqrt{2}$?

Suppose we contract $n - \frac{n}{\sqrt{2}}$ edges. CLAIM:

$\Pr[\text{None of those } n - \frac{n}{\sqrt{2}} \text{ edges cross } S^*] \approx 1/2$ (when n is large)

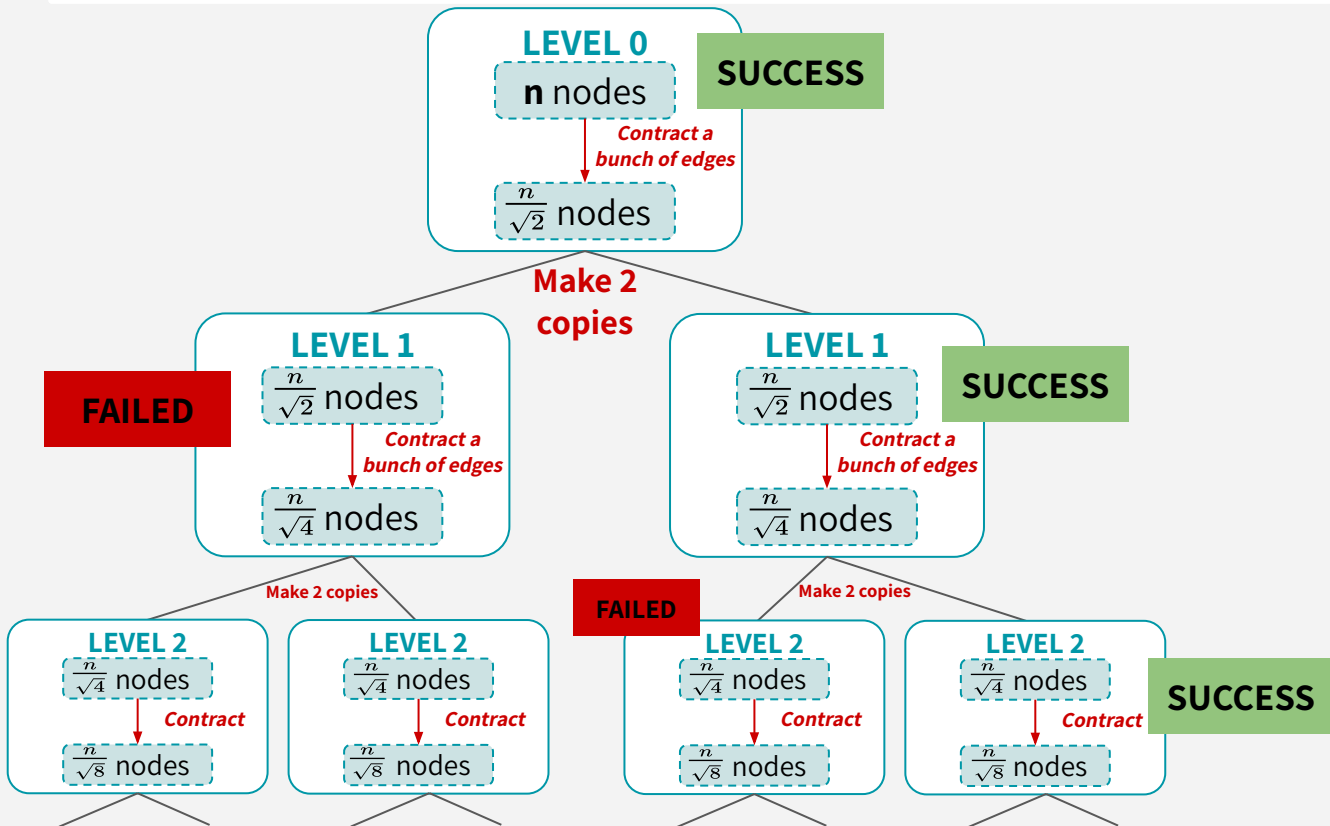
Suppose S^* is a min-cut, and suppose we select edges e_1, e_2, \dots, e_{n-t} . Then:

$\Pr[\text{None of those } n-t \text{ edges cross } S^*]$

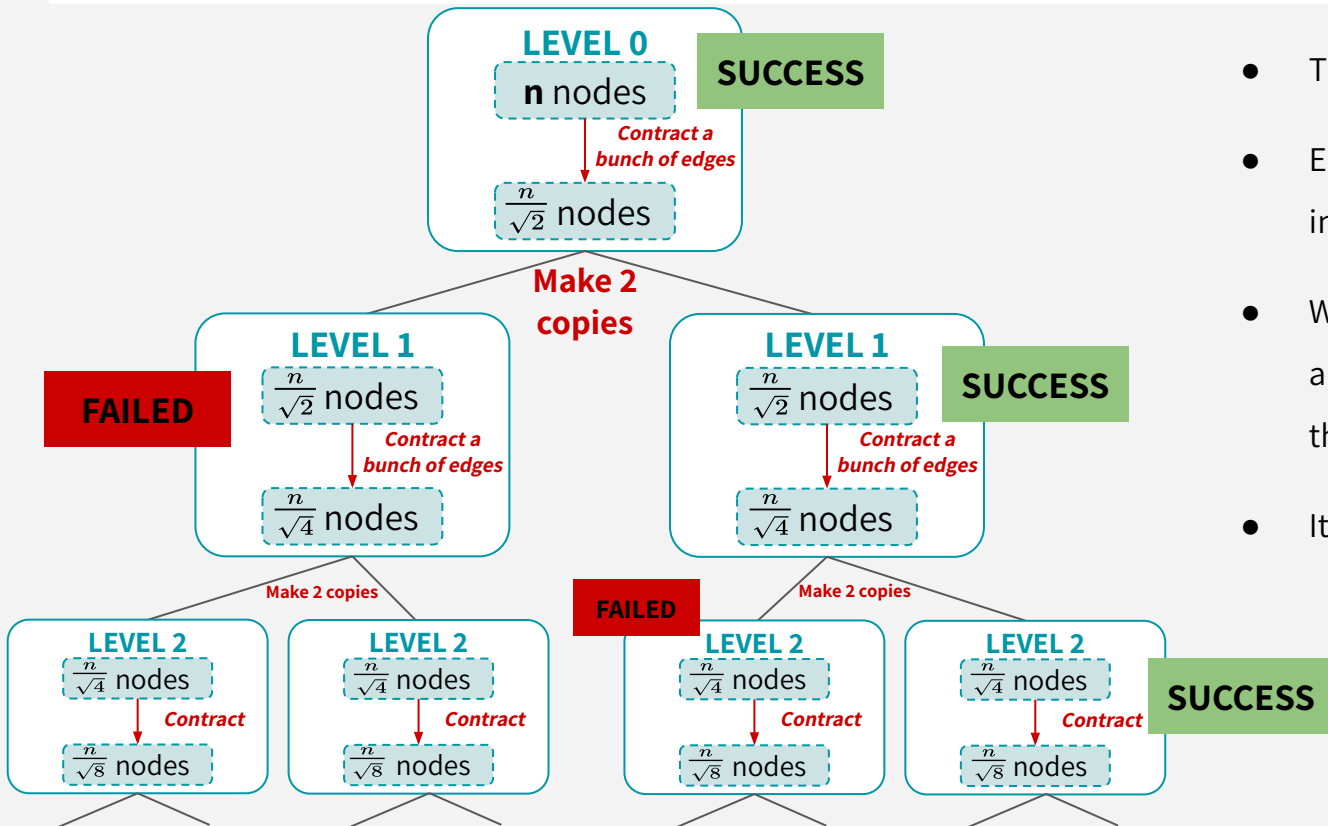
This means that in every “blob” of running Karger for a bit, the probability of success is $1/2$!

$$= \frac{\frac{n}{\sqrt{2}} \cdot (\frac{n}{\sqrt{2}} - 1)}{n \cdot (n-1)} \approx \frac{1}{2} \quad (\text{when } n \text{ is large})$$

KARGER-STEIN PROB. OF SUCCESS



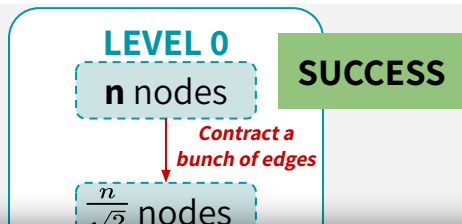
KARGER-STEIN PROB. OF SUCCESS



- T is a binary tree of depth $2 \log n$
- Each node of T succeeds or fails independently with probability $\frac{1}{2}$
- What is the probability that there's a path from the root to any leaf that's entirely successful?
- It turns out that this is

$$\Omega\left(\frac{1}{\log n}\right)$$

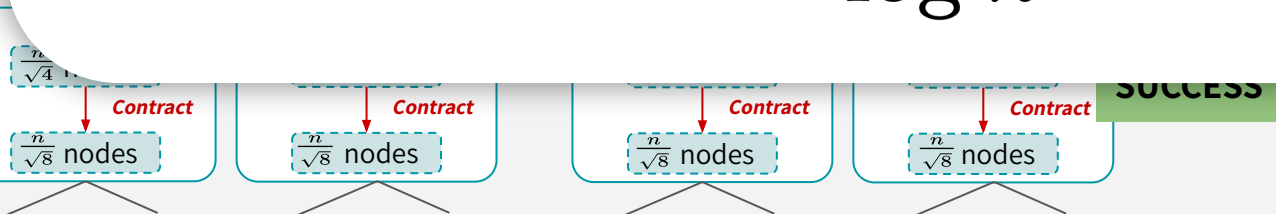
KARGER-STEIN PROB. OF SUCCESS



- T is a binary tree of depth $2 \log n$
- Each node of T succeed or fails independently with probability $\frac{1}{2}$

The probability of success for one run of KARGER-STEIN is

$$\Omega\left(\frac{1}{\log n}\right)$$



ALTOGETHER NOW

- The probability of success for one run of KARGER-STEIN is $\Omega\left(\frac{1}{\log n}\right)$
- We can amplify the success probability with repetition
 - Run Karger-Stein $O\left((\log n) \cdot \left(\log \frac{1}{\delta}\right)\right)$ times to achieve success probability $1 - \delta$
- Each iteration takes time $O(n^2 \log n)$
 - That's what we proved before.
- Choosing $\delta = 0.01 \rightarrow$ the total runtime is

$$O(n^2 \log n \cdot \log n) = O(n^2 \log^2 n)$$



سوال؟