



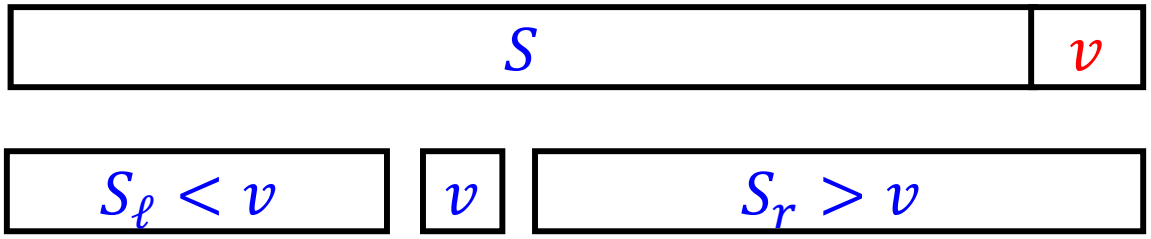
# Chapter 6
# Randomization

## Algorithm Theory
## WS 2014/15

Fabian Kuhn

# Randomized Quicksort

**Quicksort:**




**function** Quick ($S$: sequence): sequence;

{returns the sorted sequence $S$}

**begin**

**if** $\#S \leq 1$ then **return** $S$

**else** { choose pivot element $v$ in $S$;

partition $S$ into $S_\ell$ with elements $< v$,

and $S_r$ with elements $> v$

**return** | Quick($S_\ell$) | $v$ | Quick($S_r$) |

**end**;

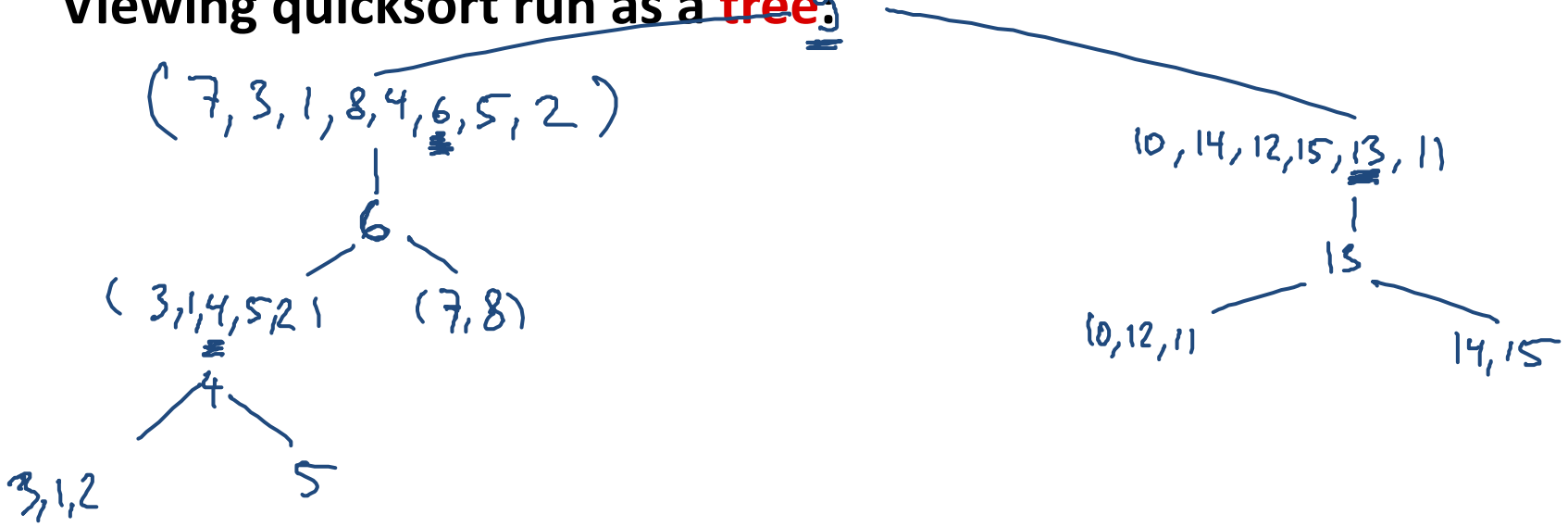
$T(n)$: exp. # comparisons to sort $n$ elements

$T(n) \leq 2n \cdot \ln(n)$

# Alternative Analysis

Array to sort: [ 7 , 3 , 1 , 10 , 14 , 8 , 12 , 9 , 4 , 6 , 5 , 15 , 2 , 13 , 11 ]

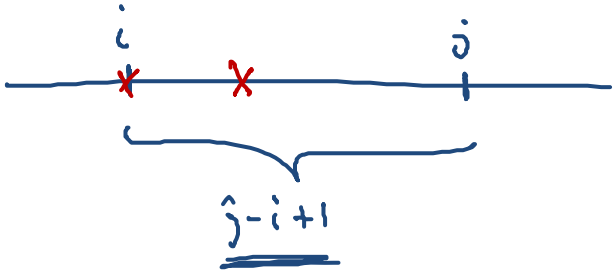
**Viewing quicksort run as a tree:**

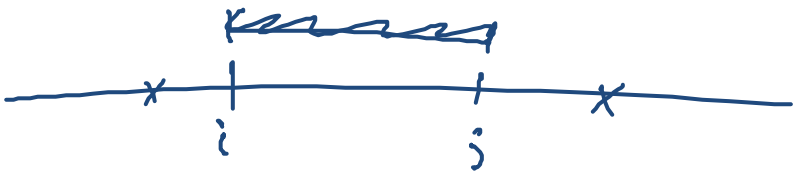


# Comparisons

- Comparisons are only between pivot and non-pivot elements
- Every element can only be the pivot once:
  → every 2 elements can only be compared once!

- W.l.o.g., assume that the elements to sort are $1,2,\dots,n$
- Elements $i$ and $j$ are compared if and only if either $i$ or $j$ is a pivot before any element $h$: $i < h < j$ is chosen as pivot
  - i.e., iff $i$ is an ancestor of $j$ or $j$ is an ancestor of $i$




$$\mathbb{P}(\text{comparison betw. } i \text{ and } j) = \frac{2}{j-i+1}$$

# Counting Comparisons

$\binom{4}{2}$

Random variable for every pair of elements $(i, j)$:

$$X_{ij} = \begin{cases} 1, & \text{if there is a comparison between } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbb{P}(X_{ij} = 1) = \frac{2}{j-i+1} \qquad \mathbb{E}[X_{ij}] = \frac{2}{j-i+1}$$

Number of comparisons: $X$ we need $\mathbb{E}[X]$

$$X = \sum_{i<j} X_{ij}$$

- What is $\mathbb{E}[X]$?

# Randomized Quicksort Analysis

**Theorem:** The expected number of comparisons when sorting $n$ elements using randomized quicksort is $T(n) \leq 2n \ln n$.

**Proof:**

- **Linearity of expectation:**
  For all random variables $X_1, \dots, X_n$ and all $a_1, \dots, a_n \in \mathbb{R}$,

$$\mathbb{E}\left[\sum_i a_i X_i\right] = \sum_i a_i \mathbb{E}[X_i].$$

$X = \sum_{i<j} X_{ij}$ $\quad \mathbb{E}[X] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathbb{E}[X_{ij}]$

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

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

# Randomized Quicksort Analysis

**Theorem:** The expected number of comparisons when sorting $n$ elements using randomized quicksort is $T(n) \leq 2n \ln n$.

**Proof:**

$$\mathbb{E}[X] = 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{j-i+1} = 2\sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{1}{k}$$

$$\leq 2\sum_{i=1}^{n-1}\sum_{k=2}^{n}\frac{1}{k}$$

$H(n)-1 \leq \ln(n)$

Harmonic series

$$H(n) = \sum_{i=1}^{n}\frac{1}{i}$$

$H(n) \leq 1 + \ln(n)$

$$\leq 2(n-1)\ln(n)$$

# Types of Randomized Algorithms

**Las Vegas Algorithm:**

- always a correct solution
- running time is a random variable

- **Example:** randomized quicksort, contention resolution

**Monte Carlo Algorithm:**

- probabilistic correctness guarantee (**m**ostly **c**orrect)
- fixed (deterministic) running time

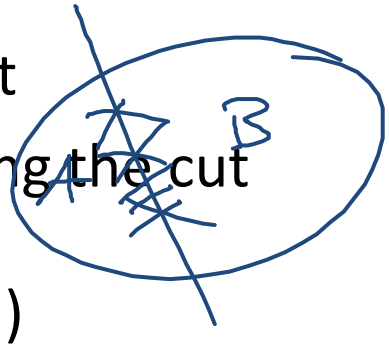
- **Example:** primality test

# Minimum Cut

**Reminder:** Given a graph $G = (V, E)$, a cut is a partition $(A, B)$ of $V$ such that $V = A \cup B$, $A \cap B = \emptyset$, $A, B \neq \emptyset$

**Size of the cut $(A, B)$:** # of edges crossing the cut

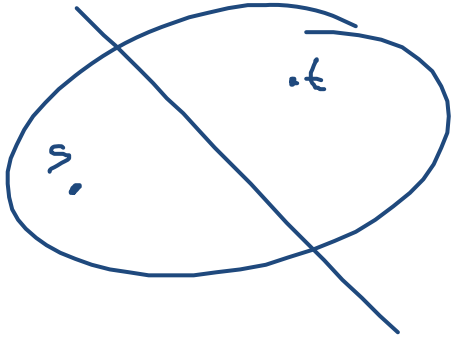
- For weighted graphs, total edge weight crossing the cut




**Goal:** Find a cut of minimal size (i.e., of size $\lambda(G)$)

**Maximum-flow based algorithm:**

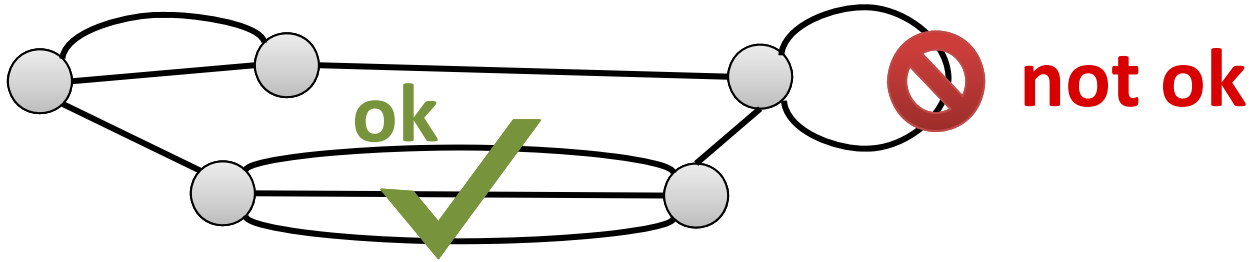
- Fix $s$, compute min $s$-$t$-cut for all $t \neq s$
- $O\big(m \cdot \lambda(G)\big) = O(mn)$ per $s$-$t$ cut
- Gives an $O\big(mn\lambda(G)\big) = O(mn^2)$-algorithm




**Best-known deterministic algorithm:** $O(mn + n^2 \log n)$

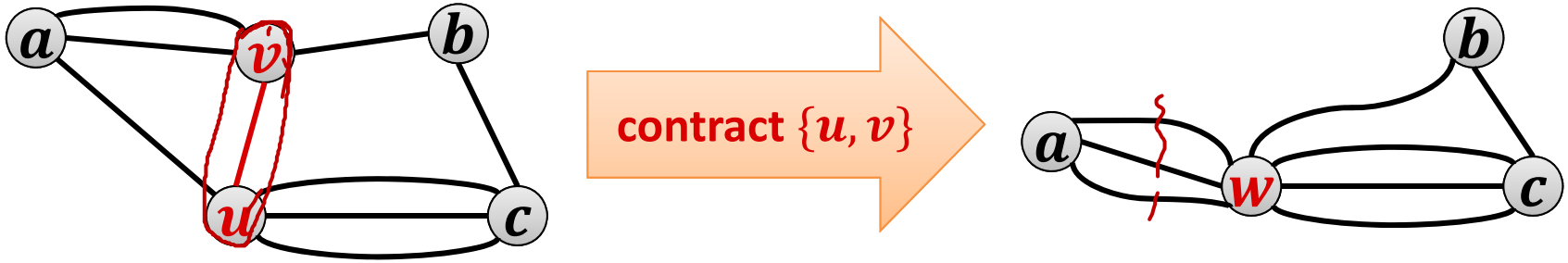
# Edge Contractions

- In the following, we consider multi-graphs that can have multiple edges (but no self-loops)




**Contracting edge $\{u, v\}$:**

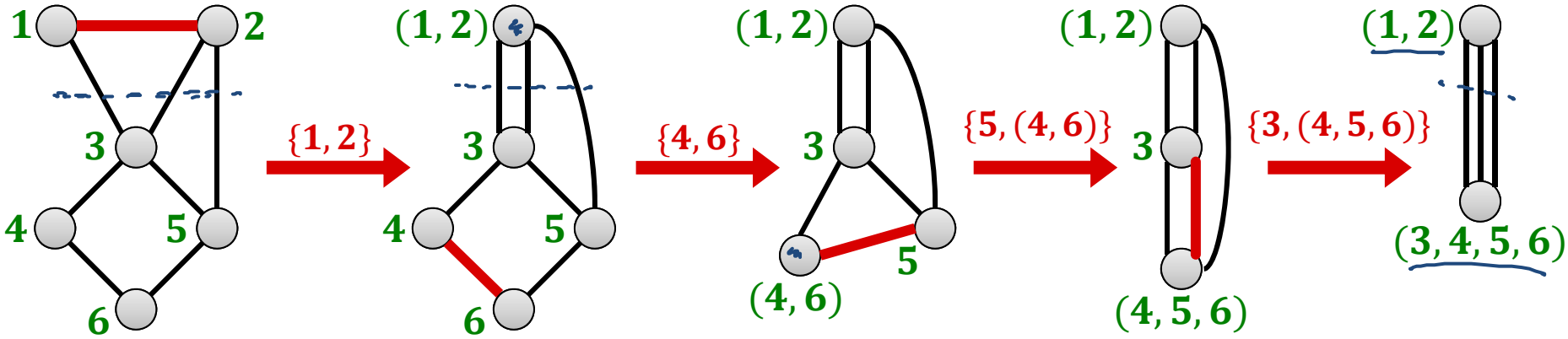
- Replace nodes $u$, $v$ by new node $w$
- For all edges $\{u, x\}$ and $\{v, x\}$, add an edge $\{w, x\}$
- Remove self-loops created at node $w$



# Properties of Edge Contractions

**Nodes:**

- After contracting $\{u, v\}$, the new node represents $u$ and $v$
- After a series of contractions, each node represents a subset of the original nodes




**Cuts:**

- Assume in the contracted graph, $w$ represents nodes $S_w \subset V$
- The edges of a node $w$ in a contracted graph are in a one-to-one correspondence with the edges crossing the cut $(S_w, V \setminus S_w)$

# Randomized Contraction Algorithm

**Algorithm:**

**while** there are $> 2$ nodes **do**

contract a uniformly random edge

**return** cut induced by the last two remaining nodes

(cut defined by the original node sets represented by the last 2 nodes)

**Theorem:** The random contraction algorithm returns a minimum cut with probability at least $1/O(n^2)$.

- We will show this next.

**Theorem:** The random contraction algorithm can be implemented in time $O(n^2)$.

- There are $n-2$ contractions, each can be done in time $O(n)$.
- You will show this in the exercises.

# Contractions and Cuts

**Lemma:** If two original nodes $u, v \in V$ are merged into the same node of the contracted graph, there is a path connecting $u$ and $v$ in the original graph s.t. all edges on the path are contracted.

**Proof:**

- Contracting an edge $\{x, y\}$ merges the node sets represented by $x$ and $y$ and does not change any of the other node sets.
- The claim the follows by induction on the number of edge contractions.

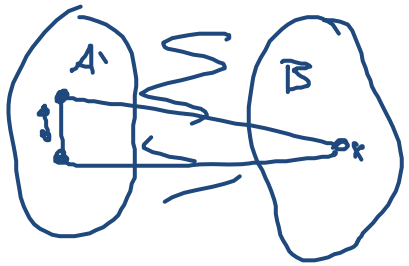
# Contractions and Cuts

**Lemma:** During the contraction algorithm, the edge connectivity (i.e., the size of the min. cut) cannot get smaller.

**Proof:**

- All cuts in a (partially) contracted graph correspond to cuts of the same size in the original graph $G$ as follows:
  - For a node $u$ of the contracted graph, let $S_u$ be the set of original nodes that have been merged into $u$ (the nodes that $u$ represents)
  - Consider a cut $(A, B)$ of the contracted graph
  - $(A', B')$ with

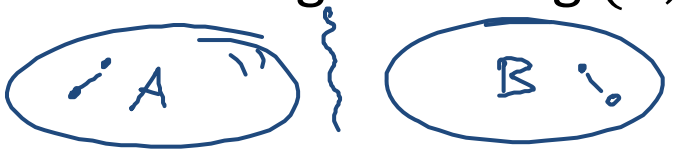
$$A' := \bigcup_{u \in A} S_u\,, \qquad B' := \bigcup_{v \in B} S_v$$




    is a cut of $G$.
  - The edges crossing cut $(A, B)$ are in one-to-one correspondence with the edges crossing cut $(A', B')$.

# Contraction and Cuts

**Lemma:** The contraction algorithm outputs a cut $(A, B)$ of the input graph $G$ if and only if it never contracts an edge crossing $(A, B)$.




**Proof:**

1. If an edge crossing $(A, B)$ is contracted, a pair of nodes $u \in A$, $v \in V$ is merged into the same node and the algorithm outputs a cut different from $(A, B)$.

2. If no edge of $(A, B)$ is contracted, no two nodes $u \in A$, $v \in B$ end up in the same contracted node because every path connecting $u$ and $v$ in $G$ contains some edge crossing $(A, B)$

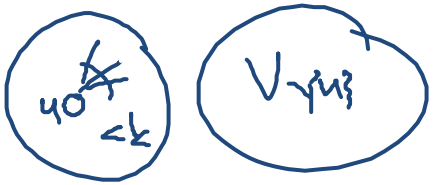
   In the end there are only 2 sets → output is $(A, B)$

# Getting The Min Cut

**Theorem:** The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$.

To prove the theorem, we need the following claim:

**Claim:** If the minimum cut size of a multigraph $G$ (no self-loops) is $k$, $G$ has at least $kn/2$ edges.

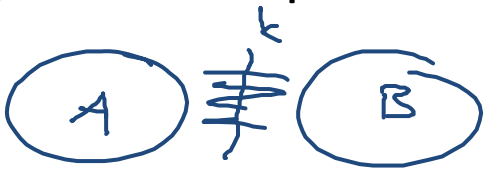


**Proof:**

- Min cut has size $k \Longrightarrow$ all nodes have degree $\geq k$
  - A node $v$ of degree $< k$ gives a cut $(\{v\}, V \setminus \{v\})$ of size $< k$
- Number of edges $m = {}^1/_2 \cdot \sum_v \deg(v) \geq \frac{1}{2} \cdot n \cdot k$

# Getting The Min Cut

**Theorem:** The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$.




**Proof:**

- Consider a fixed min cut $(A, B)$, assume $(A, B)$ has size $k$
- The algorithm outputs $(A, B)$ iff none of the $k$ edges crossing $(A, B)$ gets contracted.
- Before contraction $i$, there are $n + 1 - i$ nodes
  → and thus $\geq (n + 1 - i)k/2$ edges
- If no edge crossing $(A, B)$ is contracted before, the probability to contract an edge crossing $(A, B)$ in step $i$ is at most

$$\frac{k}{\frac{(n+1-i)k}{2}} = \frac{2}{n+1-i}.$$

# Getting The Min Cut

**Theorem:** The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$.

**Proof:**

- If no edge crossing $(A, B)$ is contracted before, the probability to contract an edge crossing $(A, B)$ in step $i$ is at most $2/n+1-i$.
- Event $\mathcal{E}_i$: edge contracted in step $i$ is **not** crossing $(A, B)$

Goal: $\mathbb{P}(\text{Alg. returns } (A,B)) = \mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3 \cap \dots \cap \mathcal{E}_{n-2})$

$= \mathbb{P}(\mathcal{E}_1) \cdot \mathbb{P}(\mathcal{E}_2 \mid \mathcal{E}_1) \cdot \mathbb{P}(\mathcal{E}_3 \mid \mathcal{E}_1 \cap \mathcal{E}_2) \cdots \mathbb{P}(\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \dots \cap \mathcal{E}_{n-3})$

$\mathbb{P}(\mathcal{E}_i \mid \mathcal{E}_1 \cap \dots \cap \mathcal{E}_{i-1}) \geq 1 - \frac{2}{n+1-i}$

# Getting The Min Cut

**Theorem:** The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$.

**Proof:**

- $\mathbb{P}(\mathcal{E}_{i+1} | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_i) \geq 1 - \frac{2}{n-i} = \frac{n-2-i}{n-i}$
- No edge crossing $(A, B)$ contracted: event $\mathcal{E} = \bigcap_{i=1}^{n-2} \mathcal{E}_i$

$$\mathbb{P}(\mathcal{E}_1 \cap \dots \cap \mathcal{E}_{n-2}) = \mathbb{P}(\mathcal{E}_1) \cdot \mathbb{P}(\mathcal{E}_2 | \mathcal{E}_1) \cdot \dots \cdot \mathbb{P}(\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \dots \cap \mathcal{E}_{n-3})$$

$$\geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \frac{n-5}{n-3} \cdot \dots \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}$$

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

# Randomized Min Cut Algorithm

**Theorem:** If the contraction algorithm is repeated $O(n^2 \log n)$ times, one of the $O(n^2 \log n)$ instances returns a min. cut w.h.p.

**Proof:**

$1 + x < e^x$

- Probability to not get a minimum cut in $c \cdot \binom{n}{2} \cdot \ln n$ iterations:

$(e^a)^b = e^{a \cdot b}$

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{c \cdot \binom{n}{2} \cdot \ln n} \leq e^{-c \ln n} = \frac{1}{n^c}$$

$1 - x < e^{-x}$

$< e^{-\frac{1}{\binom{n}{2}} \cdot c \cdot \binom{n}{2} \cdot \ln n}$

**Corollary:** The contraction algorithm allows to compute a minimum cut in $O(n^4 \log n)$ time w.h.p.

- Each instance can be implemented in $O(n^2)$ time. ($O(n)$ time per contraction)

# Can We Do Better?

- Time $O(n^4 \log n)$ is not very spectacular, a simple max flow based implementation has time $O(n^4)$.

However, we will see that the contraction algorithm is nevertheless very interesting because:

1. The algorithm can be improved to beat every known deterministic algorithm.

2. It allows to obtain strong statements about the distribution of cuts in graphs.

# Better Randomized Algorithm

**Recall:**

- Consider a fixed min cut $(A, B)$, assume $(A, B)$ has size $k$
- The algorithm outputs $(A, B)$ iff none of the $k$ edges crossing $(A, B)$ gets contracted.
- Throughout the algorithm, the edge connectivity is at least $k$ and therefore each node has degree $\geq k$
- Before contraction $i$, there are $n + 1 - i$ nodes and thus at least $(n + 1 - i)k/2$ edges
- If no edge crossing $(A, B)$ is contracted before, the probability to contract an edge crossing $(A, B)$ in step $i$ is at most

$$\frac{k}{\frac{(n+1-i)k}{2}} = \frac{2}{n+1-i}.$$

# Improving the Contraction Algorithm

- For a specific min cut $(A, B)$, if $(A, B)$ survives the first $i$ contractions,

$$\mathbb{P}(\text{edge crossing } (A, B) \text{ in contraction } i+1) \leq \frac{2}{n-i}.$$

- **Observation:** The probability only gets large for large $i$
- **Idea:** The early steps are much safer than the late steps.

  Maybe we can repeat the late steps more often than the early ones.

# Safe Contraction Phase

**Lemma:** A given min cut $(A, B)$ of an $n$-node graph $G$ survives the first $n - \left\lceil n/\sqrt{2} + 1 \right\rceil$ contractions, with probability $> 1/2$.

**Proof:**

- Event $\mathcal{E}_i$: cut $(A, B)$ survives contraction $i$
- Probability that $(A, B)$ survives the first $n - t$ contractions:

$$\geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \dots \cdot \frac{t}{t+2} \cdot \frac{t-1}{t+1} = \frac{t(t-1)}{n(n-1)}$$

$$t = \lceil n/\sqrt{2} + 1 \rceil \geq \frac{n}{\sqrt{2}} + 1$$

$$= \frac{t}{n} \cdot \frac{t-1}{n-1} \geq \frac{n/\sqrt{2}+1}{n} \cdot \frac{n/\sqrt{2}}{n-1} > \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2}$$

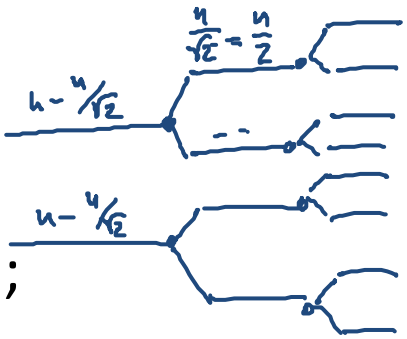
# Better Randomized Algorithm

**Let's simplify a bit:**

- Pretend that $n/\sqrt{2}$ is an integer (for all $n$ we will need it).
- Assume that a given min cut survives the first $n - {}^{n}/_{\sqrt{2}}$ contractions with probability $\geq {}^{1}/_{2}$.

**contract$(G, t)$:**

- Starting with $n$-node graph $G$, perform $n - t$ edge contractions such that the new graph has $t$ nodes.




**mincut$(G)$:**

1. $X_1 := \text{mincut}\left(\text{contract}\left(G, n/\sqrt{2}\right)\right)$;
2. $X_2 := \text{mincut}\left(\text{contract}\left(G, n/\sqrt{2}\right)\right)$;
3. **return** $\min\{X_1, X_2\}$;

# Success Probability

**mincut($G$):**

1. $X_1 := \text{mincut}\left(\text{contract}\left(G, n/\sqrt{2}\right)\right)$;
2. $X_2 := \text{mincut}\left(\text{contract}\left(G, n/\sqrt{2}\right)\right)$;
3. **return** $\min\{X_1, X_2\}$;

$\boldsymbol{P(n)}$**:** probability that the above algorithm returns a min cut when applied to a graph with $n$ nodes.

- Probability that $X_1$ is a min cut $\geq \frac{1}{2} \cdot P(n/2)$

**Recursion:**

$$P(n) \geq 1 - \left(1 - \frac{1}{2}P(n/2)\right)^2 = P(n/2) - \frac{1}{4} \cdot P(n/2)^2$$

$P(2) = 1$

# Success Probability

**Theorem:** The recursive randomized min cut algorithm returns a minimum cut with probability at least $1/\log_2 n$.

**Proof** (by induction on $n$):

$$P(n) = P\left(\frac{n}{\sqrt{2}}\right) - \frac{1}{4} \cdot P\left(\frac{n}{\sqrt{2}}\right)^2, \qquad P(2) = 1$$