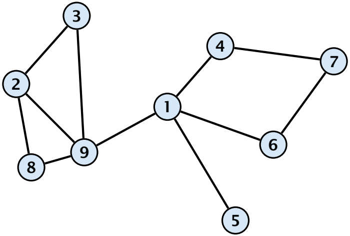
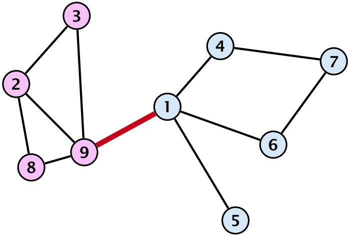
15 Global Mincut

Given an undirected, capacitated graph $G = (V, E, c)$ find a partition of V into two non-empty sets $S, V \setminus S$ s.t. the capacity of edges between both sets is minimized.



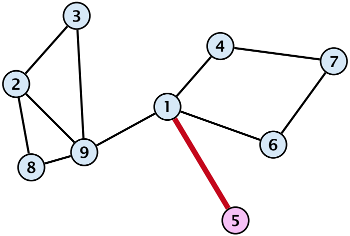
15 Global Mincut

Given an undirected, capacitated graph $G = (V, E, c)$ find a partition of V into two non-empty sets $S, V \setminus S$ s.t. the capacity of edges between both sets is minimized.



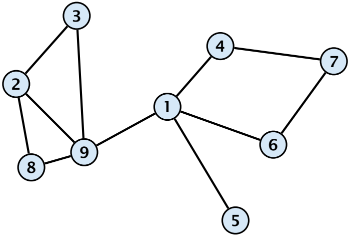
15 Global Mincut

Given an undirected, capacitated graph $G = (V, E, c)$ find a partition of V into two non-empty sets $S, V \setminus S$ s.t. the capacity of edges between both sets is minimized.



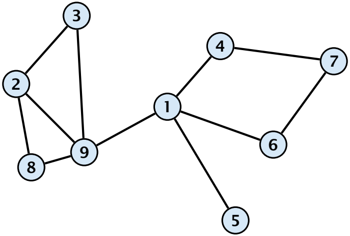
15 Global Mincut

Given an undirected, capacitated graph $G = (V, E, c)$ find a partition of V into two non-empty sets $S, V \setminus S$ s.t. the capacity of edges between both sets is minimized.



15 Global Mincut

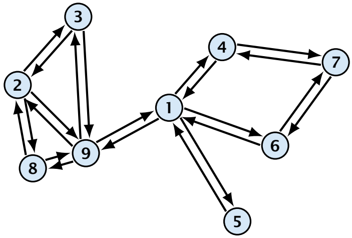
We can solve this problem using standard maxflow/mincut.



15 Global Mincut

We can solve this problem using standard maxflow/mincut.

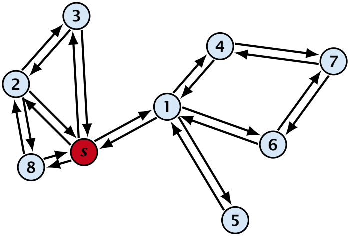
- Construct a directed graph $G' = (V, E')$ that has edges (u, v) and (v, u) for every edge $\{u, v\} \in E$.



15 Global Mincut

We can solve this problem using standard maxflow/mincut.

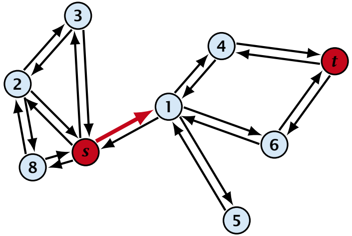
- Construct a directed graph $G' = (V, E')$ that has edges (u, v) and (v, u) for every edge $\{u, v\} \in E$.
- Fix an arbitrary node $s \in V$ as source. Compute a minimum s-t cut for all possible choices $t \in V, t \neq s$. (Time: $\mathcal{O}(n^4)$)



15 Global Mincut

We can solve this problem using standard maxflow/mincut.

- Construct a directed graph $G' = (V, E')$ that has edges (u, v) and (v, u) for every edge $\{u, v\} \in E$.
- Fix an arbitrary node $s \in V$ as source. Compute a minimum s-t cut for all possible choices $t \in V, t \neq s$. (Time: $\mathcal{O}(n^4)$)
- Let $(S, V \setminus S)$ be a minimum global mincut. The above algorithm will output a cut of capacity $\text{cap}(S, V \setminus S)$ whenever $|\{s, t\} \cap S| = 1$.



Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by “identifying” u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

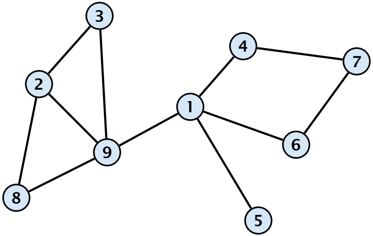
Example 1

- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

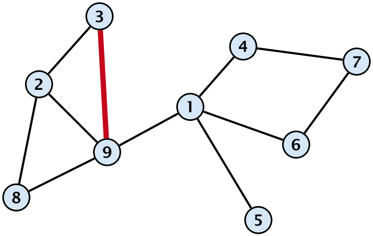


- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

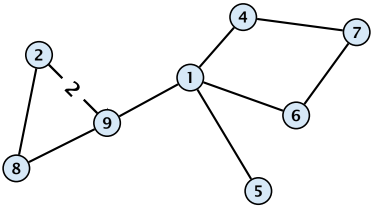


- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by "identifying" u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1

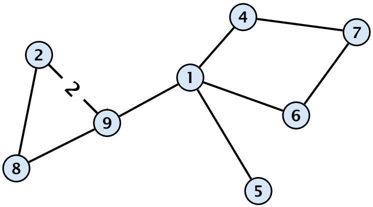


- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

- Given a graph $G = (V, E)$ and an edge $e = \{u, v\}$.
- The graph G/e is obtained by “identifying” u and v to form a new node.
- Resulting parallel edges are replaced by a single edge, whose capacity equals the sum of capacities of the parallel edges.

Example 1



- Edge-contractions do no decrease the size of the mincut.

Edge Contractions

We can perform an edge-contraction in time $\mathcal{O}(n)$.

Randomized Mincut Algorithm

Algorithm 1 KargerMincut($G = (V, E, c)$)

1: **for** $i = 1 \to n - 2$ **do**
2: choose $e \in E$ randomly with probability $c(e)/c(E)$
3: $G \leftarrow G/e$
4: **return** only cut in G

Randomized Mincut Algorithm

Algorithm 1 KargerMincut($G = (V, E, c)$)

```
for i = 1 → n − 2 do
    choose e ∈ E randomly with probability c(e)/c(E)
    G ← G/e
return only cut in G
```

- Let G_t denote the graph after the $(n-t)$-th iteration, when t nodes are left.

Randomized Mincut Algorithm

Algorithm 1 KargerMincut($G = (V, E, c)$)

1: **for** $i = 1 \to n - 2$ **do**
2: choose $e \in E$ randomly with probability $c(e)/c(E)$
3: $G \leftarrow G/e$
4: **return** only cut in G

- Let G_t denote the graph after the $(n-t)$-th iteration, when t nodes are left.
- Note that the final graph G_2 only contains a single edge.

Randomized Mincut Algorithm

Algorithm 1 KargerMincut($G = (V, E, c)$)

1: **for** $i = 1 \to n - 2$ **do**
2: choose $e \in E$ randomly with probability $c(e)/c(E)$
3: $G \leftarrow G/e$
4: **return** only cut in G

- Let G_t denote the graph after the $(n-t)$-th iteration, when t nodes are left.
- Note that the final graph G_2 only contains a single edge.
- The cut in G_2 corresponds to a cut in the original graph G with the same capacity.

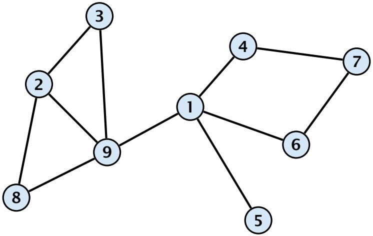
Randomized Mincut Algorithm

Algorithm 1 KargerMincut($G = (V, E, c)$)

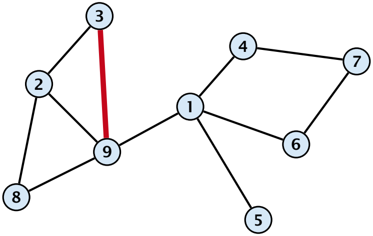
```
for i = 1 → n − 2 do
    choose e ∈ E randomly with probability c(e)/c(E)
    G ← G/e
return only cut in G
```

- Let G_t denote the graph after the $(n-t)$-th iteration, when t nodes are left.
- Note that the final graph G_2 only contains a single edge.
- The cut in G_2 corresponds to a cut in the original graph G with the same capacity.
- What is the probability that this algorithm returns a mincut?

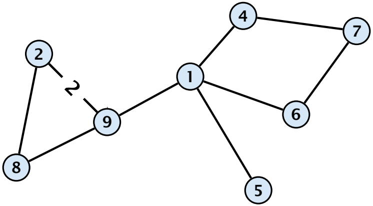
Example: Randomized Mincut Algorithm



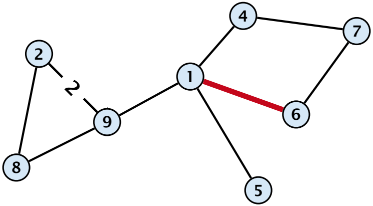
Example: Randomized Mincut Algorithm



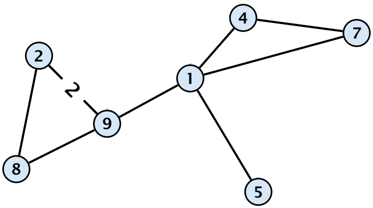
Example: Randomized Mincut Algorithm



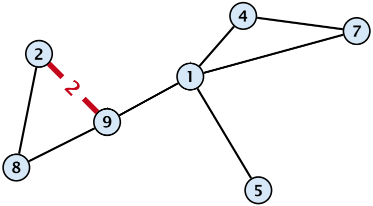
Example: Randomized Mincut Algorithm



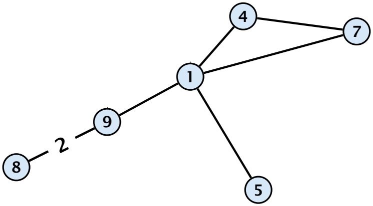
Example: Randomized Mincut Algorithm



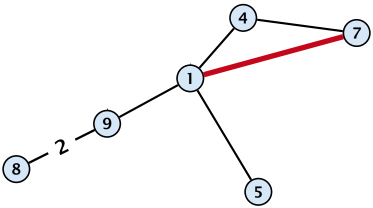
Example: Randomized Mincut Algorithm



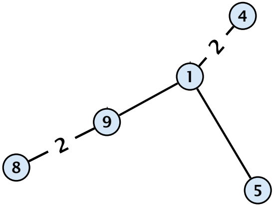
Example: Randomized Mincut Algorithm



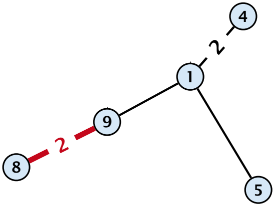
Example: Randomized Mincut Algorithm



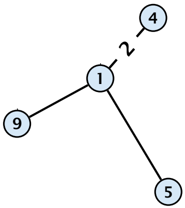
Example: Randomized Mincut Algorithm



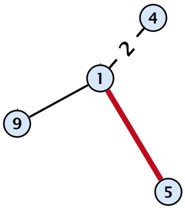
Example: Randomized Mincut Algorithm



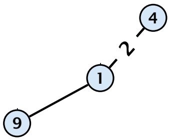
Example: Randomized Mincut Algorithm



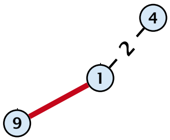
Example: Randomized Mincut Algorithm



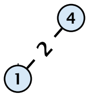
Example: Randomized Mincut Algorithm



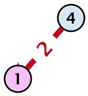
Example: Randomized Mincut Algorithm



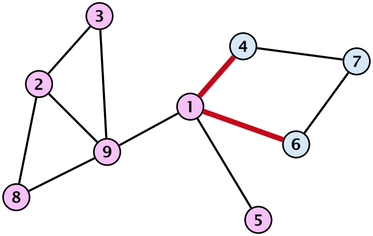
Example: Randomized Mincut Algorithm



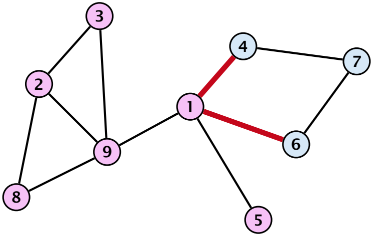
Example: Randomized Mincut Algorithm



Example: Randomized Mincut Algorithm



Example: Randomized Mincut Algorithm



What is the probability that this algorithm returns a mincut?

Analysis

What is the probability that a given mincut A is still possible after round i?

- It is still possible to obtain cut A in the end if so far no edge in $(A, V \setminus A)$ has been contracted.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \text{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\text{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\text{cap}(v) \geq \min$.
- Summing $\text{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \text{cap}(v) \geq (n-i+1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n-i+1)$.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \operatorname{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\operatorname{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\operatorname{cap}(v) \geq \min$.
- Summing $\operatorname{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \operatorname{cap}(v) \geq (n-i+1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n-i+1)$.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \text{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\text{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\text{cap}(v) \geq \min$.
- Summing $\text{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \text{cap}(v) \geq (n - i + 1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n - i + 1)$.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \text{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\text{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\text{cap}(v) \geq \min$.
- Summing $\text{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \text{cap}(v) \geq (n - i + 1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n - i + 1)$.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \text{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\text{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\text{cap}(v) \geq \min$.
- Summing $\text{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \text{cap}(v) \geq (n - i + 1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n - i + 1)$.

Analysis

What is the probability that we select an edge from A in iteration i?

- Let $\min = \text{cap}(A, V \setminus A)$ denote the capacity of a mincut.
- Let $\text{cap}(v)$ be capacity of edges incident to vertex $v \in V_{n-i+1}$.
- Clearly, $\text{cap}(v) \geq \min$.
- Summing $\text{cap}(v)$ over all edges gives

$$2c(E) = 2\sum_{e \in E} c(e) = \sum_{v \in V} \text{cap}(v) \geq (n - i + 1) \cdot \min$$

- Hence, the probability of choosing an edge from the cut is at most $\min / c(E) \leq 2/(n - i + 1)$.

Analysis

The probability that we do not choose an edge from the cut in iteration i is

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

The probability that the cut is alive after iteration $n-t$ (after which t nodes are left) is

$$\prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} = \frac{t(t-1)}{n(n-1)} \ .$$

Choosing $t = 2$ gives that with probability $1/\binom{n}{2}$ the algorithm computes a mincut.

Analysis

The probability that we do not choose an edge from the cut in iteration i is

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

The probability that the cut is alive after iteration $n-t$ (after which t nodes are left) is

$$\prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} = \frac{t(t-1)}{n(n-1)} \ .$$

Choosing $t=2$ gives that with probability $1/\binom{n}{2}$ the algorithm computes a mincut.

Analysis

The probability that we do not choose an edge from the cut in iteration i is

$$1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1} \enspace .$$

The probability that the cut is alive after iteration $n-t$ (after which t nodes are left) is

$$\prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} = \frac{t(t-1)}{n(n-1)} \enspace .$$

Choosing $t=2$ gives that with probability $1/\binom{n}{2}$ the algorithm computes a mincut.

Analysis

Repeating the algorithm $c \ln n \binom{n}{2}$ times gives that the probability that we are never successful is

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2} c \ln n} \le \left(e^{-1/\binom{n}{2}}\right)^{\binom{n}{2} c \ln n} \le n^{-c} \enspace ,$$

where we used $1 - x \le e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4 \log n)$.

Analysis

Repeating the algorithm $c\ln n\binom{n}{2}$ times gives that the probability that we are never successful is

$$\left(1-\frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}c\ln n} \le \left(e^{-1/\binom{n}{2}}\right)^{\binom{n}{2}c\ln n} \le n^{-c}\ ,$$

where we used $1-x \le e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4\log n)$.

Analysis

Repeating the algorithm $c \ln n \binom{n}{2}$ times gives that the probability that we are never successful is

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2} c \ln n} \leq \left(e^{-1/\binom{n}{2}}\right)^{\binom{n}{2} c \ln n} \leq n^{-c} \enspace ,$$

where we used $1 - x \leq e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4 \log n)$.

Analysis

Repeating the algorithm $c\ln n\binom{n}{2}$ times gives that the probability that we are never successful is

$$\left(1-\frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}c\ln n} \le \left(e^{-1/\binom{n}{2}}\right)^{\binom{n}{2}c\ln n} \le n^{-c}\ ,$$

where we used $1-x \le e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4\log n)$.

Analysis

Repeating the algorithm $c \ln n \binom{n}{2}$ times gives that the probability that we are never successful is

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

where we used $1 - x \leq e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4 \log n)$.

Analysis

Repeating the algorithm $c \ln n \binom{n}{2}$ times gives that the probability that we are never successful is

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

where we used $1 - x \leq e^{-x}$.

Theorem 2

The randomized mincut algorithm computes an optimal cut with high probability. The total running time is $\mathcal{O}(n^4 \log n)$.

Improved Algorithm

Algorithm 2 RecursiveMincut($G = (V, E, c)$)

for $i = 1 \to n - n/\sqrt{2}$ **do**
    choose $e \in E$ randomly with probability $c(e)/c(E)$
    $G \leftarrow G/e$
if $|V| = 2$ **return** cut-value;
$cuta \leftarrow$ RecursiveMincut(G);
$cutb \leftarrow$ RecursiveMincut(G);
return $\min\{cuta, cutb\}$

Running time:

- $T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + \mathcal{O}(n^2)$
- This gives $T(n) = \mathcal{O}(n^2 \log n)$.

Improved Algorithm

Algorithm 2 RecursiveMincut$(G = (V, E, c))$

for $i = 1 \to n - n/\sqrt{2}$ **do**
    choose $e \in E$ randomly with probability $c(e)/c(E)$
    $G \leftarrow G/e$
if $|V| = 2$ **return** cut-value;
$cuta \leftarrow$ RecursiveMincut(G);
$cutb \leftarrow$ RecursiveMincut(G);
return $\min\{cuta, cutb\}$

Running time:

- $T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + \mathcal{O}(n^2)$
- This gives $T(n) = \mathcal{O}(n^2 \log n)$.

Improved Algorithm

Algorithm 2 RecursiveMincut($G = (V, E, c)$)

```
for i = 1 → n − n/√2 do
        choose e ∈ E randomly with probability c(e)/c(E)
        G ← G/e
if |V| = 2 return cut-value;
cuta ← RecursiveMincut(G);
cutb ← RecursiveMincut(G);
return min{cuta, cutb}
```

Running time:

- $T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + \mathcal{O}(n^2)$
- This gives $T(n) = \mathcal{O}(n^2 \log n)$.

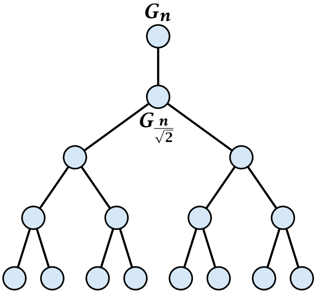
Probability of Success

The probability of contracting an edge from the mincut during one iteration through the for-loop is only

$$\frac{t(t-1)}{n(n-1)} \leq \frac{t^2}{n^2} = \frac{1}{2} \ ,$$

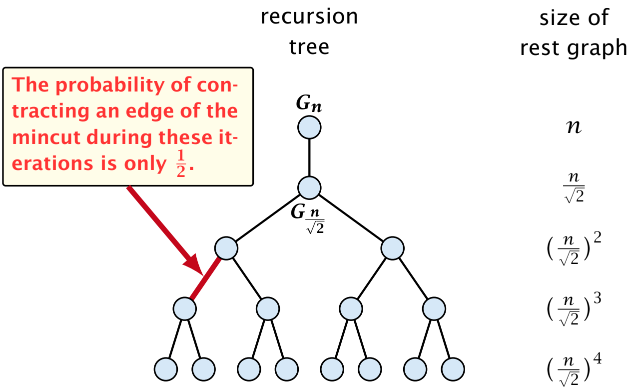
as $t = \frac{n}{\sqrt{2}}$.

Probability of Success

recursion tree	size of rest graph
	n
	$\frac{n}{\sqrt{2}}$
	$\left(\frac{n}{\sqrt{2}}\right)^2$
	$\left(\frac{n}{\sqrt{2}}\right)^3$
	$\left(\frac{n}{\sqrt{2}}\right)^4$

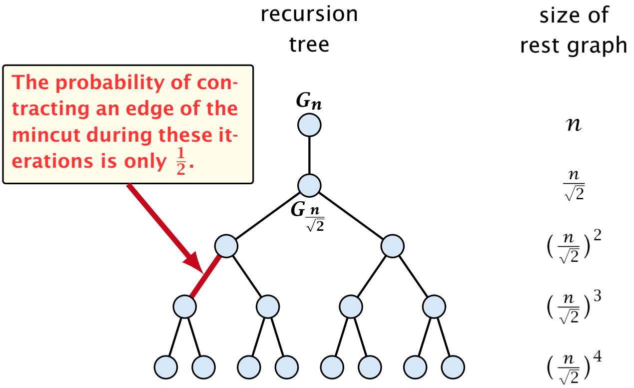
We can estimate the success probability by using the following game on the recursion tree. Delete every edge with probability $\frac{1}{2}$. If in the end you have a path from the root to at least one leaf node you are successful.

Probability of Success



We can estimate the success probability by using the following game on the recursion tree. Delete every edge with probability $\frac{1}{2}$. If in the end you have a path from the root to at least one leaf node you are successful.

Probability of Success



We can estimate the success probability by using the following game on the recursion tree. Delete every edge with probability $\frac{1}{2}$. If in the end you have a path from the root to at least one leaf node you are successful.

Probability of Success

Let for an edge e in the recursion tree, $h(e)$ denote the height (distance to leaf level) of the parent-node of e (end-point that is higher up in the tree). Let h denote the height of the root node.

Call an edge e alive if there exists a path from the parent-node of e to a descendant leaf, after we randomly deleted edges. Note that an edge can only be alive if it hasn't been deleted.

Lemma 3
The probability that an edge e is alive is at least $\frac{1}{h(e)+1}$.

Probability of Success

Let for an edge e in the recursion tree, $h(e)$ denote the height (distance to leaf level) of the parent-node of e (end-point that is higher up in the tree). Let h denote the height of the root node.

Call an edge e alive if there exists a path from the parent-node of e to a descendant leaf, after we randomly deleted edges. Note that an edge can only be alive if it hasn't been deleted.

Lemma 3

The probability that an edge e is alive is at least $\frac{1}{h(e)+1}$.

Probability of Success

Let for an edge e in the recursion tree, $h(e)$ denote the height (distance to leaf level) of the parent-node of e (end-point that is higher up in the tree). Let h denote the height of the root node.

Call an edge e alive if there exists a path from the parent-node of e to a descendant leaf, after we randomly deleted edges. Note that an edge can only be alive if it hasn't been deleted.

Lemma 3

The probability that an edge e is alive is at least $\frac{1}{h(e)+1}$.

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d$$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$

$$= p_{d-1} - \frac{p_{d-1}^2}{2}$$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$

$$= p_{d-1} - \frac{p_{d-1}^2}{2}$$

$x - x^2/2$ is monotonically increasing for $x \in [0, 1]$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$\begin{aligned} p_d &= \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right) \\ &= p_{d-1} - \frac{p_{d-1}^2}{2} \\ &\geq \frac{1}{d} - \frac{1}{2d^2} \end{aligned}$$

$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$

$x - x^2/2$ is monotonically increasing for $x \in [0, 1]$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$

$$= p_{d-1} - \frac{p_{d-1}^2}{2}$$

$x - x^2/2$ is monotonically increasing for $x \in [0,1]$

$$\geq \frac{1}{d} - \frac{1}{2d^2} \geq \frac{1}{d} - \frac{1}{d(d+1)}$$

Probability of Success

Proof.

- An edge e with $h(e) = 1$ is alive if and only if it is not deleted. Hence, it is alive with proability at least $\frac{1}{2}$.
- Let p_d be the probability that an edge e with $h(e) = d$ is alive. For $d > 1$ this happens for edge $e = \{c, p\}$ if it is not deleted **and** if one of the child-edges connecting to c is alive.
- This happens with probability

$$p_d = \frac{1}{2}\left(2p_{d-1} - p_{d-1}^2\right)$$

$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$

$$= p_{d-1} - \frac{p_{d-1}^2}{2}$$

$x - x^2/2$ is monotonically increasing for $x \in [0, 1]$

$$\geq \frac{1}{d} - \frac{1}{2d^2} \geq \frac{1}{d} - \frac{1}{d(d+1)} = \frac{1}{d+1} \;.$$

15 Global Mincut

Lemma 4
One run of the algorithm can be performed in time $\mathcal{O}(n^2 \log n)$ *and has a success probability of* $\Omega(\frac{1}{\log n})$.

Doing $\Theta(\log^2 n)$ *runs gives that the algorithm succeeds with high probability. The total running time is* $\mathcal{O}(n^2 \log^3 n)$.

15 Global Mincut

Lemma 4
One run of the algorithm can be performed in time $\mathcal{O}(n^2 \log n)$ *and has a success probability of* $\Omega(\frac{1}{\log n})$.

Doing $\Theta(\log^2 n)$ *runs gives that the algorithm succeeds with high probability. The total running time is* $\mathcal{O}(n^2 \log^3 n)$.