Chapter 7
Randomization

Algorithm Theory
WS 2015/16

Fabian Kuhn

Types of Randomized Algorithms

Las Vegas Algorithm:

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

Zulunn

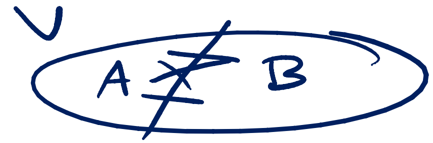
- **Example:** randomized quicksort, contention resolution

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 $V = A \cup B$, $A \cap B = \emptyset$, $A, B \neq \emptyset$

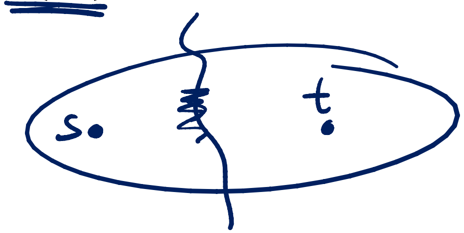
Size of the cut (A, B): # of edges crossing the cut

- For weighted graphs, total edge weight crossing the cut

edge connectivity of G

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

Maximum-flow based algorithm:



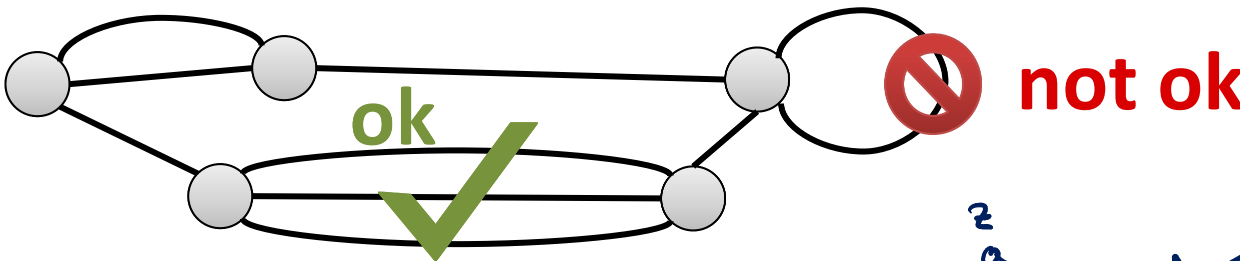
- Fix s, compute min s-t-cut for all $t \neq s$
- $O(m \cdot \lambda(G)) = O(mn)$ per s-t cut
- Gives an $O(mn\lambda(G)) = O(mn^2)$-algorithm

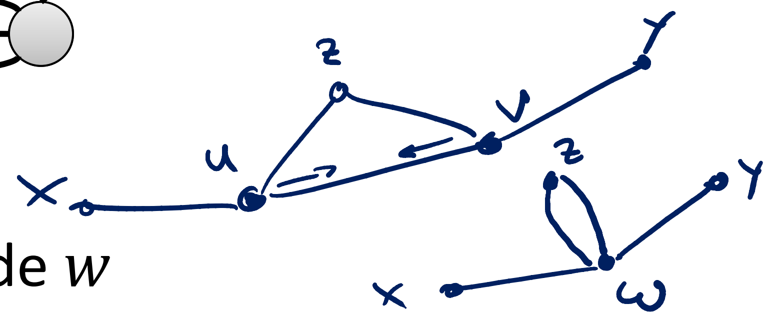
Best-known deterministic algorithm: $O(mn + n^2 \log n)$

G dense: $O(n^3)$

Edge Contractions

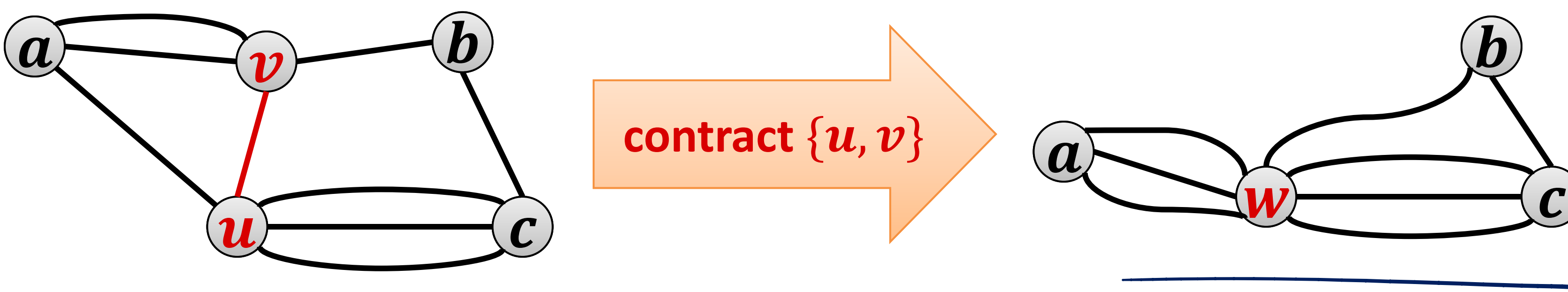
- In the following, we consider multi-graphs that can have multiple edges (but no self-loops)





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

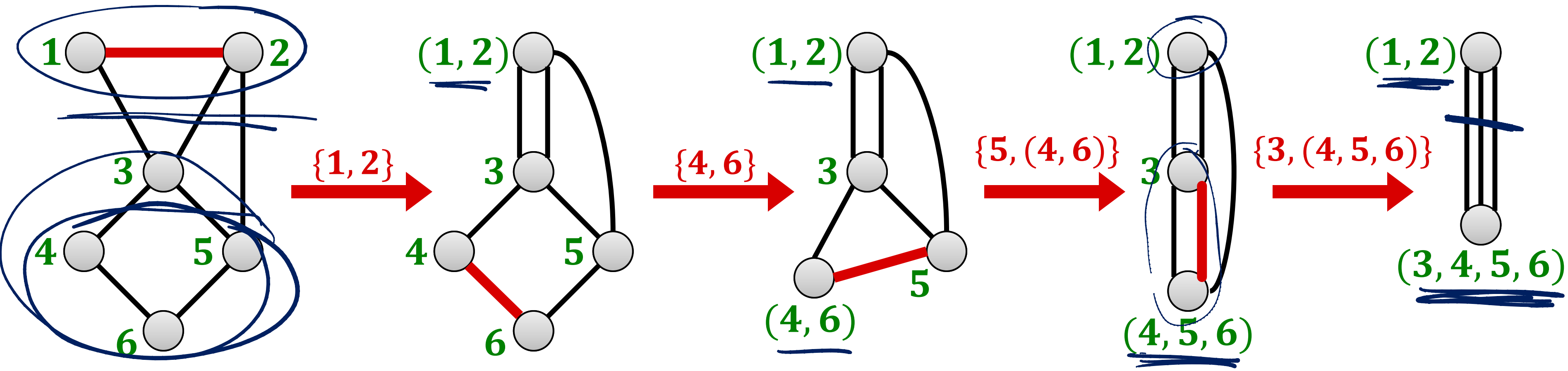
- Replace nodes u, v by new node w
- For all edges $\{u, x\}$ and $\{v, x\}$, add an edge $\{w, x\}$
- Remove self-loops created at node w



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 later.

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.

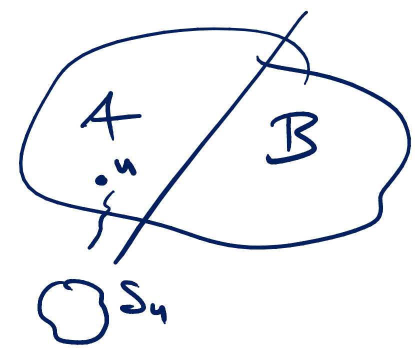
Contractions and Cuts

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

Proof:

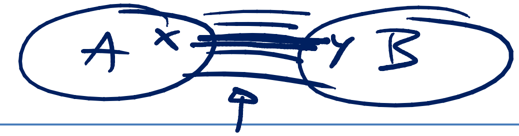
- All cuts in a (partially) contracted graph correspond to cuts of the same size in the original graph G as follows:
  - For a node u of the contracted graph, let S_u be the set of original nodes that have been merged into u (the nodes that u represents)
  - Consider a cut (A, B) of the contracted graph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

specific

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

$\sum_v \deg(v) = 2 \cdot m$

To prove the theorem, we need the following claim:

edge connectivity

$\geq$

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

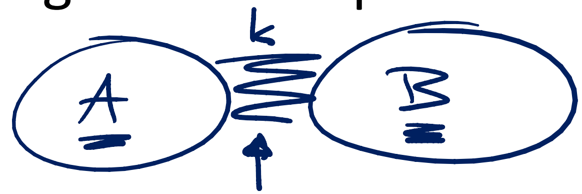
Proof:

- 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

specific

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.

$1, 2, \dots, i, \dots, n-2$

- 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

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

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

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

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

Getting The Min Cut

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

Proof:

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

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

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

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

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.

$1 - \frac{1}{n^c}$

Proof:

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

$1 + x \leq e^{+x} \quad (\forall x \in \mathbb{R})$

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

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

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

- It remains to show that each instance can be implemented in $O(n^2)$ time.

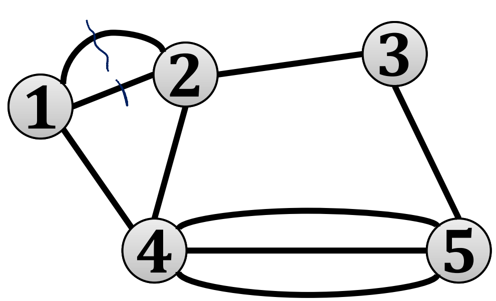
Implementing Edge Contractions

Edge Contraction:

- Given: multigraph with n nodes
  - assume that set of nodes is $\{1, \dots, n\}$
- Goal: contract edge $\{u, v\}$

Data Structure

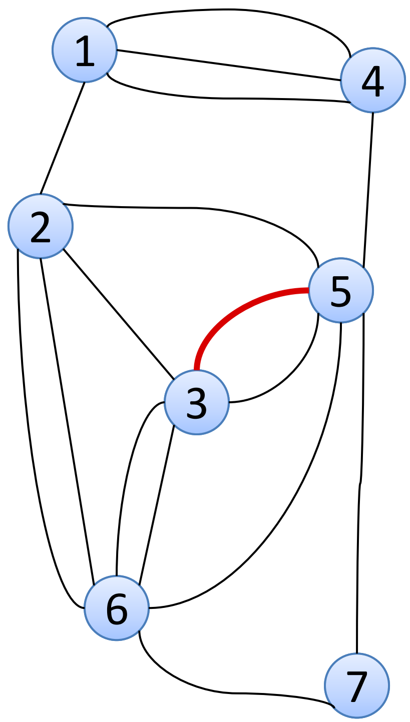
- We can use either adjacency lists or an adjacency matrix
- Entry in row i and column j: #edges between nodes i and j
- Example:



$$A = \begin{array}{cc} & \begin{array}{ccccc} & 2 & & & \end{array} \\ \begin{array}{c} 1 \\ \\ \\ \\ \end{array} & \begin{pmatrix} 0 & 2 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 3 & 0 \end{pmatrix} \end{array}$$

Contracting An Edge

Example: Contract one of the edges between 3 and 5

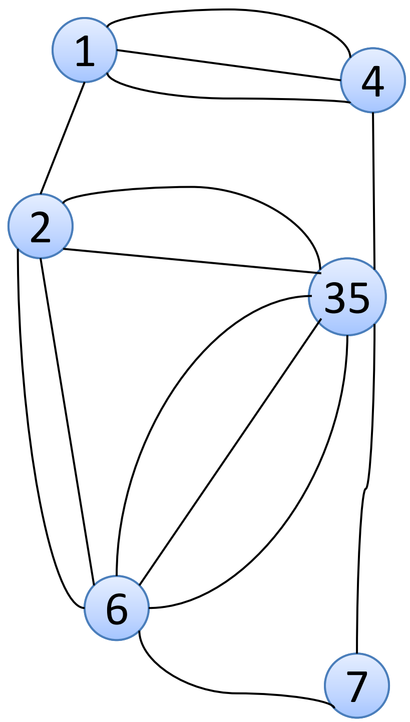


	1	2	3	4	5	6	7
1	0	1	0	3	0	0	0
2	1	0	1	0	1	2	0
3	0	1	0	0	2	2	0
4	3	0	0	0	1	0	0
5	0	1	2	1	0	1	1
6	0	2	2	0	1	0	1
7	0	0	0	0	1	1	0

{3,5}	0	2	~~0~~	1		3	1

Contracting An Edge

Example: Contract one of the edges between 3 and 5

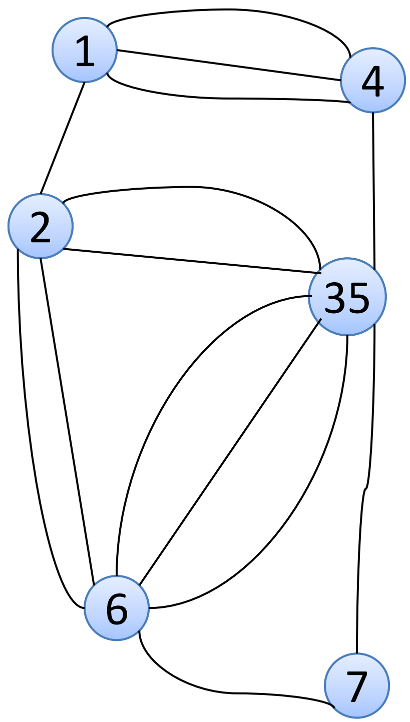


	1	2	3	4	5	6	7
1	0	1	0	3	0	0	0
2	1	0	1	0	1	2	0
3	0	1	0	0	2	2	0
4	3	0	0	0	1	0	0
5	0	1	2	1	0	1	1
6	0	2	2	0	1	0	1
7	0	0	0	0	1	1	0

{3,5}	0	2		1		3	1

Contracting An Edge

Example: Contract one of the edges between 3 and 5



	1	2	35	4		6	7
1	0	1	0	3		0	0
2	1	0	2	0		2	0
35	0	2	0	1		3	1
4	3	0	1	0		0	0
						0	
6	0	2	3	0		0	1
7	0	0	1	0		1	0

{3,5}	0	2		1		3	1

Contracting an Edge

Claim: Given the adjacency matrix of an n-node multigraph and an edge $\{u, v\}$, one can contract the edge $\{u, v\}$ in time $O(n)$.

- Row/column of combined node $\{u, v\}$ is sum of rows/columns of u and v
- Row/column of u can be replaced by new row/column of combined node $\{u, v\}$
- Swap row/column of v with last row/column in order to have the new $(n-1)$-node multigraph as a contiguous $(n-1) \times (n-1)$ submatrix

Finding a Random Edge

- We need to contract a uniformly random edge
- How to find a uniformly random edge in a multigraph?
  - Finding a random non-zero entry (with the right probability) in an adjacency matrix costs $O(n^2)$.

Idea for more efficient algorithm:

- First choose a random node u
  - with probability proportional to the degree (#edges) of u
- Pick a random edge of u
  - only need to look at one row → time $O(n)$

| 1 | 2 | 0 | 3 | 0 | 0 | 2 | - - - |

total # edges : d

$$\frac{1}{d}, \quad \left(1-\frac{1}{d}\right)\cdot\frac{2}{d-1}=\frac{2}{d}, \quad \left(1-\frac{3}{d}\right)\frac{3}{d-3}=\frac{3}{d}$$

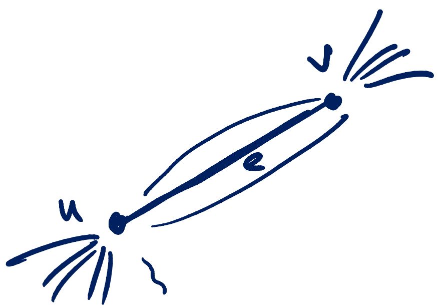
Choose a Random Node

Edge Sampling:

1. Choose a node $u \in V$ with probability

$$\frac{\deg(u)}{\sum_{v \in V} \deg(v)} = \frac{\deg(u)}{2m}$$

2. Choose a uniformly random edge of u $\leftarrow$ time $O(n)$



$$\mathbb{P}(\text{pick } e) = \frac{\deg(u)}{2m} \cdot \frac{1}{\deg(u)} + \frac{\deg(v)}{2m} \cdot \frac{1}{\deg(v)} = \frac{1}{2m} + \frac{1}{2m} = \frac{1}{m}$$

Choose a Random Node

- We need to choose a random node u with probability $\frac{\deg(u)}{2m}$
- Keep track of the number of edges m and maintain an array with the degrees of all the nodes
  - Can be done with essentially no extra cost when doing edge contractions

Choose a random node:

degsum = 0;
for all nodes $u \in V$:
    with probability $\frac{\deg(u)}{2m-\text{degsum}}$:
        pick node u; terminate
    else
        degsum += $\deg(u)$

time : $O(n)$

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.

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

- One instance consists of $n - 2$ edge contractions
- Each edge contraction can be carried out in time $O(n)$
  - Actually: $O(\text{current \#nodes})$
- Time per instance of the contraction algorithm: $O(n^2)$

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

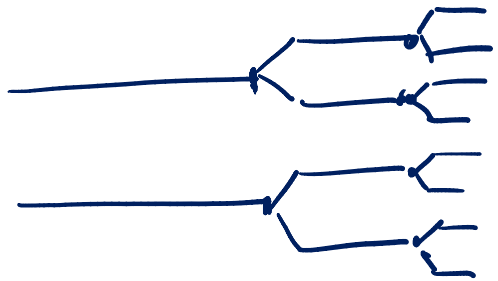
- For a specific min cut (A, B), if (A, B) survives the first i contractions,

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

- **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

$n - \frac{n}{\sqrt{2}}$

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:

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

$$= \frac{t}{n} \cdot \frac{t-1}{n-1}$$

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

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

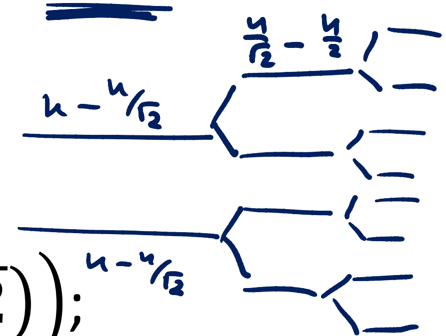
Better Randomized Algorithm

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 $\geq {}^{1}/_{2}$.

$\mathbf{contract}(G, t)$:

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



$\mathbf{mincut}(G)$:

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

Success Probability

mincut(G):

1. $X_1 := \text{mincut}\left(\text{contract}(G, n/\sqrt{2})\right);$
2. $X_2 := \text{mincut}\left(\text{contract}(G, n/\sqrt{2})\right);$
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 $\geq$ $\frac{1}{2} \cdot P(n/\sqrt{2})$

Recursion:

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

Success Probability

$P(n) \geq \frac{1}{\log_2 n}$

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

Proof (by induction on n):

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

Base case: $n = 2$ ✓

$x - \frac{x^2}{4}$

Ind. Step: $P(n) = P(n/\sqrt{2}) - \frac{1}{4} P(n/\sqrt{2})^2$

$$\overset{\text{I.H.}}{\geq} \frac{1}{\log(n/\sqrt{2})} - \frac{1}{4} \frac{1}{\log(n/\sqrt{2})^2} = \frac{1}{\log(n/\sqrt{2})}\left(1 - \frac{1}{4} \cdot \frac{1}{\log(n/\sqrt{2})}\right)$$

$$= \frac{1}{\log_2 n - 1/2}\left(1 - \frac{1}{4\log_2 n - 2}\right) = \frac{1}{\log_2 n - \frac{1}{2}}\left(\frac{4\log_2 n - 3}{4\log_2 n - 2}\right)$$

$$= \frac{4\log_2 n - 3}{4\log_2^2 n - \cancel{2}4\log_2 n + 1} \geq \frac{1}{\log n}$$

Running Time

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

2. $X_2 := \text{mincut}\left(\text{contract}(G, n/\sqrt{2})\right);$

3. **return** $\min\{X_1, X_2\}$;

Recursion:

- $T(n)$: time to apply algorithm to n-node graphs
- Recursive calls: $2T\left(n/\sqrt{2}\right)$
- Number of contractions to get to $n/\sqrt{2}$ nodes: $O(n)$

$$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + O(n^2), \qquad T(2) = O(1)$$