

# **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
- **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  $\lambda(G)$ )

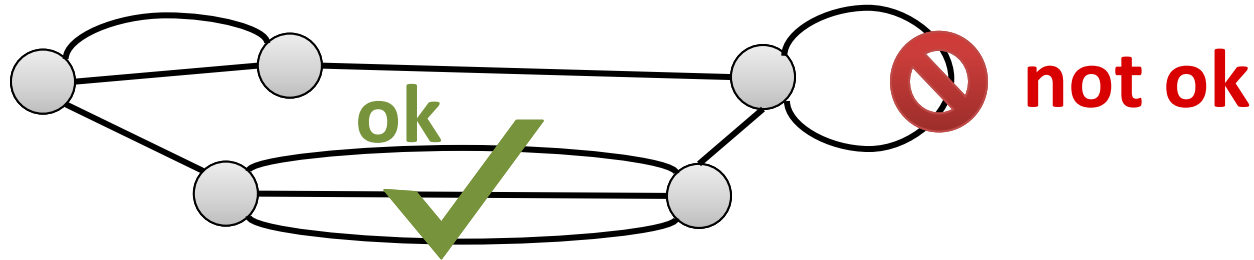
**Maximum-flow based algorithm:**

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

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

# Edge Contractions

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



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

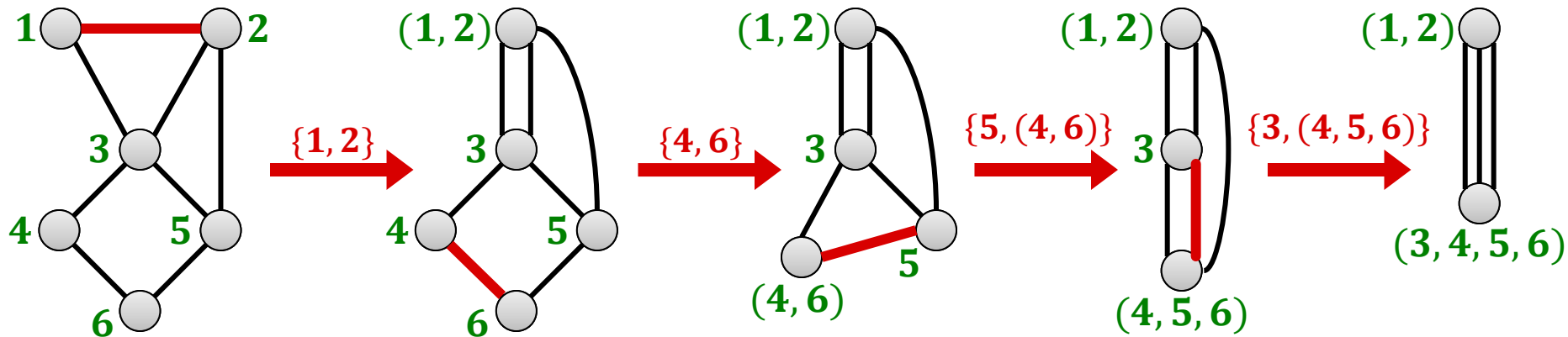
- Replace nodes  $u, v$  by new node  $w$
- For all edges  $\{u, x\}$  and  $\{v, x\}$ , add an edge  $\{w, x\}$
- Remove self-loops created at node  $w$



# 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 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

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

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  $\rightarrow$  **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  $k \implies$  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  $m = \frac{1}{2} \cdot \sum_v \deg(v)$

# 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  $(A, B)$

# Getting The Min Cut

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

**Proof:**

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

# 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:**

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

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

**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.

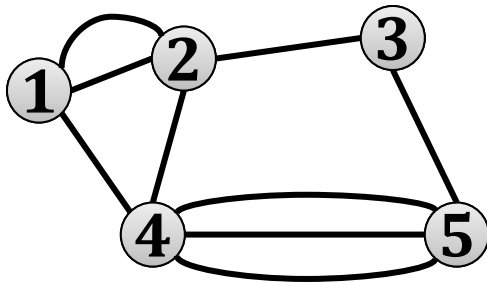
# Implementing Edge Contractions

## Edge Contraction:

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

## Data Structure

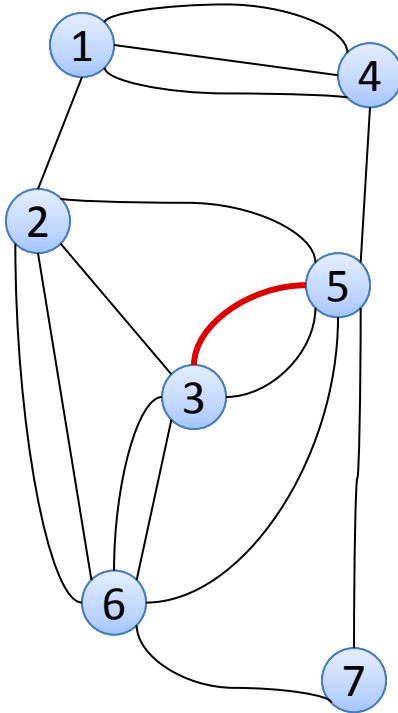
- We can use either adjacency lists or an adjacency matrix
- Entry in row  $i$  and column  $j$ : #edges between nodes  $i$  and  $j$
- Example:



$$A = \begin{pmatrix} 0 & 2 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 3 & 0 \end{pmatrix}$$

# 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

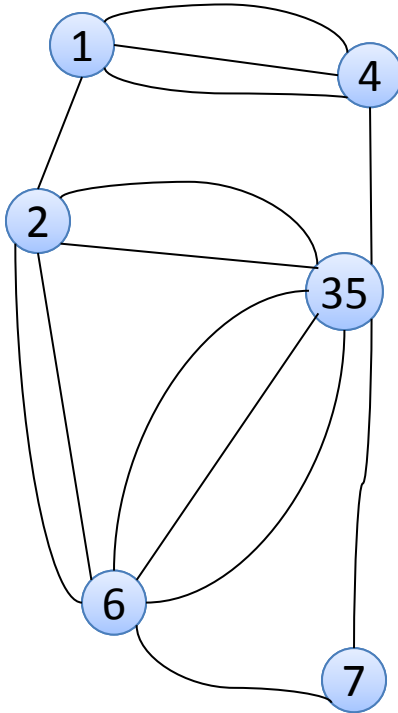
$\{3,5\}$

--	--	--	--	--	--	--



# Contracting An Edge

**Example:** Contract one of the edges between 3 and 5

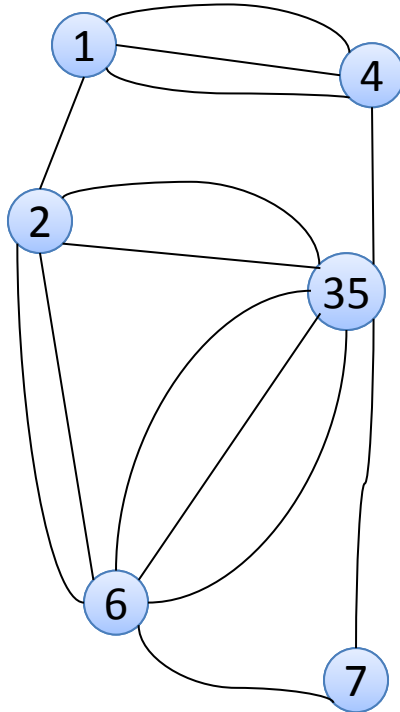


	1	2	3	4	5	6	7
1	0	1	0	3	0	0	0
2	1	0	1	0	1	2	0
3	0	1	0	0	2	2	0
4	3	0	0	0	1	0	0
5	0	1	2	1	0	1	1
6	0	2	2	0	1	0	1
7	0	0	0	0	1	1	0

$\{3,5\}$	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
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  $\rightarrow$  time  $O(n)$

# 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  $u$

# 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( $u$ )
```

# 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.
1. 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:

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

**mincut( $G$ ):**

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

**$P(n)$ :** 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  $\geq$

**Recursion:**

# 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  $n$ ):

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

# Running Time

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

## Recursion:

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

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