Chapter 7
Randomization

Algorithm Theory
WS 2016/17

Fabian Kuhn

Randomized Quicksort

Quicksort:

S | v

$S_\ell < v$ | v | $S_r > v$

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_ℓ with elements $< v$,

    and S_r with elements $> v$

    return $\boxed{\text{Quick}(S_\ell) \mid v \mid \text{Quick}(S_r)}$

end;

Randomized Quicksort Analysis

Randomized Quicksort: pick uniform random element as pivot

Running Time of sorting n elements:

- Let's just count the number of comparisons
- In the partitioning step, all $n-1$ non-pivot elements have to be compared to the pivot
- **Number of comparisons:**

$$\boldsymbol{n-1\ +\ \#\mathbf{comparisons\ in\ recursive\ calls}}$$

- **If rank of pivot is r:**
  recursive calls with $\boldsymbol{r-1}$ and $\boldsymbol{n-r}$ elements

Law of Total Expectation

- Given a random variable X and
- a set of events $A_1, \dots, A_k$ that partition Ω
  - E.g., for a second random variable Y, we could have
    $$A_i \coloneqq \{\omega \in \Omega : Y(\omega) = i\}$$

Law of Total Expectation

$$\mathbb{E}[X] = \sum_{i=1}^{k} \mathbb{P}(A_i) \cdot \mathbb{E}[X \mid A_i] = \sum_{y} \mathbb{P}(Y = y) \cdot \mathbb{E}[X \mid Y = y]$$

Example:

- X: outcome of rolling a die
- $A_0 = \{X \text{ is even}\}, \;\; A_1 = \{X \text{ is odd}\}$

Randomized Quicksort Analysis

Random variables:

- C: total number of comparisons (for a given array of length n)
- R: rank of first pivot
- C_ℓ, C_r: number of comparisons for the 2 recursive calls

$$\mathbb{E}[C] = n - 1 + \mathbb{E}[C_\ell] + \mathbb{E}[C_r]$$

Law of Total Expectation:

$$\begin{aligned}\mathbb{E}[C] &= \sum_{r=1}^{n} \mathbb{P}(R = r) \cdot \mathbb{E}[C|R = r] \\ &= \sum_{r=1}^{n} \mathbb{P}(R = r) \cdot (n - 1 + \mathbb{E}[C_\ell|R = r] + \mathbb{E}[C_r|R = r])\end{aligned}$$

Randomized Quicksort Analysis

We have seen that:

$$\mathbb{E}[C] = \sum_{r=1}^{n} \mathbb{P}(R = r) \cdot (n - 1 + \mathbb{E}[C_\ell | R = r] + \mathbb{E}[C_r | R = r])$$

Define:

- $\boldsymbol{T(n)}$: expected number of comparisons when sorting n elements

$$\mathbb{E}[C] = T(n)$$
$$\mathbb{E}[C_\ell | R = r] = T(r-1)$$
$$\mathbb{E}[C_r | R = r] = T(n-r)$$

Recursion:

$$\boldsymbol{T(n) = \sum_{r=1}^{n} \frac{1}{n} \cdot \big(n - 1 + T(r-1) + T(n-r)\big)}$$

$$\boldsymbol{T(0) = T(1) = 0}$$

Randomized Quicksort Analysis

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

Proof:

$$T(n) = \sum_{r=1}^{n} \frac{1}{n} \cdot \big(n - 1 + T(r-1) + T(n-r)\big), \qquad T(0) = 0$$

Randomized Quicksort Analysis

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

Proof:

$$T(n) \leq n - 1 + \frac{4}{n} \cdot \int_1^n x \ln x \, dx$$

$$\int x \ln x \, dx = \frac{x^2 \ln x}{2} - \frac{x^2}{4}$$

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

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

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

Number of comparisons: $\boldsymbol{X}$

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

- 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 $a_1, \dots, a_n \in \mathbb{R}$,

$$\mathbb{E}\left[\sum_{i}^{n} a_i X_i\right] = \sum_{i}^{n} a_i \mathbb{E}[X_i]\,.$$

Randomized Quicksort Analysis

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

Proof:

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

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 $(\boldsymbol{A}, \boldsymbol{B})$: # 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 $\lambda(G)$)

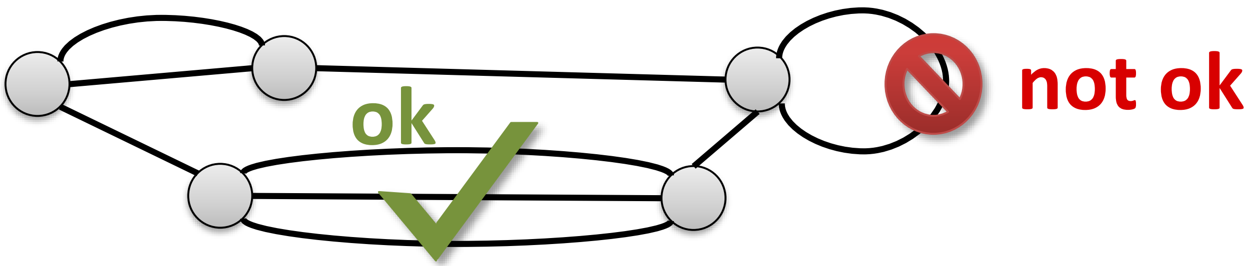
Maximum-flow based algorithm:

- 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)



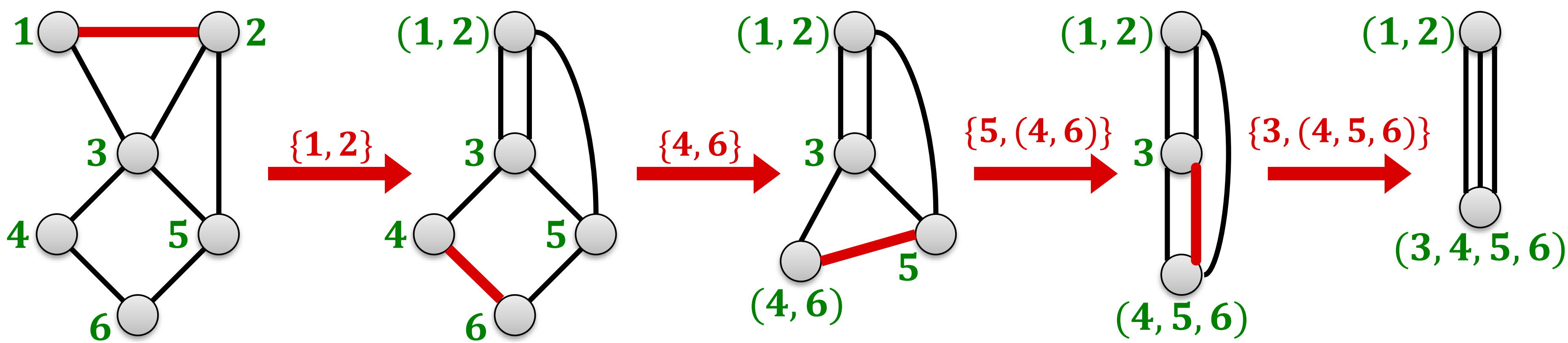
Contracting edge $\{u, v\}$:

- 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 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

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 $m = {}^1/_2 \cdot \sum_v \deg(v)$

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:

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

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.