ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Mohsen Ghaffari, Rasmus Kyng, David Steurer

Algorithms, Probability, and Computing **Special Assignment 1** **HS20**

- Write an exposition of your solution using a computer, where we strongly recommend to use LaTeX. **We do not grade hand-written solutions.**
- You need to submit your solution via Moodle until **Tuesday, October 27, 2020** by **10 pm**. Late solutions will not be graded.
- For geometric drawings that can easily be integrated into LaTeX documents, we recommend the drawing editor IPE, retrievable at `http://ipe.otfried.org` in source code and as an executable for Windows.
- Write short, simple, and precise sentences.
- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer" or "justify intuitively", then a formal proof is **always** required. You can of course refer in your solutions to the lecture notes and to the exercises, if a result you need has already been proved there.
- We would like to stress that the ETH Disciplinary Code applies to this special assignment as it constitutes part of your final grade. The only exception we make to the Code is that we encourage you to verbally discuss the tasks with your colleagues. However, you need to write down the names of all your collaborators at the beginning of the writeup. It is strictly prohibited to share any (hand)written or electronic (partial) solutions with any of your colleagues. We are obligated to inform the Rector of any violations of the Code.
- There will be two special assignments this semester. Both of them will be graded and the average grade will contribute 20% to your final grade.
- As with all exercises, the material of the special assignments is relevant for the (midterm and final) exams.

The following two concentration results might be useful to solve the exercises.

Theorem 1 (Chernoff Bound). *Let* $X_1, X_2, \ldots, X_n$ *be independent and Bernoulli-distributed random variables. Let* $X := \sum_{i=1}^{n} X_i$ *and* $\delta \in [0,1]$ *be arbitrary. Then,*

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]}$$

and

$$\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{2}\delta^2 \mathbb{E}[X]}.$$

Theorem 2 (Chernoff Bound variant). *Let* $X_1, X_2, \ldots, X_n$ *be Bernoulli-distributed random variables and* $X := \sum_{i=1}^{n} X_i$ *. Let* $p \in [0,1]$ *and* $\delta \in [0,1]$ *be arbitrary. If for each* $i \in [n]$ *and* $x_1, x_2 \ldots, x_{i-1} \in \{0,1\}$*, it holds that* $E[X_i | X_0 = x_0, X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq p$*, then*

$$\Pr[X \geq (1+\delta)np] \leq e^{-\frac{1}{3}\delta^2 np}.$$

Similarly, if for each $i \in [n]$ *and* $x_1, x_2 \ldots, x_{i-1} \in \{0,1\}$*, it holds that* $E[X_i | X_0 = x_0, X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \geq p$*, then*

$$\Pr[X \leq (1-\delta)np] \leq e^{-\frac{1}{2}\delta^2 np}.$$

Exercise 1 — 25 points

(*Black-Box Minimum Spanning Tree*)

In this exercise, we ask you to design a randomized algorithm that computes a Minimum Spanning Tree of some weighted input graph G. We assume that G is connected and that the edge weights are pairwise distinct. Note that you have seen a randomized algorithm that solves this problem in $O(m)$ time during the first lecture of the course. This time, your algorithm cannot directly access the input graph. Instead, it is only allowed to invoke each of the following black-box routines a *constant* number of times.

1. Algorithm $\mathcal{A}_{\mathrm{MSF}}$ takes as input some weighted graph H with n nodes and $O(n^{3/2})$ edges. It returns a Minimum Spanning Forest of the graph H.

2. Algorithm $\mathcal{A}_{\mathrm{sample}}$ takes as input some weighted graph H and some $p \in [0,1]$. It returns a weighted graph obtained from H by including each edge of H independently with probability p.

3. Algorithm $\mathcal{A}_{T-\mathrm{heavy}}$ takes as input some weighted graph H and some weighted Forest T, with both H and T having the same set of vertices. It returns a weighted graph obtained from H by removing all T-heavy edges.

Your algorithm needs to output a Minimum Spanning Tree with probability at least $1-O(1/n)$. You only need to analyze the success probability, but not the runtime of the algorithm. You can obtain partial points by proving that your algorithm succeeds with positive constant probability.
Hint: First, compute the Minimum Spanning Forest of a graph obtained from subsampling the edges in the input graph. Use this Minimum Spanning Forest to remove edges from the input graph that are not contained in the Minimum Spanning Tree of the input graph. Give a bound on the number of remaining edges by modifying the analysis seen on page 6 of Chapter 1. In particular, the analysis on page 6 shows that the expected number of not T-heavy edges in the graph G is $O(n)$, where T is the MSF in the graph obtained from G by independently including each edge with probability $1/2$. How does the bound on the number of not T-heavy edges change if we sample with probability p? Finally, how can you convert a result that holds in expectation into a bound that holds with probability $1-O(1/n)$? This step should remind you about one of the past exercises.

Exercise 2 30 points

(*Minimum Cuts in Hypergraphs of Rank* 3)

A hypergraph is a generalization of a graph in which an edge can join an arbitrary number of nodes. Formally, a hypergraph H is a tuple $H=(V,E)$ with $E \subseteq 2^V \setminus \{\emptyset\}$. In a multihypergraph, the same edge can be present multiple times. The rank of a (multi)hypergraph is defined as $\max_{e\in E}|e|$. For a non-trivial partition of the vertices $(S, V\setminus S)$, where $S \neq \emptyset$ and $S \neq V$, the corresponding cut in the (multi)hypergraph is defined as the (multi)set of all the hyperedges that have at least one endpoint in each side of the cut. The notion of contracting a (hyper)edge directly generalizes from the usual graph setting, see Figure 1 for an example.

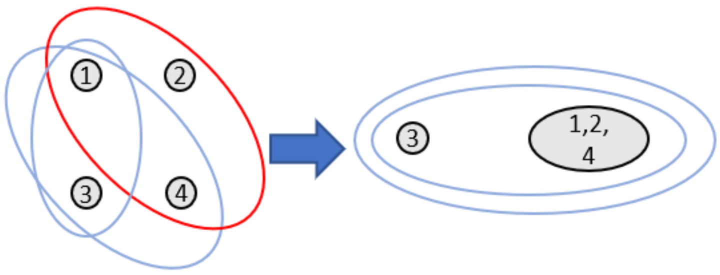

Figure 1: On the left, you can see the (multi)hypergraph $H=(V,E)$ with $V=\{1,2,3,4\}$ and $E=\{\{1,2,4\},\{1,3,4\},\{1,3\}\}$. After contracting the hyperedge $e=\{1,2,4\}$, one obtains the multihypergraph $H/e=(V',E')$ with $V'=\{3,(124)\}$ and $E'=\{\{3,(124)\},\{3,(124)\}\}$. Note that E' is a multiset.

To solve the problems, you can use the following fact.

1. Let $H=(V,E)$ denote a (multi)hypergraph of rank at most 3. Let $E_{\geq 2}$ denote the (multi)set consisting of those edges in E that join at least 2 vertices. Assume that

$E_{\geq 2} \neq \emptyset$. In $O(n^2)$ time, one can select an edge from $E_{\geq 2}$ uniformly at random and contract it.

The input for all the problems is a hypergraph of rank at most 3. Note that a hypergraph of rank at most 3 contains $O(n^3)$ edges. We mean with a randomized algorithm an algorithm that outputs the correct solution with probability at least 1/2. For exercises c) and e), we consider algorithms for the problem of computing the minimum cut size. Moreover, they should *never* output a value that is strictly smaller than the minimum cut size.

(a) Devise an algorithm with a running time of $O(n^3)$ that outputs a random cut of the input hypergraph such that the following holds: for each minimum cut C of the input hypergraph, the algorithm outputs C with probability $\Omega(1/n^3)$.

(b) Use exercise a) to deduce that the number of distinct minimum cuts in a hypergraph of rank at most 3 is bounded by $O(n^3)$.

(c) Devise a randomized algorithm with a running time of $O(n^6)$.

(d) Devise a randomized algorithm with a running time of $O(n^6 \log n)$ that outputs *all* of the minimum cuts.
Hint: Let C denote some fixed minimum cut. What is the probability to output C by running the algorithm from exercise a) N times and outputting all of the N cuts?

(e) Let $\alpha > 3$ be arbitrary. Assume that there exists a randomized algorithm with a running time of $O(n^\alpha)$. Show that this implies the existence of a randomized algorithm with a running time of $O(n^{3+\frac{3(\alpha-3)}{\alpha}})$.

(f) Prove that the sequence $(a_i)_{i\geq 0}$ with $a_0 = 6$ and $a_i = 3 + \frac{3(a_{i-1}-3)}{a_{i-1}}$ for $i \geq 1$ converges and compute its limit.
Hint: Each monotonically decreasing sequence that is bounded from below converges.

(g) What is the best possible upper bound on the randomized complexity of finding the minimum cut size of a given hypergaph of rank at most 3 that one can infer from the previous exercises?

Exercise 3 25 points

(*Concentration for Quicksort*)

In the lecture, we have seen that randomized quicksort performs $O(n \log n)$ comparisons *in expectation* (Theorem 2.7), where n denotes the number of distinct elements to sort. By using Markov's inequality, one can therefore deduce that quicksort performs at most $O(n \log n)$ comparisons with positive constant probability. In this exercise, we ask you to prove that quicksort performs at most $O(n \log n)$ comparisons with probability $1 - O(1/n)$. To do so, solve the following subproblems.

(a) Assume that the maximum recursion depth of one particular execution is $O(\log n)$. Argue why this implies that the total number of comparisons performed during that particular execution is $O(n \log n)$.

(b) Let x denote an arbitrary element. Prove that x participates in at most $O(\log n)$ recursive calls (The depth of x in the corresponding binary tree is $O(\log n)$) with probability $1-O(1/n^2)$. To do so, define for each recursion depth i a random variable X_i. The random variable X_i is set to 1 if x participates in at most i recursive calls, or the subproblem of the $(i+1)$-th level that x is part of contains at most a $(2/3)$-fraction of the elements from the subproblem of the i-th level that x is part of. Otherwise, X_i will be set to 0. Let $X := \sum_{i=0}^{100 \log_{3/2} n} X_i$. First, show that $\Pr[X < \log_{3/2}(n)] = O(1/n^2)$. Then, show that $X \geq \log_{3/2}(n)$ implies that x participates in at most $O(\log n)$ recursive calls.

(c) Prove that the maximum recursion depth is $O(\log n)$ with probability $1 - O(1/n)$.

(d) Conclude with the help of the previous exercises that quicksort performs $O(n \log n)$ comparisons with probability $1 - O(1/n)$.

Exercise 4 — 20 points

(*Area of a Polygon below a Query Line*)

In this exercise, you are given as input a fixed convex polygon C, where $p_0, p_1, \ldots, p_{n-1}$ are the vertices of C in counterclockwise order. After preprocessing, you need to answer each query in $O(\log n)$ time. The query consists of one non-vertical line and the output is the area of the convex polygon under the query line. See Figure 2 for an example. You can use unlimited space and preprocessing time in order to get the full number of points. For those of you who like a challenge, devise an algorithm that works with $O(n)$ space.

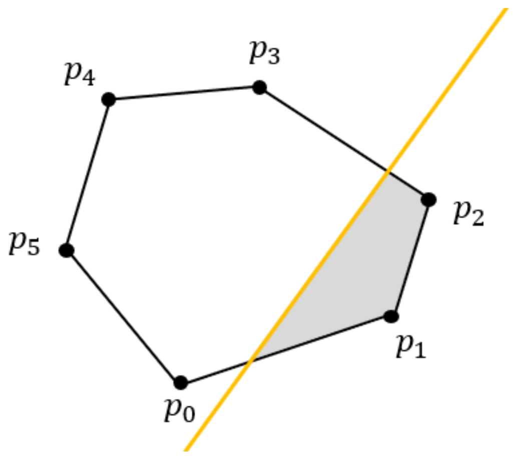

Figure 2: A query consists of one non-vertical line, in this example the yellow line. The output in this example should be the area of the grey region.