ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Rasmus Kyng, Angelika Steger, David Steurer, Emo Welzl

Algorithms, Probability, and Computing **Special Assignment 1** **HS22**

- Write your solutions using a computer, where we strongly recommend to use LaTeX. **We do not grade hand-written solutions.**
- The solution is due on **Tuesday, November 1st, 2022** by **2:15 pm**. Please submit one file per exercise on Moodle.
- For geometric drawings that can easily be integrated into LaTeX documents, we recommend the drawing editor IPE, retrievable at `http://ipe.otfried.org` or through various package managers.
- 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 include a list of all of your collaborators in each of your submissions. 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.

Exercise 1

20 **points**

(*Alternatively Sampled Minimum Spanning Trees*)

We consider a slightly different way of sampling edges in the randomized minimum spanning tree algorithm:

> MODIFIED RANDOMIZED MST(G):
> Perform three iterations of Borůvka's algorithm
> In the new graph (V, E):
> Select $E' \subset E$ with $|E'| = \lceil p \cdot |E| \rceil$ uniformly at random among all such sets
> Recursively compute an MSF T of (V, E')
> Use FINDHEAVY to find all edges in $E - E'$ which are not T-heavy
> Add all *not* T-heavy edges to T and delete all other edges, yielding E''
> Recurse on (V, E'') (until the graph contains only one vertex)

(a) Prove the following statement:
Let $G = (V, E)$ be a weighted graph with an injective weight function, with $|V| = n$ and $|E| = m$. Let E' be a random subset of E with exactly r edges. Let T be a minimum spanning forest of (V, E'). Let L be the set of edges in E which are *not* T-heavy. Then, $E[|L|] \leq nm/r$.

(b) For which values of p can you conclude that MODIFIED RANDOMIZED MST runs in expected time $O(n + m)$? As usual, you may assume that all weights are distinct. You can refer to the analysis of the original algorithm in the lecture notes. You do not need to prove correctness of the algorithm.

Exercise 2

25 **points**

(*Minimum* k*-Cuts*)

Given an undirected (multi-)graph $G = (V, E)$, a k-cut is a subset $C \subseteq E$ such that the graph $G' = (V, E \setminus C)$ consists of at least k connected components. Note that a 2-cut is simply a cut as we defined it in the lecture. The size of a k-cut C is the number of edges it contains, i.e., $|C|$. We consider the problem of finding a minimum k-cut using a randomized algorithm. We write $\mu(G, k)$ for the size of a minimum k-cut in G.

(a) Given a multigraph $G = (V, E)$ on n vertices and an edge $e \in E$ picked uniformly at random, show that $P[\mu(G, k) = \mu(G/e, k)] \geq 1 - \frac{2(k-1)}{n}$.
You can obtain partial points if your solution is correct assuming n *being a multiple of* $(k-1)$.

Now consider the following adaption of the BasicMinCut algorithm in the lecture notes:

```
BasicMin-k-Cut(G, k):
while G has more than f(k) vertices do
    pick a random edge e in G
    G ← G/e
end while
compute the size of all k-cuts in G
return the size of the smallest k-cut found
```

(b) Design an algorithm based on BasicMin-k-Cut for a suitable $f(k)$ which computes the correct size of the minimum k-cut of any graph G with probability at least $1/2$. Determine the runtime of your algorithm in terms of n and k. For any constant $k \geq 2$, your algorithm should run in polynomial time (in n).
Note: Such an algorithm running in $O(n^{g(k)} \cdot h(k))$ is called "slicewise polynomial".

(c) For the case $k = 3$, use the bootstrapping approach to show that there exists a series of algorithms $(A_i)_{i \geq 0}$ such that A_i finds a minimum 3-cut in time $O(n^{r(i)})$ with probability at least $1/2$, where $r(i+1) = 6 - 8/r(i)$ and $r(0) = 9$. You may assume the existence of an algorithm A_0 with runtime $O(n^9)$.
Note: $r(i) \to 4$ for $i \to \infty$, but you do not need to prove this.

Exercise 3

30 **points**

(*Average Distances in Random Search Trees*)

We consider a random binary search tree for the n distinct keys $1, 2, \ldots, n$. For two keys $1 \leq a < b \leq n$, the distance between a and b is the length of the shortest path (number of edges) from a to b in the tree. We write $Q_{n,a,b}$ for the random variable describing this distance. We are interested in computing the expected total distances, i.e., the expectation of the random variable

$$P_n := \sum_{1 \leq a < b \leq n} Q_{n,a,b}.$$

(a) Prove that $P_n = \sum_v w(v) \cdot (n - w(v))$ where $w(v)$ is the random variable describing the number of nodes in the subtree rooted at v.
Hint: Analyze which shortest paths use the edge between v and the parent of v.

(b) Let $p_n := E[P_n]$. Prove that

$$p_n = \frac{1}{n} \sum_{r=1}^{n} \Big(p_{r-1} + p_{n-r} + x_{r-1} \cdot (n - r + 1) + x_{n-r} \cdot r + (r-1)(n-r+1) + r(n-r) \Big),$$

where x_n is the expected overall depth of a random search tree on n vertices, as in the lecture notes.
Hint: Decompose the shortest paths into pieces in the left subtree, right subtree, and the edges incident to the root.

(c) Use (b) to find a formula for p_n involving only n, p_{n-1}, and the value $\sum_{i=0}^{n-1} x_i$.

Exercise 4

20 **points**

(*Point Location in an Onion*)

Let $P \subset \mathbb{R}^2$ be a set of n points in general position. We define the *onion* of P

$$(R_0, V_0), (R_1, V_1), \ldots, (R_t, V_t)$$

as in the lecture notes.

Given a point $p \in \mathbb{R}^2$, such that $p \notin P$ and $P \cup \{p\}$ is in general position, the *onion depth* of p is the number d such that $p \in R_i$ for all $0 \leq i \leq d$ and $p \notin R_i$ for all $i > d$.

Describe two data structures allowing you to compute the onion depth of any query point p:

(a) This data structure can use $O(n)$ space and has to answer each query in $O(\log^2 n)$ time.

(b) This data structure can use $O(n^2)$ space and has to answer each query in $O(\log n)$ time.

For each subtask, explain in detail what the data structure stores and analyze its space usage. Describe how queries can be answered and analyze the time complexity. Additionally, give a rough sketch of how the data structure can be built from the point set P and analyze the preprocessing time needed. Your preprocessing strategy does not have to be optimal.