**ETH**

**Eidgenössische Technische Hochschule Zürich**
**Swiss Federal Institute of Technology Zurich**

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, Emo Welzl, Peter Widmayer

**Algorithms, Probability, and Computing** | **Final Exam** | **HS16**

**Candidate**

First name: ..........................................................

Last name: ..........................................................

Student ID (Legi) Nr.: ..........................................................

I attest with my signature that I was able to take the exam under regular conditions and that I have read and understood the general remarks below.

Signature: ..........................................................

**General remarks and instructions**

1. Check your exam documents for completeness (pages numbered until p. 14).
2. You can solve the exercises in any order. You should not be worried if you cannot solve all exercises! Not all points are required to get the best grade.
3. Immediately inform an assistant in case you are not able to take the exam under regular conditions. Later complaints will not be accepted.
4. Pencils are not allowed. Pencil-written solutions will not be reviewed.
5. No auxiliary material allowed. All electronic devices must be turned off and are not allowed to be on your desk. We will write the current time on the blackboard every 15 minutes.
6. Attempts to cheat/defraud lead to immediate exclusion from the exam and can have judicial consequences.
7. Provide only one solution to each exercise. Cancel invalid solutions clearly.
8. **All solutions must be understandable and well-founded. Write down the important thoughts in clear sentences and keywords. Unless stated otherwise, no points will be awarded for unfounded or incomprehensible solutions. You can write your solutions in English or German.**
9. You may use anything that has been introduced and proved in the lecture or in the exercise sessions without reproving it – unless we explicitly ask you to reproduce a proof. However, if you need something *different* than what we have shown, you must write a new proof or at least list all necessary changes.
10. Write your student-ID (**Legi-number**) on **all** sheets (and your name only on this cover sheet).

Good luck!

| | achieved points (maximum) |
|---|---|
| 1 | (30) |
| 2 | (20) |
| 3 | (20) |
| 4 | (25) |
| 5 | (25) |
| $\Sigma$ | (120) |

## Random(ized) Search Trees

### Exercise 1a (10 points)

Find a closed form for the following recurrence.

$$a_n = \begin{cases} 1 & \text{if } n = 1, \\ \frac{1}{n^2} \sum_{k=1}^{n-1} k a_k & \text{if } n \geq 2. \end{cases}$$

**Exercise 1b** **(10 points)**

Let $X_n$ denote the number of *nodes with two children* in a random binary search tree with $n \geq 0$ nodes, and let $x_n := \mathbf{E}[X_n]$. We have $x_0 = x_1 = x_2 = 0$ and $x_3 = \frac{1}{3}$.

(i) Find a recurrence formula for $x_n$, for all $n \geq 3$.
Use the usual method of conditioning on the root.

(ii) Compute $x_n$ for all $n \geq 1$.

*Remark:* You are free to either solve your recurrence from (i), or to use some other method. In the former case, if you get terms of the form $\frac{1}{m(m+1)}$ then it may help to write $\frac{1}{m(m+1)} = \frac{1}{m} - \frac{1}{m+1}$.

**Exercise 1c** **(10 points)**

Let us call a node in a binary tree *left-leaning* if the number of nodes in its left subtree is strictly larger than the number of nodes in its right subtree. Let $L_n$ denote the number of left-leaning nodes in a random search tree on $n$ nodes.

(i) Show: $\mathbf{E}[L_n] \leq \frac{n}{2}$.

(ii) Find constants $\alpha, \beta \in (0, 1)$ such that $\Pr[L_n \geq \alpha n] \leq \beta$.

## Point Location

### Exercise 2a (10 points)

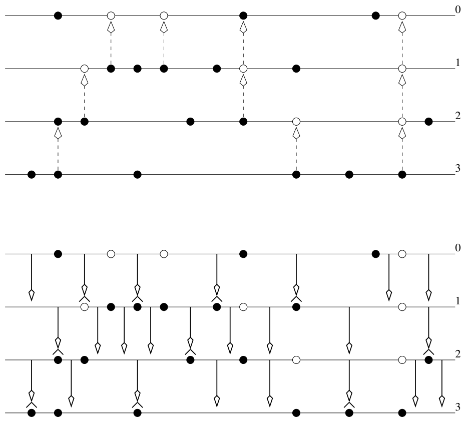
Let $S \subseteq \mathbb{R}^2$ be a set of $n$ points in the plane. We assume that the set $\{\text{dist}(p, q) : p, q \in S,\ p \neq q\}$ has exactly $\binom{n}{2}$ elements.

Let $(p_1, \ldots, p_n)$ be a permutation of the points in $S$, drawn uniformly at random from the set of all permutations of $S$. Imagine that the points are inserted into the plane one by one, in this order. After each insertion we take note of the longest distance between any two of the points currently present. If the longest distance among $p_1, \ldots, p_k$ is different from the longest distance among $p_1, \ldots, p_{k-1}$, then we call this a *longest distance change*. Here we do not count the insertion of $p_1$ or $p_2$ as a longest distance change (so the first time we may possibly observe a longest distance change is when inserting $p_3$).

Compute the expected number of longest distance changes. Compute exactly.

**Exercise 2b** **(10 points)**

On the side you see the illustration of the fractional cascading data structure from the lecture notes. The black bullets represent the original data (sets $S_0, \ldots, S_t$ that contain $n$ numbers overall).



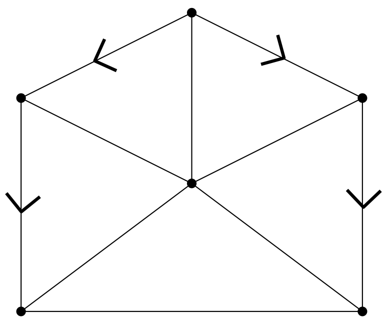

(i) What is the purpose of this data structure with respect to the sets $S_0, \ldots, S_t$ (i.e., what kind of query is it designed for)?

(ii) Once one has computed a fractional cascading data structure, how is a query performed, and how fast can this be done?

## Randomized Algebraic Algorithms

### Exercise 3a (7 points)

Consider the graph depicted below, which is already partially oriented. Orient the remaining edges in order to obtain a Pfaffian orientation. Do not forget to give a justification!



### Exercise 3b (7 points)

Let $a \in \{0,1,2\}^n$ be a vector, and let $r \in_{\text{u.a.r.}} \{0,1,2\}^n$ be a random vector.
Compute $\Pr\left[a^T r = 0 \mod 3\right]$.

*Hint:* You might want to distinguish two cases concerning what $a$ is.

**Exercise 3c** **(6 points)**

Let $p$ be a prime, and let $GF(p) = \{0, \ldots, p-1\}$ denote the set of integers modulo $p$.

Let $n \leq p$. Show that, among all matrices $A \in GF(p)^{n \times n}$, at most an $\frac{n}{p}$-fraction has $\det(A) = 0 \bmod p$.

## Linear Programming

### Exercise 4a (10 points)

A linear program *in standard form* is a linear program of the form

$$\text{maximize } \tilde{c}^T\tilde{x} \quad \text{subject to} \quad \tilde{A}\tilde{x} \leq \tilde{b} \text{ and } \tilde{x} \geq 0. \tag{1}$$

Recall that its dual is

$$\text{minimize } \tilde{b}^T\tilde{y} \quad \text{subject to} \quad \tilde{A}^T\tilde{y} \geq \tilde{c} \text{ and } \tilde{y} \geq 0. \tag{2}$$

Show that the dual of

$$\text{minimize } c^T x \quad \text{subject to} \quad Ax = b,\ x \geq 0 \tag{3}$$

is

$$\text{maximize } b^T y \quad \text{subject to} \quad A^T y \leq c \tag{4}$$

by changing (3) to standard form (1) and applying the definition of duality. Make sure that your calculation steps are clear! In particular, state explicitly what you choose for $\tilde{A}, \tilde{b}, \tilde{c}$.

**Exercise 4b** **(10 points)**

We are given a set $B = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^2$ of blue points and another set $R = \{(u_1, v_1), \ldots, (u_m, v_m)\} \subseteq \mathbb{R}^2$ of red points. For simplicity we assume $(0, 0) \in B$.

A *separating circle* is a circle

$$(x - a)^2 + (y - b)^2 = r^2$$

with the properties that all blue points lie strictly inside ($(x_i - a)^2 + (y_i - b)^2 < r^2$ for all $i = 1, \ldots, n$) and all red points lie strictly outside ($(u_i - a)^2 + (v_i - b)^2 > r^2$ for all $i = 1, \ldots, m$).
Formulate a linear program with the property that a separating circle exists if and only if the optimal value of your linear program is strictly positive. Do not forget to prove correctness.

*Hint:* You can get rid of squared variables by introducing one (!) auxiliary variable which takes the role of $r^2 - a^2 - b^2$. The assumption $(0, 0) \in B$ will be useful for proving correctness.

**Exercise 4c** **(5 points)**

Recall the Loose Spanning Tree LP for a graph $G = (V, E)$ with edge weights $c \in \mathbf{R}^E$:

$$\begin{array}{llcll} \text{minimize } c^T x & & & & \\ \text{subject to} & \sum_{e \in E} x_e & = & n-1 & \\ & \sum_{e \in \delta(S)} & \geq & 1 & \text{for all } S \subseteq V \text{ with } \emptyset \neq S \neq V, \\ & 1 \geq x_e & \geq & 0 & \text{for all } e \in E. \end{array}$$

Give an example of a weighted graph on $n = 6$ vertices on which a minimum spanning tree has cost at least $\frac{3}{2}c^T x^* > 0$. Here $x^*$ denotes, as usual, an optimal solution of the LP. Do not forget to justify why your example works.

## Local Graph Algorithms

### Exercise 5a (10 points)

In the questions below, you are allowed to ignore rounding issues. You can for example always assume that $\log(k)$ is an integer number.

(i) Prove or disprove: There is a deterministic LOCAL algorithm that, given any $k \geq 4$ and given a proper $k$-coloring of the network, computes a proper $\log_2(k)$-coloring in a single round – assuming such a coloring exists.

(ii) Let $\Delta$ denote, as usual, the maximal degree of the network graph.
Prove that there does *not* exist a deterministic LOCAL algorithm that does the following: Given a proper $k$-coloring of the network, where $\log(\log(\log(k))) \geq \Delta + 1$, it computes a $\log(\log(\log(k)))$-coloring in a single round.

**Exercise 5b** **(15 points)**

Recall the randomized LOCAL algorithm that computes a weak-diameter network decomposition with high probability: In order to compute the first subgraph $G_1$, every node $u \in V$ independently picks a random radius $r_u$ according to the distribution

$$\Pr[r_u = y] = \left(\frac{1}{2}\right)^y \quad (y = 1, 2, \ldots).$$

Now we defined, for each node $v \in V$,

$$\text{Center}(v) = \min\{u \in V : \text{dist}_G(u, v) \leq r_u\}.$$

(i) Based on the definitions above, define the clusters of $G_1$, and prove that these clusters are pairwise non-adjacent.

(ii) Prove that we have $\max_{u \in V} r_u \leq 3 \log_2 n$ with high probability. For simplicity you can assume that $\log_2(n)$ is an integer number. (We ask you to reproduce the steps of the calculation seen in class.)
*Hint:* First find an upper bound for $\Pr[r_u > 3 \log_2 n]$ for all $u \in V$; then use a union bound.

(iii) Using the previous questions, prove that with high probability the maximum cluster diameter in $G_1$ is at most $O(\log n)$.