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 — Final Exam — HS22

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: ....

Instructions

1. The exam consists of 5 exercises.
2. Check your exam documents for completeness (18 one-sided pages with 5 exercises).
3. You have **3 hours** to solve the exercises.
4. 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.
5. If you're unable to take the exam under regular conditions, immediately inform an assistant.
6. **Pencils** are not allowed. Pencil-written solutions will not be reviewed.
7. No auxiliary material is allowed. Electronic devices must be turned off and should not be on your desk. We will write the current time on the blackboard every 15 minutes.
8. Provide only one solution to each exercise. Please clearly mark/scratch the solutions that should not be graded.
9. **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.**
10. You may use anything that has been introduced and proven in the lecture or in the exercise sessions. You do not need to re-prove it, but you need to state it clearly and concretely. However, if you need something *different* than what we have shown, you must write a new proof or at least list all necessary changes.
11. Write your student-ID (**Legi-number**) on **all** sheets (and your name only on this cover sheet).

	achieved points (maximum)	reviewer's signature
1	(20)	
2	(20)	
3	(20)	
4	(20)	
5	(20)	
Σ	(100)	

Exercise 1: Random Binary Search Tree (20 points)

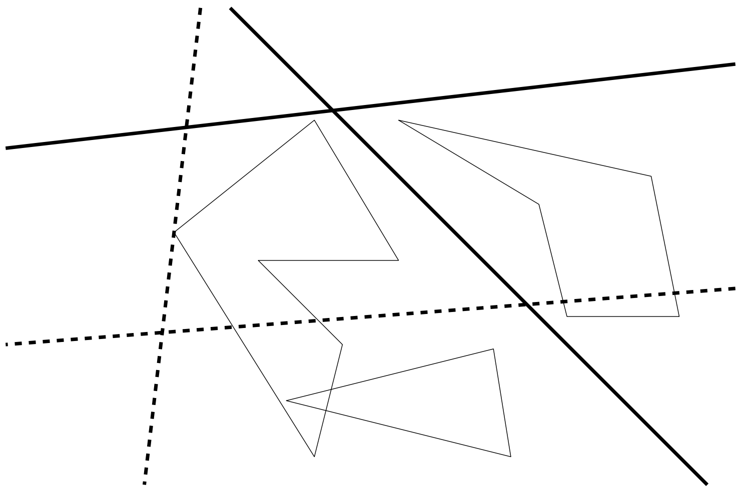
Let $n \geq 1$. Consider a random binary search tree on the nodes $\{1, \ldots, n\}$ as defined in the lecture. For every integer k with $k \geq 1$, let $\Delta_n^{(k)}$ be the number of edges in the tree for which the difference of the two endpoints is exactly k. Compute $\mathbf{E}\left[\Delta_n^{(k)}\right]$ for all $n \geq 1$, and all $k \geq 1$.

Hint: You may use that $\frac{1}{(d+1)d(d-1)} = \frac{1}{2(d+1)} - \frac{1}{d} + \frac{1}{2(d-1)}$ without proof.

Exercise 2: Point Location (20 points)

You are given an arrangement of some polygons in the plane. Each polygon is given by a list of its vertices, in order. Let n be the sum of the number of vertices of each polygon. A line ℓ is said to be *free*, if it does not intersect (this includes touching) any polygon.

See the figure below for an example arrangement where $n = 14$, with some lines which are free (bold), and some lines which are not free (dashed).



Your task is to design a data structure which allows you to decide in $O(\log n)$ time whether a non-vertical query line ℓ is free or not.

You should also describe how this data structure can be built. Analyze the preprocessing time you need and the size of your data structure, but these do not have to be optimal.

To make the task easier, assume that no two polygons share a vertex, and any two edges intersect in at most one point. Furthermore, assume that no edge of the input polygons is vertical.

Exercise 3: Linear Programming and Polytopes (20 points)

Let P be a set of n points in $\mathbb{R}^d$ in general position[1].

Devise an algorithm that computes the vertices of the convex hull of P.

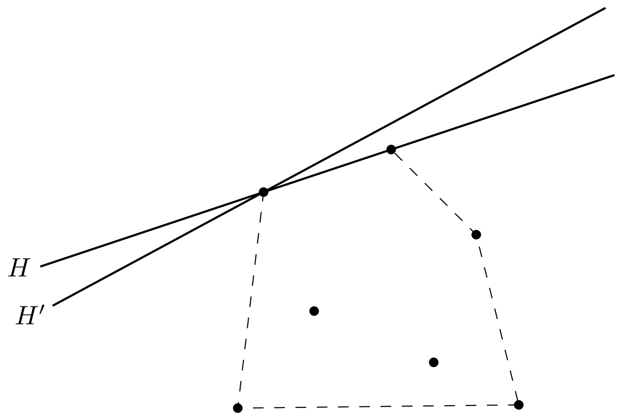
For the runtime of your algorithm, you may treat d as a constant. Let k be the (at the beginning unknown) number of vertices of the convex hull of P (i.e., the number of points in the output of your algorithm). Your algorithm may

- solve at most $O(n)$ linear programs, with each linear program containing $O(k)$ constraints, and
- perform at most $O(nk)$ additional arithmetic operations.

Hint: You may use (without proof) the following two facts holding for any set of points $P \subset \mathbb{R}^d$ in general position:

- For any hyperplane H such that P is contained in one of the half-spaces bounded by H, all points in $H \cap P$ are vertices of the convex hull of P.
- For any vertex p of the convex hull of P, there exists a hyperplane H' such that $P \cap H' = \{p\}$ and P is contained in one of the half-spaces bounded by H'.

These facts are illustrated below:



[1]General position in $\mathbb{R}^d$ means that for all $1 \leq k < d$, any affine subspace of dimension k contains at most $k+1$ points.

Exercise 4: Pfaffian Orientations (20 points)

For this exercise you may use that an orientation is Pfaffian if and only if every nice cycle is oddly oriented.

(a) **(6 points)** Prove that every connected planar graph G on $n \geq 3$ vertices has at least 2^{n-1} distinct Pfaffian orientations.

(b) **(7 points)** Let G be a graph on the vertex set V. Let C be a subset of V. Now consider a Pfaffian orientation $\vec{G}$ of G. Prove that the orientation obtained by flipping in $\vec{G}$ all edges with one endpoint in C and one endpoint in $V \setminus C$ is also a Pfaffian orientation.

(c) **(7 points)** Prove that every connected graph G on $n \geq 2$ vertices which has at least one Pfaffian orientation has at least 2^{n-1} distinct Pfaffian orientations.
You may use the result from (b) even if you did not prove it.

Exercise 5: Parallel Algorithms (20 points)

Throughout this exercise, we consider the ARBITRARY CRCW PRAM model.

Let G be an undirected, unweighted and connected graph with $n = |V(G)|$ vertices and $m = |E(G)|$ edges.

Recall that a *DFS tree of* G *rooted at* r is a spanning tree T of G, such that if we root T towards r, then for every edge $\{u, v\} \in E(G)$, either u is an ancestor of v or v is an ancestor of u. In other words, there does not exist an edge in G connecting two different branches of T. Note that there might exist more than one such DFS tree rooted at r.

We say that a subtree T' of G is a *partial* DFS tree rooted at r, if $r \in V(T')$ and T' can be extended to a DFS tree rooted at r. That is, there exists a DFS tree T rooted at r with $E(T) \supseteq E(T')$.

For the exercises below, you can assume that there exists a parallel algorithm $\mathcal{A}'$ that takes as input any undirected, unweighted and connected n-node m-edge graph G with $n \geq 2$ as well as a root r, and computes a partial DFS tree T' rooted at r with the following guarantee: each connected component of $G[V \setminus V(T')]$ has at most $n/2$ vertices. $\mathcal{A}'$ has work $O(m)$ and depth $O(\sqrt{n})$.

(a) **(10 points)** Let T' be a partial DFS tree and let C be a connected component in $G[V \setminus V(T')]$. We say that a node $v \in V(T')$ is *neighboring* the connected component C if there exists an edge $\{u, v\} \in E(G)$ with $u \in C$. We denote by $v_C \in V(T')$ the node of largest depth in T' neighboring C (the node v_C is uniquely defined, but you do not need to provide a proof for this).

Give a deterministic parallel algorithm that for any given partial DFS tree T' computes for every connected component C of $G[V \setminus V(T')]$ the node v_C in $O(m \log^2 n)$ work and $O(\log^2 n)$ depth.

(b) **(5 points)** Give a recursive algorithm for computing a DFS tree rooted at a given node r. You may use $\mathcal{A}'$ and an algorithm solving subtask (a), even if you did not solve that subtask. You do not have to prove that your algorithm is correct.

(c) **(5 points)** Give an asymptotically tight bound for the work and depth of the algorithm developed in the previous exercise.
You do not have to give a proof. Even if you do not provide the asymptotically tight bound, you can earn partial points if you give a suitable recurrence relation for both the work and depth.