ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Algorithms, Probability, and Computing | Special Assignment 1 | HS24

- 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 October 29th by 2 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.

Exercise 1

10 **points**

(*Random Binary Search Trees*)

Let $n \in \mathbb{N}$, the keys $2, 4, 8, \ldots, 2^n$ are inserted in a uniformly random order in a binary search tree. Let $k_1, k_2, \ldots, k_\ell$ denote the value of the keys on the right spine of the tree, i.e. on the path from the root to the biggest key. In this exercise, we want to compute $X_n = \mathrm{E}[k_1 + k_2, \cdots + k_\ell]$. Note that ℓ is a random number, k_1 is the value in the root node, and $k_\ell = 2^n$.

(a) Write X_n as a function of n and X_i for $i = 1, 2, \ldots, n-1$;

(b) Write X_n as a function of only n and X_{n-1};

(c) Write X_n in a closed form. The result can contain sums and products but should not be written as a function of some X_i for $i \in \mathbb{N}$.

Exercise 2

13 **points**

(*Number of spanning trees in the complete graph*)

The objective of the exercise is to find the number of distinct spanning trees in a complete graph with n vertices labelled with n distinct labels. An edge between vertices i and j is labelled $\{i, j\}$, and two spanning trees are considered different if they contain any differently labelled edges. For example, on the graph with $n = 3$ labelled vertices, we have 3 distinct spanning trees.

Let $K_n = (V, E)$ be the complete graph with n vertices and suppose that the vertices are labelled $1, 2, \ldots, n$. Let $R \subseteq V : |R| = k$ be a set of k vertices that are fixed (for simplicity, you can think of $R = \{1, 2, \ldots, k\}$). Denote with $T_{n,k}$ the number of (labelled) forests on $\{1, \ldots, n\}$ consisting of k trees whose roots are the vertices in R. Note that $T_{n,k}$ does not depend on R but only on its size and by fixing the roots, the number of possible forests decreases, eg. $T_{3,2} = 2$ as the vertices 1 and 2 cannot be in the same tree (using $R = \{1, 2\}$). For coherence, define also $T_{0,0} = 1$ and $T_{n,0} = 0$ for $n > 0$.

(a) Compute $T_{n,n}$ for $n \geq 1$.

(b) Show that for all $1 \leq k \leq n$,

$$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}.$$

Hint: It might be helpful to use different sets R in recursive cases.

(c) Using the recursive expression above, prove that

$$T_{n,k} = kn^{n-k-1}.$$

(d) Use the previous result to deduce the number of different spanning trees in a complete graph with n labelled vertices.

Exercise 3

12 **points**

(*Point Location*)
Given a collection S of n points in the plane and a constant $d > 0$, consider the problem of, given a query point q, finding all the points in S that are at most at distance d from q. Devise a data structure for this problem. In order to get the maximum score, the preprocessing time should be polynomial in the number of points n and if k is the number of points to report, the query should take $O(\log(n) + k)$ operations in expectation.

Exercise 4

15 **points**

(*Warm-up: Dynamic List Ranking*)
Suppose you have access to a *static* data structure which does the following: given a collection of n integer-valued elements, it takes $O(n \log n)$ preprocessing time; and after that, given a query element q, it returns the number of elements smaller than q in the initial input in $O(\log n)$ time[1]. The data structure is static in the sense that if we want to add a new element to the sequence, then we have to spend $O(n \log n)$ time and rebuild the data structure.
We want to use this data structure to solve the following *dynamic* problem. You are given a sequence of n elements $e_1, e_2, \ldots, e_n$ one at a time and each time, given a new element e_k, you have to tell what is the rank of this element, i.e. how many elements smaller than e are there in the sequence $e_1, e_2, \ldots, e_k$.

(a) Find an algorithm that solves the problem above using the data structure as a black box. After receiving n elements as input, the total amount of time required by the algorithm should be $O(n \log^2 n)$.

Note that you do not have access to an oracle that compares two elements without using the static data structure.
Hint: Use the data structures of geometrically increasing size.

(*Count points below a line dynamically*)
In this exercise, we devise a *dynamic* version of the data structure for reporting the number of points below a line with nearly the same total update time as the static version. We suppose that we start with an empty data structure. Then, at most n point insertions, at most n point removals, and an arbitrary amount of queries happen in some arbitrary order.

Our data structure should take $O(\log^2 n)$ time to process each query. Furthermore, the total time used for processing insertions and removals should be $O(n^2 \log n)$, and the total space occupied by the data structure should be $O(n^2)$.

(b) Reduce the problem above to the problem of counting the number of lines below a query point when only insertions and queries occur. Formally, you are required to demonstrate that, given a data structure capable of counting the number of lines below a query point—supporting both queries and insertions—it is possible to construct a data structure that addresses the problem described above while supporting insertions, deletions, and queries. Additionally, for any insertion of n points into both data structures, the total construction time of the two data structures must be asymptotically equivalent, as should the query and total insertion/deletion times. For a full score, your algorithm must work correctly even when asked to remove points that were not previously inserted.

(c) Devise an algorithm that solves the problem of counting the number of lines below a query point when only insertions and queries occur. In order to get a full score, your algorithm should take $O(\log^2 n)$ time to process each query, the total time used for processing insertions should be $O(n^2 \log n)$, and the total space occupied by the data structure should be $O(n^2)$.
Hint: Use the strategy from part (a).

[1] This can be achieved by sorting the elements and then performing binary search