ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Mohsen Ghaffari, Angelika Steger, David Steurer

Algorithms, Probability, and Computing **Special Assignment 1** **HS19**

- Write an exposition of your solution using a computer, where we strongly recommend to use LaTeX. **We do not grade hand-written solutions**.
- The solution is due on **Tuesday, October 29, 2019** by **2:15 pm**. Please bring a print-out of your solution with you to the lecture. If you cannot attend (and please only then), you may alternatively email your solution as a PDF to "chih-hung.liu@inf.ethz.ch" with subject "[APC19] SA1 (your full name)", likewise **until 2:15 pm**. We will send out a confirmation that we have received your file. Make sure you receive this confirmation within the day of the due date, otherwise complain timely.
- If you attend an exercise group, write down your group, so that we can give your assignment back to you in the exercise class.
- 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. 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.

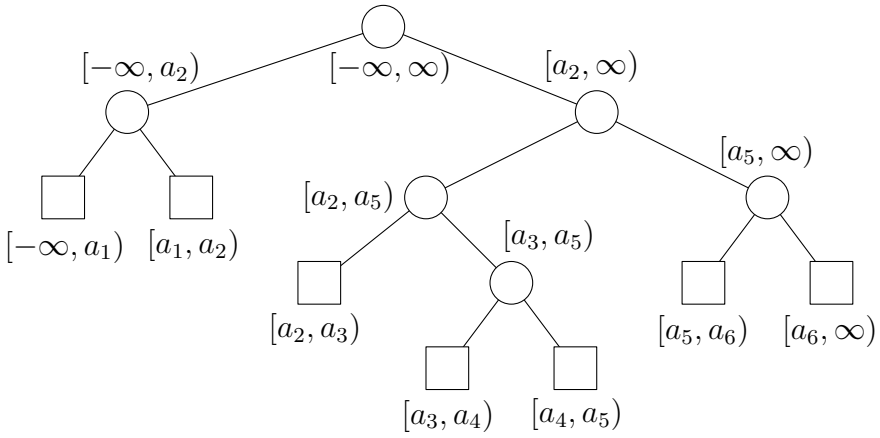

Figure 1: The binary tree corresponds to the permutation $(a_2, a_5, a_3, a_1, a_6, a_4)$.

Exercise 1

20 **points**

(*Random Search Trees*) We have learned about random search trees (Chapter 2) and how to partition the real line into $n + 1$ intervals by inserting n distinct numbers in a random order (the warm-up of Section 3.4). In this exercise, we consider a combination of these two concepts.

Let S be a set of n distinct real numbers, and let $(a_1, a_2, \ldots, a_n)$ be the sorted permutation of elements in S, so that the n numbers in S partition the real line into $n + 1$ disjoint intervals, $[-\infty, a_1), [a_1, a_2), \ldots, [a_{n-1}, a_n), [a_n, \infty)$.[1] Consider a family of binary trees whose leaves represent the $n + 1$ intervals and whose internal nodes represent the union of intervals of leaves in their own subtrees. Fig. 1 shows an illustration.

Generate a random permutation $P = (p_1, p_2, \ldots, p_n)$ on S and build such a binary tree by inserting $p_1, p_2, \ldots, p_n$ sequentially: To insert p_i, (1) locate the leaf v whose interval $[\ell, r)$ contains p_i , (2) create two nodes v_ℓ and v_r to represent the two intervals $[\ell, p_i)$ and $[p_i, r)$, respectively, and (3) insert v_ℓ and v_r as the left and right children of v, respectively.

Let $D_1(n)$ denote the random variable for the depth of the leftmost leaf in the above binary tree. Complete the following tasks.

(a) Compute $\mathbb{E}[D_1(0)]$ and $\mathbb{E}[D_1(1)]$.

(b) For $n \geq 2$ and $k \in [1, n]$, give a recursive formula for

$$\mathbb{E}[D_1(n) \mid p_1 = a_k].$$

(c) Give a recursive formula for $\mathbb{E}[D_1(n)]$.

(d) Calculate $\mathbb{E}[D_1(n)]$.

[1] For simplifying the description, we use $[\infty, a_1)$ as the first interval instead of (∞, a_1).

Exercise 2

20 **points**

(*Randomized Incremental Construction*) You have learned a randomized incremental construction algorithm for trapezoidal decompositions (Section 3.4). Actually, for Voronoi diagrams (Section 3.3), there also exist randomized incremental construction algorithms. But we will not ask you to create such an algorithm. Instead, you will need to analyze some properties of the randomized incremental construction.

Let S be a set of n points in the plane, and let $(p_1, p_2, \ldots, p_n)$ be a random permutation on S. Assume we build the Voronoi diagram incrementally by inserting $p_1, p_2, \ldots, p_n$. For a fixed point x, the nearest neighbor of x changes after the insertion of p_i if $d(x, p_i) < \min_{j<i} d(x, p_j)$, where $d(y, z)$ is the Euclidean distance between two points, y and z, in the plane.

Complete the following tasks:

(a) For a fixed point x in the plane, prove that its nearest neighbor changes $\mathcal{O}(\log n)$ times with probability at least $1 - \frac{1}{n^6}$ during the randomized incremental construction.

Hint: If p_1 is the k-th nearest neighbor of x in S, after the insertion of p_1, the problem size is reduced from n to $k-1$. By this reasoning, after every insertion, the problem size becomes at most half with probability at least $\frac{1}{3}$. After being halved $\lfloor \log_2 n \rfloor$ times, the problem size will become a constant. Consider an experiment of flipping a coin with head probability of $\frac{1}{3}$ until getting $\lfloor \log_2 n \rfloor$ heads, and analyze the number of flips. Apply the following Chernoff bound: if X is the sum of independent Bernoulli random variables,

$$\Pr[X < (1-\delta) \cdot \mathbb{E}[X]] \leq e^{-\frac{\mathbb{E}[X] \cdot \delta^2}{2}}, \qquad 0 < \delta < 1.$$

(b) For a point x in the plane, we can define a sequence on the points of S according to their distances from x. (Some points in S may share the same distance from x. We can arbitrarily label the points in S from 1 to n, so that when two points share the same distance from x, we order them according to their labels.)

Prove that there are $\mathcal{O}(n^4)$ such sequences on S over all points x in the plane.

Hint: You can apply the following fact without proving it: for m lines in the plane, the resulting arrangement has $\mathcal{O}(m^2)$ faces, $\mathcal{O}(m^2)$ edges, and $\mathcal{O}(m^2)$ vertices.

(c) Prove that the probability that every point in the plane changes its nearest neighbor $\mathcal{O}(\log n)$ times during the randomized incremental construction is at least $1 - \frac{1}{n}$ for sufficiently large n. (Even if you do not solve (a) and (b), you can apply their statements.)

Exercise 3

20 **points**

(*Bootstrapping*) Given a set S of n distinct natural numbers, we attempt to sort the elements in S. However, we cannot compare elements in S directly, but are only able to ask an oracle if $a > b$ for two elements $a, b \in S$. Unfortunately, the oracle gives a wrong outcome with probability $1/64$ for each query, and once it answers a query, it will not change the outcome anymore, i.e., the repetitions of comparing two elements always get the same outcome. Due to the unreliability of the oracle, it is impossible to compute a correctly sorted permutation on S. Hence, the resulting permutation by an algorithm is evaluated by "dislocations" as follows:

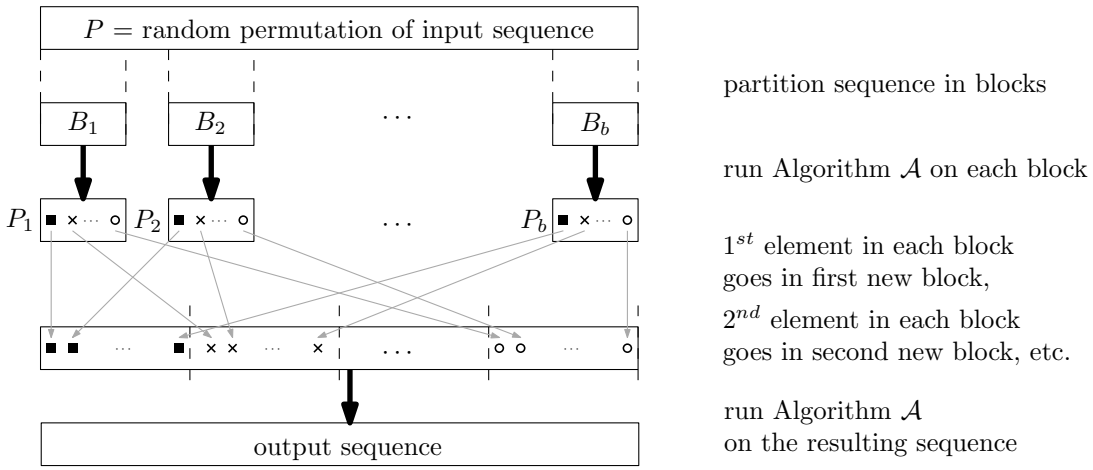

Figure 2: The four-step algorithm in Exercise 3(a).

For any permutation P on S and any element $x \in S$, the *dislocation* of x with respect to P is the absolute difference between the true rank of x in S and the position of x in P (i.e., the index of x along P). The *maximum dislocation* of P is the *maximum* over the dislocations of all elements in S with respect to P.

Consider the following *black-box* algorithm:

> **Algorithm** $\mathcal{A}$. Let S' be a subset of S, let P' be a permutation on S' and let D' be the maximum dislocation of P'. Given P' and an integer $W \geq D'$ as the input, Algorithm $\mathcal{A}$ sorts S' using $\mathcal{O}(|S'| \cdot W)$ comparisons such that with probability $1 - \frac{1}{n^8}$, the maximum dislocation of the resulting permutation is $\mathcal{O}(\log n)$.[2]

By directly applying Algorithm $\mathcal{A}$ on S, one can sort S using $\mathcal{O}(n^2)$ comparisons such that with probability $1 - \frac{1}{n^8}$, the maximum dislocation of the resulting permutation is $\mathcal{O}(\log n)$.

In this exercise, we attempt to reduce the number of comparisons by recursively applying Algorithm $\mathcal{A}$, i.e., *bootstrapping technique*, where the success probability to achieve $\mathcal{O}(\log n)$ maximum dislocation must be at least $1 - \frac{1}{n}$.

(a) Consider the following four-step algorithm (as illustrated in Figure 2):

1. Generate a *random permutation* P on S and partition this permutation P into $b = \frac{n}{\beta}$ blocks $B_1, B_2, \ldots, B_b$ of the same size β. (You will choose β later.)
2. Run Algorithm $\mathcal{A}$ with $W = \beta$ on each block B_i to obtain a permutation P_i.
3. Combine all the permutations P_i together into a permutation P' as follows: Place the first elements of P_i for $1 \leq i \leq b$, then the second elements, and so on. In other words, the j-th element in B_i is placed as the $\big((j-1) * b + i\big)$-th element in P'.
4. Run Algorithm $\mathcal{A}$ with $W = D$ on this new permutation P'. (You will choose D later.)

[2] Note that the maximum dislocation and the probability are related to n rather than $|S'|$. This is because we will apply $\mathcal{A}$ to subsets of S while guaranteeing the entire success probability of at least $1 - \frac{1}{n}$.

Complete the following tasks:

i. Analyze the number of comparisons in terms of n, β, and D.

ii. You are given a fact that with probability $1 - \frac{1}{n^7}$, the maximum dislocation of P' is at most $c \cdot \frac{n}{\sqrt{\beta}} \log n$ for some constant c, and you are also given the value of c, so that you can decide the value of D. (Note that you do not need to prove the fact.)

Formulate the number of comparisons in terms of n and β.

iii. Choose the value of β, so that the number of comparisons is $\tilde{\mathcal{O}}(n^{\frac{5}{3}})$. (Recall that the $\tilde{\mathcal{O}}$-notation omits polylog factors.)

(b) The four-step algorithm in (a) only performs one recursion level. Design an algorithm with two recursion levels, so that the number of comparisons is reduced to $\tilde{\mathcal{O}}(n^{\frac{11}{7}})$.

One can view the four-step algorithm in (a) as Algorithm $\mathcal{A}'$, and build an almost identical four-step algorithm except that its second step uses Algorithm $\mathcal{A}'$ instead of "Algorithm $\mathcal{A}$ with $W = \beta$." Again, you are given a fact that if the maximum dislocation of P_i is $\mathcal{O}(\log n)$ for $1 \le i \le b$, with probability $1 - \frac{1}{n^8}$, the maximum dislocation of P' is at most $c \cdot \frac{n}{\sqrt{\beta}} \log n$ for some constant c, and you are also given the value of c, so that you can decide the value of D.

(c) We want to minimize the number of comparisons using more recursion levels. Let h be the number of recursion levels, and let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_h$ be the sequence of algorithms in which $\mathcal{A}_0 = \mathcal{A}$ and $\mathcal{A}_i$ applies $\mathcal{A}_{i-1}$.

Answer the following questions.

i. If $\mathcal{A}_i$ requires $\tilde{\mathcal{O}}(n^{a_i})$ comparisons, how many comparisons does A_{i+1} require?

ii. In 3.(c).i, you have derived a recursive formula from a_{i+1} to a_i. Please use this formula to prove that $a_i = \frac{1}{2}(\frac{1}{2^{i+1}-1} + 3)$.

iii. How should you choose h, so that $n^{a_h} = \tilde{\mathcal{O}}(n^{\frac{3}{2}})$?