Homework assignment 3

2-AIN-205, Spring 2025

Termín: 12.5.2025, 22:00, google classroom

Before you start working on this homework assignment, read general instructions at the end of this document. You can write you solutions in Slovak or English. The solutions must be your work. Do not copy from others and do not attempt to find the solutions in literature or on the internet! Grade of 40 points will be considered as "full marks" for this assignment. The rest are bonus points.

1. [*20 points*] **Random permutations.** We will generate random permutations using the following method. We start with an array S which is initialized as $S[i] = i$. For all i between 2 and n, we exchange the element $S[i]$ with element $S[\text{rand}(1, i)]$, where $\text{rand}(a, b)$ is a function that returns a random number between a and b with a uniform probability distribution.

   a) Show that this algorithm is able to generate each of the $n!$ permutations and that each permutation is generated with equal probability.

   b) If we assume that random number rand(a, b) can be generated in $O(1)$ time, then the algorithm clearly runs in $O(n)$ time. How would you implement $\text{rand}(a, b)$ if you only had an ability to generate one random bit (number 0 or 1) at a time? Show the algorithm and state the worst-case and expected running time.

2. [*20 points*] **A random walk on a tree.** We are given a binary tree with n vertices, where each internal vertex has exactly two children. We start in the root of the tree and in each step we choose either left or right child uniformly randomly and move to this new vertex. We repeat this until we are in the leaf of the tree. Prove that the expected number of steps of this algorithm is upper bounded by $\log_2(n + 1)$.

   Hint: The input tree does not have to be balanced. Try induction on the number of vertices.

3. [*20 points*] **Programming task** (see general instructions).
   Submission link: `https://judge.ksp.sk/public/submit/87ec349ed29853b8ad9b5d2d4b12fb37/`

   We are given an array of n integer numbers $a_1, a_2, \ldots, a_n$ and number k. Determine whether the array contains two subarrays of length k that contain exactly the same numbers in the same order, i.e. whether there exist two number $i < j$ such that sequences $a_i, a_{i+1}, \ldots, a_{i+k-1}$ and $a_j, a_{j+1}, \ldots, a_{j+k-1}$ are exactly the same.

   Input format: In the first row there are numbers n and k. In the second row there is n numbers $a_1, \ldots, a_n$, where $0 \leq a_i \leq 100$.

   Output format: The output is a single line with numbers i and j (separated by a space), or -1 if the solution does not exist. If there are multiple solutions, output the solution with the smallest i and if there are multiple solutions with the same i, output the smallest corresponding value of j.

   Constraints: Your program should give a correct answer for inputs where $n, k \leq 1000000$.

   Example input:

   ```
   5 2
   1 1 1 1 1
   ```

   Example output:

   ```
   1 2
   ```

   Example input:

   ```
   5 2
   1 2 3 1 2
   ```

   Example output:

   ```
   1 4
   ```

   Example input:

   ```
   5 2
   1 2 3 4 5
   ```

   Example output:

   ```
   -1
   ```

General Instructions

Theoretical tasks. Submit answers to theoretical tasks in one pdf file per task in google classroom (your answer can be typeset or scanned, but please make sure that everything is legible and easy to read). No late submissions are allowed.

Write your solutions so that they contain all information necessary to easily understand them, but at the same time try to aim for brevity. Prove all claims, including in the cases when it is not explicitly written in the problem statement.

If the task is about solving an algorithmic problem, submit the best algorithm you can design. The first criterion is that the algorithm *is correct*, the second criterion is the *running time and memory complexity*. Correct and slow algorithm is worth more points than fast incorrect algorithm. Inefficient but correct algorithms will always receive at least 50% of points. In your solution, you should:

- Describe the main idea of the algorithm.
- Write a pseudocode for your algorithm.
- Prove the correctness of your algorithm.
- Analyse its running time and memory complexity.

Programming tasks. You need to submit a functional program, no written part is required. Your solutions will be evaluated immediately on several inputs and you will learn number of points (there are several sets of inputs and you only gain points if you solve correctly all inputs in a particular set). You can submit a solution several times, we will only take into account the last submitted solution before the deadline. To submit your solution, please use "Univerzita Komenského" login on `https://judge.ksp.sk/` system. You can find instructions on how to submit your programming task solutions at `https://judge.ksp.sk/docs/`.