Homework 3

Strukture podataka i algoritmi I - I053

Homework instructions

The submission deadline is **November 8, 2023** at 9:00. You can type the tasks in LaTeX or write them by hand and scan them. Programming tasks should be submitted as .cpp files. All files need to be submitted to Teams. You can achieve a maximum of 100 points.

Task 1 (20 pts.)**.** Implement the Quicksort algorithm that sorts a vector inplace. Pick the middle element of the vector as the pivot. What's the time and space complexity of this algorithm? Give a short explanation.

Task 2 (20 pts.)**.** Use the fast exponentiation algoirithm to compute the n-th Fibonacci number in $O(\log n)$ time and $O(1)$ space. Use the identity

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1}. \end{bmatrix}$$

Task 3 (20 pts.)**.** Implement the binary search algorithm.

Task 4 (20 pts.)**.** We call an array with n elements unimodal if there exists $1 < i < n$ such that

$$a_1 < a_2 < \cdots < a_i > a_{i+1} > \cdots > a_n.$$

Modify the binary search algorithm to find the maximum element of an unimodal array in $O(\log n)$ time.

Task 5 (20 pts.)**.** You have an array of n integers. You need to split it into exactly k subarrays such that the maximum sum in a subarray is as small as possible. **A subarray is a contiguous part an array**.
The first line of input contains $1 \leq n \leq 2 * 10^5$ and $1 \leq k \leq n$, the array size and the required number of subarrays.
The second line of input contains n integers, $1 \leq a_i \leq 10^9$, the array elements.
The only output line should contain a single integer, the maximum sum of a subarray in the optimal split.

INPUT	**INPUT**
5 3	4 3
2 4 7 3 5	1 2 4 3
OUTPUT	**OUTPUT**
8	4

In the first test case, the optimal split is $[2, 4], [7], [3, 5]$.
Some hints:

- How can you use binary search for this problem?
- Since n can be $2 * 10^5$, your algorithm shouldn't be slower that $O(n \log n)$. The running time on your computer shouldn't be slower than a few seconds.
- Try generating more inputs to check if your implementation is correct.
- If you're having trouble reading larger inputs, read a blog post regarding reading files in competitive programming by -is-this-fft-.

Task 6 (**extra** 20 pts.)**.** Your task is to compute the number of ways you can get a sum of n after throwing 6-sided playing dice a finite number of times **in** $O(\log n)$ **time**. For example, if $n = 9$, some possibilities are $1 + 3 + 4 + 1$ and $3 + 3 + 3$.
You need to modify the fast Fibonacci algorithm from task 2.