Lab Assignment 2

CS 302 – Advanced Data Structures and File Processing

Problem 1

You are given an *unsorted* array A of non-negative integers. It contains the same number of even and of odd numbers. Implement an algorithm such that all odd numbers in A are at odd indices and all even numbers are at even indices. Additionally, even numbers should be sorted in increasing order and odd numbers should be sorted in decreasing order. Your algorithm should run in $\mathcal{O}(n \log n)$ time.

Problem 2

You are given an array A with n distinct elements. Implement an $\mathcal{O}(n \log n)$-time algorithm that creates an array B where all elements are in range from 0 to $n-1$ and where the order of elements is the same as in A. That is, 0 has the same index in B as the smallest element in A, 1 has the same index in B as the second smallest element in A, and so on. For example, if $A = [4, 9, 1, 5]$, then $B = [1, 3, 0, 2]$.

Hint: Use key-value pairs where the key is the original element in A and the value is its index in A. The given file contains a class `IntKVPair`. It represents a key-value pair of integers and an `IntKVPair`-array can be sorted by their keys.

Implementation

You are given a file *Lab2.java* (which you can download from canvas). The file contains a class `Lab2` with the two functions `problem1` and `problem2`. Implement your solutions in the corresponding functions. **Do not make any changes outside of these two functions (e. g. by adding helper functions); such changes will be undone.** Do not output anything to the terminal.

The program already implemented in the file `Lab2.java` randomly generates test cases. The seed of the random number generator is set to ensure the same test cases whenever to program is executed. Note that the purpose of the tests is for you to avoid major mistakes. **Passing all given tests *does not* imply that your algorithm is correct, especially that is has the expected runtime.**

You may use the function `java.util.Arrays.sort()` to sort arrays; you can assume it runs in $\mathcal{O}(n \log n)$ time. Do not use hash tables or similar data structures.

Submission

For your submission, upload the file *Lab2.java* with your implementation to canvas.

This is an individual assignment. Therefore, a submission is required from each student.

Deadline: is on Canvas.