Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Karel Kubicek, Johannes Lengler

14 October 2019

Algorithms & Data Structures Programming task 5 HS 19

Rules: Follow the general course policy and a few extra programming rules:

- All the submission have to be programmed independently. If you struggle solving the task on your own, you can use Moodle, or you can discuss it with your peers, but afterwards, you should be able to write your solution on your own.
- In the Moodle discussion, follow the No-Spoiler-Policy. Mind that you should never post your code online, not even partially.
- During the exam, you would not have access to Internet except Java documentation and Judge. We highly recommend you to practice for this and solving the tasks without searching for solution online.
- Please stick to the API (defined functions and their usage) given in the programming template. You can write auxiliary functions, but you are not allowed to use extra libraries. If we are asking you to follow some approach for solving the task, you can be awarded the bonus points only if the solution is the defined one. If you need feedback for your alternative approach, every project has local defined tests that should provide same information about functionality as Judge. We can randomly check if you follow the rules and treat violations as cheating.
- Although you are given two weeks to solve the tasks, we highly recommend you to solve at least one of the problems in the first week.
- You may submit to the judge as often as you want. The best submission counts.

The solutions for this sheet need to be submitted to the judge by Sunday, 27 October, 23:59:59.
Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 5.1 *Binary search* (**2 bonus point**).

Your task is to program an algorithm for finding the index of an item in a sorted array. However, other than in the lecture, an item may occur several times. In this case, you have to return the index of the right-most one. If the item is not in the array, return -1.

Please find a more detailed description of the task in the `Main.java` file. The task is to implement the method `Main.binary_search`. The data format is described by the method `Main.read`. The Eclipse project also contains a set of basic tests, execute them by running `JUnitTest.java`.

You get one point for each passing test set. One of them, called `unique`, contains only unique numbers, so the standard binary search algorithm solves it. The other one called `repeat`, tests searching in an array with multiple repeating elements. Your solution has to be in $\mathcal{O}(\log(n))$, where n is the length of the input sorted array to pass both test sets correctly.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H5P1`. The enrollment password is “`asymptotic`”.

Exercise 5.2 *Exponentiation of complex numbers* **(2 bonus point)**.

In this exercise, you are going to develop an algorithm to compute powers a^n, with $a \in \mathbb{C}$ and $n \in \mathbb{N}$. The base a is a so-called Gaussian integer, i.e., a complex number whose real and imaginary parts are both integers.

Please find more detailed description in the `Main.java` file. The task is to implement methods `Main.pow` and `Main.Complex.multiply`, where you can find more details. The data format is described by method `Main.read`. The Eclipse project also contains a set of basic tests, run them by running `JUnitTest.java`.

The judge is again testing two test sets, with each for one point. To pass `power_of_two`, it is fine if your solution can exponentiate only by exponents that are a power of 2. But you have to do so in $\mathcal{O}(\log(n))$. The test set `generic` requires you to exponentiate with an arbitrary exponent in $\mathcal{O}(\log(n))$.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H5P2`. The enrollment password is "`asymptotic`".

***Exercise 5.3** *Sorting array of three numbers* **(challenge exercise, no bonus points)**.

You are given an array containing only numbers 1, 2, and 3. You are required to sort this array in ascending order with three restrictions:

1. You have to move the numbers in the array using the predefined swap function. I.e., you cannot just count number of occurrences and then rewrite the content of the array. This is reasonable restriction for many cases when you operate with more complex objects than just numbers – sorting people into three groups, when each person has also other attributes than just a number.
2. You have to do the task in place, without allocating any extra array.
3. Your code has to run in O(n), where n is the length of the array.

More precisely, we measure the number of swap calls, and it has to be less than $1.2n$. For reference, in average it is possible to use roughly $0.5n$ swap calls.

Submission: As this task is not graded, there is no Judge submission. Follow the tests to test whether your algorithm works.