Algorithms and Data Structures
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH08-320201
February 22nd, 2019

Homework 3

- Submit one ZIP file per homework sheet which contains one PDF file (including pictures, computations, formulas, explanations, etc.) and your source code file(s) with one makefile and without adding executable, object or temporary files.
- The implementations of algorithms has to be done using C, C++, Python or Java.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320201/2019_1/Grading_Criteria_ADS.pdf

Problem 3.1 *Fibonacci Numbers* (15 points)

(a) (6 points) Implement all four methods to compute Fibonacci numbers that were discussed in the lecture: (1) naive recursive, (2) bottom up, (3) closed form, and (4) using the matrix representation.

(b) (4 points) Sample and measure the running times of all four methods for increasing n. For each method, stop the sampling when the running time exceeds some fixed amount of time (same for all methods). If needed, you may use classes or structs for large numbers (self-written or library components). Create a table with your results (max. 1 page).
Hint: The "gap" between samples should increase the larger n gets,
e.g. $n \in \{0, 1, 2, 3, 4, 5, 6, 8, 10, 13, 16, 20, 25, 32, 40, 50, 63, ...\}$.

(c) (2 points) For the same n, do all methods always return the same Fibonacci number? Explain your answer.

(d) (3 points) Plot your results in a line plot, so that the four approaches can be easily compared. Briefly interpret your results.
Hint: Use logarithmic scales for your plot.

Problem 3.2 *Divide & Conquer and Solving Recurrences* (11 points)

Consider the problem of multiplying two large integers a and b with n bits each (they are so large in terms of digits that you cannot store them in any basic data type like `long long int` or similar). You can assume that addition, substraction, and bit shifting can be done in linear time, i.e., in $\Theta(n)$.

(a) (2 points) Derive the asymptotic time complexity depending on the number of bits n for a brute-force implementation of the multiplication.

(b) (4 points) Derive a Divide & Conquer algorithm for the given problem by splitting the problem into two subproblems. For simplicity you can assume n to be a power of 2.

(c) (1 point) Derive a recurrence for the time complexity of the Divide & Conquer algorithm you developed for subpoint (b).

(d) (3 points) Solve the recurrence in subpoint (c) using the recursion tree method.

(e) (1 point) Validate the solution in subpoint (d) by using the master theorem to solve the recurrence again.

How to submit your solutions

You can submit your solutions via *Grader* at **`https://grader.eecs.jacobs-university.de`** as a generated PDF file and/or source code files.
If there are problems with *Grader* (but only then), you can submit the file by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that starts with CH08-320201**.

Please note, that after the deadline it will not be possible to submit solutions. It is useless to send solutions by mail, because they will not be graded.

This homework is due by Friday, March 1st, 23:00.