Algorithms and Data Structures
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-231-A
March 16th, 2020

Homework 7

- 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/ch_231_a/2020_1/Grading_Criteria_ADS.pdf

Problem 7.1 *Sorting in Linear Time* (19 points)

(a) (3 points) Implement the algorithm for Counting Sort and then use it to sort the sequence $< 9, 1, 6, 7, 6, 2, 1 >$.

(b) (3 points) Implement the algorithm for Bucket Sort and then use it to sort the sequence $< 0.9, 0.1, 0.6, 0.7, 0.6, 0.3, 0.1 >$.

(c) (3 points) Given n integers in the range 0 to k, design and write down an algorithm (only pseudocode) with pre-processing time $\Theta(n+k)$ for building auxiliary storage, which counts in $O(1)$ how many of the integers fall into the interval $[a, b]$.

use classic radix sort because the other one requires recursion and this asks for linear

(d) (4 points) Given a sequence of n English words of length k, implement an algorithm that sorts them alphabetically in $\Theta(n)$. Let k and n be flexible, i.e., automatically determined when reading the input sequence. You can consider k to behave like a constant in comparison with n. Example sequence of words to sort:
$< "word", "category", "cat", "new", "news", "world", "bear", "at", "work", "time" >$.

(e) (2 points) Given any input sequence of length n, determine the worst-case time complexity for Bucket Sort. Give an example of a worst-case scenario and the prove corresponding the complexity.

(f) **Bonus** (4 points) Given n 2D points that are uniformly randomly distributed within the unit circle, design and write down an algorithm (only pseudocode) that sorts in linear time the points by increasing Euclidean distance to the circle's origin. Write also a pseudocode function for the computation of the Euclidean distance between two 2D points.
can be partitioned in n+1 pieces

Problem 7.2 *Radix Sort* (9 points)

Consider Hollerith's original version of the Radix Sort, i.e., a Radix Sort that starts with the most significant bit and propagates iteratively to the least significant bit (instead of vice versa).

(a) (4 points) Implement Hollerith's original version of the Radix Sort.

(b) (3 points) Determine and prove the asymptotic time complexity and the asymptotic storage space required for your implementation.

(c) **Bonus** (2 points) Write down the pseudocode for an algorithm which sorts n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

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 CH-231-A**.

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 Monday, March 23rd, 23:00.