Assignment 5

CSCI 3110: Design and Analysis of Algorithms

Due June 18, 2019

Banner ID: ____________________ Name: ______________________________

Banner ID: ____________________ Name: ______________________________

Banner ID: ____________________ Name: ______________________________

Assignments are due on the due date before class and have to include this cover page. Plagiarism in assignment answers will not be tolerated. By submitting their answers to this assignment, the authors named above declare that its content is their original work and that they did not use any sources for its preparation other than the class notes, the textbook, and ones explicitly acknowledged in the answers. Any suspected act of plagiarism will be reported to the Faculty's Academic Integrity Officer and possibly to the Senate Discipline Committee. The penalty for academic dishonesty may range from failing the course to expulsion from the university, in accordance with Dalhousie University's regulations regarding academic integrity.

1. (10 pts) Develop an $\mathrm{O}(n \lg n)$-time algorithm that solves the following problem: Given an unsorted array of n real numbers and a real number x, decide whether there are two numbers y and z in this array such that $x = 2y - z$. The output of your algorithm should be "No" if no such numbers y and z exist; otherwise, it should report y and z. (Note that there may be more than one such pair of numbers. It is sufficient to report one of them.) Prove that your algorithm is correct. Argue briefly why your algorithm takes $\mathrm{O}(n \lg n)$ time.

2. (10 pts) Consider the task of searching a sorted array $A[1 \ldots n]$ for a given element x: a task we usually perform by binary search in time $\mathrm{O}(\lg n)$. Show that any algorithm that accesses the array only via comparisons (that is, by asking questions of the form "is $A[i] \leq z$?"), must take $\Omega(\lg n)$ steps.

3. A k-way merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

   (a) (5 pts) Here's one strategy: Using the merge procedure, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?

   (b) (5 pts) Give a more efficient solution to this problem, using divide-and-conquer. What is its time complexity in terms of k and n?

4. (a) (10 pts) Show that any array of integers $x[1 \ldots n]$ can be sorted in $\mathrm{O}(n + M)$ time, where $M = \max_i x_i - \min_i x_i$.

   (b) (Bonus: 5 pts) For small M, this is linear time: why doesn't the $\Omega(n \lg n)$ lower bound apply in this case?