Homework 2

ECE 345 Algorithms and Data Structures
Fall Semester, 2020

Due: Tuesday, October 13, <u>2PM</u>

- All page numbers are from 2009 edition of Cormen, Leiserson, Rivest and Stein.

- For each algorithm you asked to design you should give a detailed *description* of the idea, proof of algorithm correctness, termination, analysis of time and space complexity. If not, your answer will be incomplete and you will miss credit. You are allowed to refer to pages in the textbook.

- Do not write C code! When asked to describe an algorithm give analytical pseudocode.

- **Read the handout with the instructions on how to submit your Homework online. Failure to adhere to those instructions may disqualify you and you may receive a mark of zero on your HW.**

- Write *clearly*, if we cannot understand what you write you may not get credit for the question. Be as formal as possible in your answers. Don't forget to include your name(s) and student number(s) on the front page!

- No Junk Clause: For any question, if you don't know the answer, you may write "I DON'T KNOW" in order to receive 20% of the marks.

---

1. [**Search, 15 points**] You are given a set S of n real numbers where for each element $x \in S, 1 < |x| < 10$. You are also given a black box that returns true if there is some combination $(x, y, z) \in S$ such that $x \cdot y = z$. Devise an algorithm where you can determine whether there exists $(x, y, z) \in S$ such that $x \cdot y + z = 0$. Assuming the black box runs in constant time, the algorithm should run in O(n) time.

   (*Hint*: Construct a set S' so that the following holds: there exist $x', y', z' \in S'$ such that $x' \cdot y' = z'$ if and only if there exist $a, b, c \in S$ such that $x \cdot y + z = 0$.)

2. [**Search, 15 points**] Say the black box from the previous question returns only z. Write an O($n \log n$) algorithm to find x and y. In other words, given a set S with n real numbers and a number z, write an O($n \log n$) algorithm to find $(x, y) \in S$ such that $x \cdot y = z$.

3. [**Sorting, 10 points**]

   You are given a sorted array with n distinct integers. Suddenly, the elements randomly change value. Each element either increases by 2, decreases by 2, or stays the same. Describe a way to sort this list again (by the element's new values) in O(n) time, using comparison-based sorting.

4. [**Sorting, 15 points**]

   You are given a sorted sequence of distinct integers $\{a_1, a_2, a_3, \ldots, a_n\}$. Give an $O(\log n)$ time algorithm to determine whether or not there exists an index i such that $a_i = i$ (e.g. $\{-2, 2, 3, 6\}$ contains such an index: $a_3 = 3$).

5. [**Sorting in Linear Time, 10 points**] For each of the below scenarios, choose the better sorting solution between counting sort and radix sort. Justify your answer.

   (a) You have a list of 100 000 distinct credit card numbers (16 decimal digits each).

   (b) You have a list of 4 billion distinct IPv4 address (32 binary digits each).