CS 270 (Fall 2025) — Assignment 2

Due: Friday, September 19, by 11:59pm

General Instructions. The following assignment is meant to be challenging, and we anticipate that it will take most of you at least 10–15 hours to complete. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-$O/\Omega/\Theta$) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc.) take constant time.

Submission. Homework submission will be through the Gradescope system. Instructions and links have been provided through the course website and Piazza. The only accepted format is PDF. For this homework, we will still allow photographs/scans of handwritten solutions, but if they are illegible, you may lose points. Starting with Homework 3, only typed solutions will be accepted, and we highly recommend producing your solutions using LaTeX (the text markup language we are also using for this assignment).

(1) Given an unsorted array $A := \{a_1, \ldots, a_n\}$, we may assume that all elements are distinct. The median value of this array is the number a_i such that exactly half of the numbers in A are smaller than a_i and half are larger than a_i.

Now, assume there is a subroutine ApproxMedian(B) that returns a value p that is close to the median of B for any array B. Concretely, at least one-third of the numbers in B are smaller than p, and at least one-third of the numbers in B are larger than p. You do not need to worry about the implementation of this subroutine, and each call to ApproxMedian() takes constant time.

Given this ApproxMedian subroutine as a black box, can you design an efficient algorithm to find the median? Please provide an $O(n)$-time algorithm with pseudocode, a running-time analysis, and a correctness proof.

(2) Following the problem we discussed in our lecture: Given n points in a two-dimensional space, assume that all these points are located on three parallel horizontal lines. Can you design an $O(n \log n)$-time algorithm to find the closest pair among these points? Please provide pseudocode, a running-time analysis, and a correctness proof.

(3) Given two large n-bit integers a and b, can you design an algorithm to multiply a and b? We need an algorithm that is faster than $O(n^{1.9})$.