Problem Set 2

Reading: Chapters 5-9, excluding 5.4 and 8.4

Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered in the exercises.

Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date and the names of any students with whom you collaborated.

You will often be called upon to “give an algorithm” to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm.

Remember, your goal is to communicate. Full credit will be given only to correct algorithms which are *which are described clearly*. Convoluted and obtuse descriptions will receive low marks.

Exercise 2-1. Do Exercise 5.2-4 on page 98 in CLRS.

Exercise 2-2. Do Exercise 8.2-3 on page 170 in CLRS.

Problem 2-1. Randomized Evaluation of Polynomials

In this problem, we consider testing the equivalence of two polynomials in a finite field.

A *field* is a set of elements for which there are addition and multiplication operations that satisfy commutativity, associativity, and distributivity. Each element in a field must have an additive and multiplicative identity, as well as an additive and multiplicative inverse. Examples of fields include the real and rational numbers.

A *finite field* has a finite number of elements. In this problem, we consider the field of integers modulo p. That is, we consider two integers a and b to be “equal” if and only if they have the same remainder when divided by p, in which case we write $a \equiv b \bmod p$. This field, which we denote as $\mathbf{Z}/p$, has p elements, $\{0, \ldots, p-1\}$.

Consider a polynomial in the field $\mathbf{Z}/p$:

$$a(x) = \sum_{i=0}^{n} a_i x^i \bmod p \tag{1}$$

A *root* or *zero* of a polynomial is a value of x for which $a(x) = 0$. The following theorem describes the number of zeros for a polynomial of degree n.

Theorem 1 *A polynomial $a(x)$ of degree n has at most n distinct zeros.*

Polly the Parrot is a very smart bird that likes to play math games. Today, Polly is thinking of a polynomial $a(x)$ over the field $\mathbf{Z}/p$. Though Polly will not tell you the coefficients of $a(x)$, she will happily evaluate $a(x)$ for any x of your choosing. She challenges you to figure out whether or not a is equivalent to zero (that is, whether $\forall x \in \{0, \dots, p-1\} : \ a(x) \equiv 0 \bmod p$).

Throughout this problem, assume that $a(x)$ has degree n, where $n < p$.

(a) Using a randomized query, describe how much information you can obtain after a single interaction with Polly. That is, if a is *not* equivalent to zero, then for a query x chosen uniformly at random from $\{0, \dots, p-1\}$, what is the probability that $a(x) = 0$? What if a *is* equivalent to zero?

(b) If $n = 10$ and $p = 101$, how many interactions with Polly do you need to be 99% certain whether or not a is equivalent to zero?

Later, you are given three polynomials: $a(x)$, $b(x)$, and $c(x)$. The degree of $a(x)$ is n, while $b(x)$ and $c(x)$ are of degree $n/2$. You are interested in determining whether or not $a(x)$ is equivalent to $b(x) * c(x)$; that is, whether $\forall x \in \{0, \dots, p-1\} : \ a(x) \equiv b(x) * c(x) \bmod p$.

Professor Potemkin recalls that in Problem Set 1, you showed how to multiply polynomials in $\Theta(n^{\lg_2(3)})$ time. Potemkin suggests using this procedure to directly compare the polynomials. However, recalling your fun times with Polly, you convince Potemkin that there might be an even more efficient procedure, if some margin of error is tolerated.

(c) Give a randomized algorithm that decides with probability $1 - \epsilon$ whether or not $a(x)$ is equivalent to $b(x) * c(x)$. Analyze its running time and compare to Potemkin's proposal.

Problem 2-2. Distributed Median

Alice has an array A[1..n], and Bob has an array B[1..n]. All elements in A and B are distinct. Alice and Bob are interested in finding the median element of their combined arrays. That is, they want to determine which element m satisfies the following property:

$$|\{i \in [1, n] : A[i] \leq m\}| + |\{i \in [1, n] : B[i] \leq m\}| = n \tag{2}$$

This equation says that there are a total of n elements in both A and B that are less than or equal to m. Note that m might be drawn from either A or B.

Because Alice and Bob live in different cities, they have limited communication bandwidth. They can send each other one integer at a time, where the value either falls within $\{0, \ldots, n\}$ or is drawn from the original A or B arrays. Each numeric transmission counts as a "communication" between Alice and Bob. One goal of this problem is to minimize the number of communications needed to compute the median.

- **(a)** Give a deterministic algorithm for computing the combined median of A and B. Your algorithm should run in $O(n \log n)$ time and use $O(\log n)$ communications. (Hint: consider sorting.)
- **(b)** Give a randomized algorithm for computing the combined median of A and B. Your algorithm should run in expected $O(n)$ time and use expected $O(\log n)$ communications. (Hint: consider RANDOMIZED-SELECT.)

Problem 2-3. American Gladiator

You are consulting for a game show in which n contestants are pitted against n gladiators in order to see which contestants are the best. The game show aims to rank the contestants in order of strength; this is done via a series of 1-on-1 matches between contestants and gladiators. If the contestant is stronger than the gladiator, then the contestant wins the match; otherwise, the gladiator wins the match. If the contestant and gladiator have equal strength, then they are "perfect equals" and a tie is declared. We assume that each contestant is the perfect equal of exactly one gladiator, and each gladiator is the perfect equal of exactly one contestant. However, as the gladiators sometimes change from one show to another, we do *not* know the ordering of strength among the gladiators.

The game show currently uses a round-robin format in which $\Theta(n^2)$ matches are played and contestants are ranked according to their number of victories. Since few contestants can happily endure $\Theta(n)$ gladiator confrontations, you are called in to optimize the procedure.

- **(a)** Give a randomized algorithm for ranking the contestants. Using your algorithm, the expected number of matches should be $O(n \log n)$.
- **(b)** Prove that any algorithm that solves part (a) must use $\Omega(n \log n)$ matches in the *worst case*. That is, you need to show a lower bound for any deterministic algorithm solving this problem.