Problem Set 8

This problem set is due **in recitation** on **Friday, December 5th.**

Reading: Chapters §30.1-30.2, §31.1-31.8.

There are **4** problems. Each problem is to be done on a separate sheet (or sheets) of three-hole punched paper. 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.

Problem 8-1. Modular Operations

For this problem, do not assume that arithmetic operations have $O(1)$ cost. You may assume that the operations $(a \cdot b)$ and $(a \bmod b)$ may be done in time $O(\log a \log b)$. On inputs of length n, we define an algorithm to be *polynomial-time* if it runs in $O(n^k)$ time for some constant k.

(a) Given a, b, c and prime p, give a polynomial-time algorithm that computes $a^{b^c} \bmod p$. Note that the size of the input is $\log a + \log b + \log c + \log p$.

(b) Given two integers, a and b, give a polynomial-time algorithm to find the closest integer to $\sqrt[b]{a}$.

(c) Given an integer x, give a polynomial-time algorithm to determine if x is a power, i.e. if there exists integers, $a, b \neq 1$ such that x can be written as $x = a^b$. **Hint:** Use part **(b)**.

Problem 8-2. Chinese Remainder Theorem

Given integers $p, q, n = pq$, where p and q are prime, there exists a 1-1 and onto mapping between $\mathbb{Z}_n^*$ and $(\mathbb{Z}_p^*, \mathbb{Z}_q^*)$, which is quite useful. Let's explore it. The mapping is $f(x) = (x \bmod p, x \bmod q)$. You may assume that the operations $(a \cdot b)$ and $(a \bmod b)$ may be done in time $O(\log a \log b)$.

(a) Give an algorithm to compute x given $f(x), p, q$.

(b) Now, let us define the following multiplication operator: $(r, s) \odot (t, u) = (rt \bmod p, su \bmod q)$. Show that it is closed under $(\mathbb{Z}_p^*, \mathbb{Z}_q^*)$.

(c) Ben Bitdiddle designs a new multiplication unit, called the B-Diddy, that multplies two numbers $x, y \in \mathbb{Z}_n^*$ using the following algorithm:
1. Map x and y into pairs $(r, s), (t, u) \in (\mathbb{Z}_p^*, \mathbb{Z}_q^*)$.
2. Compute $(r, s) \odot (t, u) = (v, w)$.
3. Map (v, w) back into $z \in \mathbb{Z}_n^*$.

Analyze the B-Diddy's runtime and compare it to standard multiplication over $\mathbb{Z}_n^*$. Which is better to use?

(d) Suppose you are given inputs p, q, n and $(x^3 \bmod p, x^3 \bmod q)$. Give an algorithm to compute $x \bmod n$. You may assume that $gcd(3, \phi(n)) = 1$.

Problem 8-3. Snowball Throwing

Several 6.046 students hold a team snowball throwing contest. Each student throws a snowball with a distance in the range from 0 to $10n$. Let M be the set of distances thrown by males and F be the set of distances thrown by females. You may assume that the distance thrown by each student is unique and is an integer. Define a team score to be the combination of one male and one female throw.

Give an $O(n \log n)$ algorithm to determine every possible team score, as well as how many teams could achieve a particular score. This multi-set of values is called a *cartesian sum* and is defined as:

$$C = \{m + f : m \in M \text{ and } f \in F\}$$

Problem 8-4. Comparing Polynomials

Two single variable degree-d polynomials, P and Q, with coefficients from $\mathbb{Z}_p$ are said to be identical if $P(x) = Q(x)$ for all $x \in \mathbb{Z}_p$. Suppose you want to determine if two degree-d polynomials, F, G with coefficients in $\mathbb{Z}_p$ are identical, where $p > 2d$. However, you are not explicitly given F or G. Rather, you are given two black boxes, which on any input $x \in \mathbb{Z}_p$ return $F(x)$ and $G(x)$, respectively.

Give an efficient Monte-Carlo algorithm to determine if F and G are identical. If $F = G$, your algorithm should output the correct answer with probability 1. If $F \neq G$, your algorithm should output the correct answer with probability at least $3/4$.

You may use the following fact: A degree-d non-zero polynomial has at most d values of x for which it evaluates to 0.