Charles Harrison (csharris)
CS 157 — Spring 2013 **Homework Study Guide** March 16, 2013

Modular Arithmetic

Modular Arithmetic is a system for dealing with restricted ranges of Integers. $x \mod n$ is the remainder when x is divided by n. So if

$$x = qn + r$$

then

$$x \mod n = r$$

Two numbers are said to be *conguent modulo n* if they differ by a multiple of n, that is

$$x \equiv y \mod n \iff n \text{ divides } (x - y)$$

This definition defines a set of n *equivalence classes*, where each class has the form $i + kn$ for $k \in \mathbb{Z}$. For example, there are three equivalence classes modulo 3:

. . . -9 -6 -3 0 3 6 9 . . .
. . . -8 -5 -2 1 4 7 10 . . .
. . . -7 -4 -1 2 5 8 11 . . .

Where two elements in any one class are equivalent modulo 3.

Substitution Rule: if $x \equiv x' \mod n$ and $y \equiv y' \mod n$, then

$$x + y \equiv x' + y' \mod n \text{ and } xy \equiv x'y' \mod n$$

Identities:

- Associativity: $x + (y + z) \equiv (x + y) + z \mod n$
- Communitivity: $xy \equiv yx \mod n$
- Distributivity: $x(y + z) \equiv xy + xz \mod n$

Modular exponentiation is a technique for taking large exponents $x^y \mod n$ quickly. It involves doing intermediate computations modulo n.

- Naive solution: Perform the operation in y steps by taking

$$first = x \mod n$$

$$first * (x \mod n) = x^2 \mod n$$

etc. This method involves taking $O(y)$ multiplications, and if y is z bits long, we take $O(2^z)$ multiplications. This is pretty bad.

- Better solution using divide and conquer. Start with x and square repeatedly modulo n

$$x = x \mod n$$
$$x^2 = x * x \mod n$$
$$x^4 = x^2 * x^2 \mod n$$

etc. We require $\log_2 y$ multiplications to generate $x^y \mod n$. See `modular_exp.py` for an implementation.

Modular division In arithmetic in $\mathbb{R}$, every number $a \neq 0$ has an inverse, $\frac{1}{a}$, and $\frac{n}{a} = na^{-1}$. We can do a similar thing with modular arithmetic.

x is the multiplicative inverse of a modulo n if $ax \equiv 1 \mod n$, denoted a^{-1}.

Modular division theorem: For any $a \mod n$, a has a multiplicitive inverse modulo $n \iff$ it is relatively prime to n. When this inverse exists, it can be found in $O(n^3)$ time by running the extended Euclid's algorithm.