CISC 5835—Algorithms for Data Analytics
Fall, 2019

Chapter 1 Homework

1. [10 points] Compute the multiplicative inverse for each of the following, or explain why it doesn't exist. If the multiplicative inverse mod$n$ exists, make sure to express it as a number in the range $\{0 \ldots, n-1\}$.

(a) 28 mod 97.

(b) 28 mod 35.

2. [20 points] Recall that $x \equiv y \bmod N$ means N divides $x - y$ without remainder.

(a) Show that

$$x \equiv y \bmod N \implies x^a \equiv y^a \bmod N \qquad \text{for any non-negative integer } a. \tag{1}$$

Hint: You can prove this by induction on a.
For a non-inductive proof, it suffices to show that $(x^a - y^a)/(x - y) \in \mathbb{Z}$ (why?). Express this fraction as a polynomial in x and y. If you're stuck, try some small values of a (such as 2, 3, maybe 4) until you see the general pattern.

(b) Show that the converse of this result is false, by giving a counterexample. That is, find non-negative integers x, y, a, and N such that $x^a \equiv y^a \bmod N$, but $x \not\equiv y \bmod N$.

3. [10 points] What is $5^{2^{1000000}} \bmod 24$?
Hint: Don't try to calculate $5^{2^{1000000}}$, since the exponent $2^{1000000}$ is over 300,000 digits long!

4. [10 points] What is $4^{1536} \bmod 35$?
Hint: Don't try to calculate 4^{1536} directly, since it has over 924 digits. Use the following extension[1] of Fermat's Little Theorem: If p and q are distinct prime numbers and the integer a is not a multiple of pq, then

$$a^{(p-1)(q-1)} \equiv 1 \bmod pq.$$

[1]This extension of Fermat's Little Theorem is a special case of Euler's Theorem, which you may look up. This should not be confused with Euler's identity ($e^{i\pi} + 1 = 0$) or Euler's formula in graph theory or the zillions of other things Euler did. No wonder they called him "the prince of mathematicians".

5. [10 points] The fast modular exponentiation algorithm on page 19 also works for fast (non-modular) exponentiation; simply leave off the mod$N$ part. However, note that x^y will do multiplications by 1, which are useless; for example, it computes x^1 as

$x^1 = x \cdot x^0$	make recursive call
$= x \cdot 1$	return value from recursive call
$= x$	

We can get rid of these useless multiplications by adding the statement

if $y = 1$: return x

before computing z in line 3 in the original version of the algorithm.

(a) [5 points] How many multiplications does this revised fast exponentiation algorithm use to compute a^{15}?

(b) [5 points] Find a method to calculate a^{15} using fewer multiplications than in part (a).