Number Theory: Mods and Primes

Aditya Arjun

CS 104C

Fall 2020

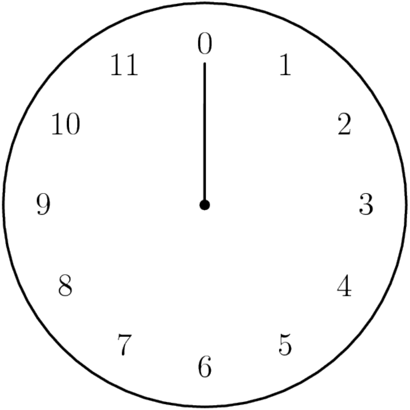
Modular Arithmetic

- As you've seen in past weeks' homeworks, many problems ask you to output an answer "modulo $10^9 + 7$"
- 1000! has 2568 decimal digits
- We use this number because
  - It is prime,
  - It fits into an integer,
- How do we work with "modulo"?

Division Algorithm.

- For every pair of integers a, b, there exist unique integers q, r such that $a = qb + r$ and $0 \leq r < b$.
  As an example, if $a = 11, b = 5$, $a = 2b + 1$. The quotient is $q = 2$, and the remainder is $r = 1$.
- In most languages, $a/b = q$, $a\%b = r$.
- Be careful with negative values!

Modular Arithmetic



- Clocks work "modulo 12".
- What is 3 hours after 11? 2.
- Then we say $11 + 3 \equiv 2 \pmod{12}$.

Modular rules

Which of the following are correct?

- $(a+b)\%m = ((a\%m)+(b\%m))\%m$?
- $(q_1 m + r_1 + q_2 m + r_2)\%m = ((q_1+q_2)m + (r_1+r_2))\%m = r_1 + r_2\%m$
- $(a-b)\%m = ((a\%m)-(b\%m))\%m$?
- $(a \cdot b)\%m = ((a\%m)\cdot(b\%m))\%m$?
- $(a/b)\%m = ((a\%m)/(b\%m))\%m$?
- $a^b\%m = (a\%m)^b\%m = (a\%m)^{(b\%m)}\%m$?

Modular rules

Which of the following are correct?

- $(a + b)\%m = ((a\%m) + (b\%m))\%m$?
- $(a - b)\%m = ((a\%m) - (b\%m))\%m$?
- $(a \cdot b)\%m = ((a\%m) \cdot (b\%m))\%m$?
- $(a/b)\%m = ((a\%m)/(b\%m))\%m$?
- $a^{b}\%m = (a\%m)^{b}\% = (a\%m)^{(b\%m)}\%m$?

Answers: First 3 are true, fourth makes little sense, first part of the last one is true.

Again, be careful with negatives and mod. We usually implement subtraction as (a % m - b % m + m) % m.

Definitions

- d is a **divisor** of n if d divides n evenly.
- Equivalent: $n \% d = 0$
- A number is prime if it has exactly two positive divisors (1 and itself).

```
boolean isPrime(int n) {
    for (int d = 2; d < n; ++d)
        if (n % d == 0)
            return false;
    return true;
}
```

- Time complexity?

Definitions

- d is a **divisor** of n if d divides n evenly.
- Equivalent: $n\%d = 0$
- A number is prime if it has exactly two positive divisors (1 and itself).

```
boolean isPrime(int n) {
    for (int d = 2; d < n; ++d)
        if (n % d == 0)
            return false;
    return true;
}
```

- Time complexity? $O(N)$
- Can we do better?

Primes

- **Problem:** Given a number N, determine whether it is prime
- **Solution:** Check all possible divisors **up to** $\sqrt{N}$

```
boolean isPrime(int n) {
    for (int d = 2; d * d <= n; ++d)
        if (n % d == 0)
            return false;
    return true;
}
```

- Time complexity? $O(\sqrt{N})$

Primes

- **Problem:** Given a number N, get all primes up to N
- **Solution:** Assume every number is prime. Eliminate all the multiples of 2, then all the multiples of 3, ...

```
List<Integer> getPrimes(int n) {
    boolean composite = new boolean[n + 1];
    List<Integer> primes = new ArrayList<Integer>();
    for (int i = 2; i <= n; ++i) {
        if (composite[i]) continue;
        primes.add(i);
        for (int j = 2 * i; j <= n; j += i)
            composite[j] = true;
    }
    return primes;
}
```

- Time complexity? $O(N \log \log N)$

GCDs

- The greatest common divisor of two numbers a and b is the largest integer g such that both a and b are multiples of g.
- Complex Approach: factor both numbers
- Euclidean algorithm: Let $a = qb + r$. Note that if some value g divides both a and b, then g divides r.
  Proof: $a = qb + r \implies r = a - qb$. Since g divides both a and b, g divides r.

```
int gcd(int a, int b) {
    if (b == 0) {
        return a;
    }
    return gcd(b, a % b);
}
```

- Time complexity? $O(\log N)$

Modular rules, again

- **Problem:** Given N people, calculate how many ways you can make a committee of M people.

Modular rules, again

- **Problem:** Given N people, calculate how many ways you can make a committee of M people.
- $\binom{N}{M}$
- **Output your answer modulo $\mathbf{10^9 + 7}$.**
- How do we compute the answer to this?

Modular rules, again

- Recall that we can't divide two numbers modulo a third
- **Problem:** Divide two numbers modulo a third (say, for binomial coefficients)
- Let's say the third number is prime
- If we want to divide a by b modulo p, we'll take $a \cdot b^{-1} \pmod{p}$, where b^{-1} is a "multiplicative inverse" of b
- Formally, b^{-1} is a multiplicative inverse of b if $b \cdot b^{-1} \equiv 1 \pmod{p}$
- We can use Fermat's little theorem: $b^{p-1} \equiv 1 \pmod{p}$
- Then $b^{-1} = b^{p-2}$
- **Problem:** How can we compute this?
- **Answer:** Exponentiation by squaring

Modular rules, again

- Recall that we can't divide two numbers modulo a third
- **Problem:** Divide two numbers modulo a third (say, for binomial coefficients)
- Let's say the third number is prime
- If we want to divide a by b modulo p, we'll take $a \cdot b^{-1} \pmod{p}$, where b^{-1} is a "multiplicative inverse" of b
- Formally, b^{-1} is a multiplicative inverse of b if $b \cdot b^{-1} \equiv 1 \pmod{p}$
- We can use Fermat's little theorem: $b^{p-1} \equiv 1 \pmod{p}$
- Then $b^{-1} \equiv b^{p-2}$
- **Problem:** How can we compute this?
- **Answer:** Exponentiation by squaring $a^p = (a^2)^{p/2}$

Exponentiation by Squaring

- If $2|p$, $a^p = (a^2)^{p/2}$
- Otherwise, $2|(p-1)$, so we can write $a^p = a(a^2)^{(p-1)/2}$
- Say we can compute a^p in $T(p)$ operations
- $T(p) = 1 + T(p/2)$
- Gives us a complexity of $O(\log(p))$
- Intuitively, we can view this as using the binary representation of the power to compute the result more efficiently.
- For example, $11_{10} = 1011_2 = 8 + 2 + 1$.
- This gives us that $a^{11} = a^8 \cdot a^2 \cdot a^1$