Competitive Programming Club

Meeting 5 - Number Theory

Introduction

Lecture:

1. Sieve of Eratosthenes

2. Properties of Modular Arithmetic

3. Finding Modular Inverses Using Euler's Theorem

4. Exponentiation by Squaring

Practice Problems and Guidance:

Practice Problems Link: https://vjudge.net/contest/627246 (5 problems, all from ICPC regional contests). I will resolve any questions and debug your code.

Brute Force Version of the Sieve of Eratosthenes

Prime Numbers: Numbers with exactly two factors, like 2, 3, 5, and 7.

Composite Numbers: Numbers with more than two factors, like 4, 6, 8, and 9.

Goal: Identify all prime numbers in the range [1, n].

How It Works: Starting from 2 up to n, make all multiples (at least twice) of each number as composite.

Result: Numbers that remain unmarked are primes.

Time Complexity: O(n log n).

Note: While not the fastest method available, it is the simplest to implement and widely used in contests due to its practical efficiency.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Prime Numbers

2 3 5 7

Sieve (Pseudocode)

Case 1: n <= 10^7

```
marked[1] = True
for i from 2 to n:
    for j from 2 * i to n step i:
        marked[i] = True
```

Optimization Tip: Store the marked array using a bitset for faster performance.

Sieve (Pseudocode)

Case 2: 10^7 < n <= 10^8

Time Complexity: O(n log log n)

```
marked[1] = True
for i from 2 to sqrt(n):
    if not marked[i]:
        for j from 2 * i to n step i:
            marked[i] = True
```

Case 3: n > 10^8

Use Euler’s Sieve (Linear Sieve).

Definition of Modular Arithmetic

- Definition: Given an integer a and modulus m, the result of modular arithmetic is $a \mod m$, which is the remainder after dividing a by m.
- Formula: $a \equiv b \pmod{m}$ means that a and b are congruent under modulus m.
- **Example:** The result of $17 \mod 5$ is 2 because the remainder of $17 \div 5$ is 2. Thus, $17 \equiv 2 \pmod{5}$.

Addition Property

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$.

Subtraction Property

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a - c \equiv b - d \pmod{m}$.

Multiplication Property

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a \cdot c \equiv b \cdot d \pmod{m}$.

Division Property (Doesn't Hold)

- Division generally does not hold in modular arithmetic with modulus m.
- Reason: Direct division by a number usually results in a non-integer.
- **Example:** For instance, $17 \div 4 \mod 5$.
- Modular division can be achieved through modular inverses (which will be discussed in the following section).

Definition of Modular Inverse

- **Modular Inverse**: For an integer a, the modular inverse under modulus m is an integer x such that:

$$a \cdot x \equiv 1 \pmod{m}$$

- This equation implies that the product of a and x equals 1 under modulus m.
- Once the modular inverse x is found, division in modular arithmetic can be performed as $\frac{1}{a} \mod m$.

Euler's Theorem and the Concept of Coprimeness

- **Coprimeness**: Two numbers a and m are coprime if their greatest common divisor (GCD) is 1.
- **Euler's Theorem**: If a is coprime with the modulus m, then:

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

- Here, $\phi(m)$ is the Euler function, representing the count of integers less than or equal to m that are coprime with m.
- If m is a prime number, then $\phi(m) = m - 1$.

Using Euler's Theorem to Compute Modular Inverse

- According to Euler's Theorem:

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

- To find the modular inverse of a, we can rewrite the equation as:

$$a^{-1} \equiv a^{\phi(m)-1} \pmod{m}$$

- This means that the modular inverse a^{-1} can be computed using exponentiation.

Exponentiation by Squaring

When calculating $a^b \mod m$, multiplying a sequentially requires $O(b)$ multiplications, which is inefficient. The Exponentiation by Squaring algorithm reduces the computation to $O(\log b)$ multiplications.

Algorithm Outline

1. **Decompose the Exponent by Bits**: Express the exponent b in binary. For instance, $b = 11$ in binary is '1011', which can be split into $8 + 2 + 1$, or $b = 2^3 + 2^1 + 2^0$.

2. **Precompute Powers**: Compute the powers $a, a^2, a^{2^2}, \ldots, a^{2^k} \mod m$, squaring the result each time.

3. **Selective Multiplication**: For each binary bit, if the bit is '1', multiply the corresponding power into the final result.

Example

To compute $7^{11} \mod 26$:

1. Convert 11 to binary: '1011'.
2. Compute the required powers: $7^1, 7^2, 7^4, 7^8 \mod 26$.
   - $7^1 \equiv 7 \mod 26$
   - $7^2 \equiv 23 \mod 26$
   - $7^4 \equiv 9 \mod 26$
   - $7^8 \equiv 3 \mod 26$
3. Use the binary bits:
   - Bit 1: '1', so the result is $1 \times 7 \equiv 7 \mod 26$.
   - Bit 2: '1', so the result is $7 \times 23 \equiv 5 \mod 26$.
   - Bit 3: '0', skip.
   - Bit 4: '1', so the result is $5 \times 3 \equiv 15 \mod 26$.

The final result is: $7^{11} \mod 26 = 15$.

Pseudocode Implementation

```
function modExp(a, b, m):
    result = 1
    while b > 0:
        if b % 2 == 1:
            result = result * a % m
        a = a * a % m
        b = b // 2
    return result
```

Summary and Practice Problems

Recap: This lecture covered the Sieve of Eratosthenes, properties of modular arithmetic, finding modular inverses using Euler's theorem, and exponentiation by squaring.

Practice Problems: Please practice the following problems to reinforce your understanding of number theory: https://vjudge.net/contest/627246.

If you have any questions about the lecture or need help with hints or debugging your code, feel free to ask me. I hope this meeting helps you deepen your understanding of number theory in competitive programming.