Lecture 2
Math, Number Theory and Counting

Abdulla Nur Faisal

Undergraduate Student
Computer Science and Engineering, Eastern University
Dhaka, Bangladesh

Abstract. This lecture is a part of competitive programming training lectures prepared for Eastern University, Dhaka. The lecture introduces some Math, Number Theory and Counting based concepts : GCD, LCM, Prime Numbers, Sieve of Eratosthenes, Modular Arithmetic, Binary Exponentiation, Modular Multiplicative Inverse, Counting Principles.

1 Discussion of problems from Long Contest - 1

2 Greatest Common Divisor (GCD), Least Common Multiple (LCM)

– Euclid's Algorithm

$$\gcd(a, b) = \begin{cases} a, & \text{if } b = 0 \\ \gcd(b, a \bmod b), & \text{otherwise.} \end{cases}$$

– Use the `__gcc` function in C++

3 Dealing with Prime Numbers

– Primality Check

```
//Complexity: O(sqrt(N))
bool isPrime(int n) {
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

Listing 1.1. Primality Check

– Fundamental theorem of arithmetic

$$n = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_k^{e_k}$$

– Prime Factorization

```
//Complexity: O(sqrt(N))
vector<int> factorize(int n) {
    vector<int> primeFactors;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            while (n % i == 0) {
                primeFactors.push_back(i);
                n /= i;
            }
        }
    }
}
```

Listing 1.2. Prime Factorization

– Prime Generation (Sieve of Eratosthenes)

```
//Complexity: O(Nlog(log(N))
const int maxn = 1e5;
bool mark[maxn];
for (int i = 2; i < maxn; i++) {
    if (mark[i] == false) {
        for (int j = 2 * i; j < maxn; j += i) {
            mark[j] = true;
        }
    }
}
```

Listing 1.3. Sieve of Eratosthenes

4 Divisors, Sigma Function

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \times \frac{p_2^{e_2+1} - 1}{p_2 - 1} \times \cdots \times \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

5 Co-prime numbers, Totient / Phi Function

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \cdots \times \frac{p_k - 1}{p_k}$$

6 Modular Arithmetic

7 Binary Exponentiation

$$a^n = \begin{cases} 1 & \text{if } n == 0 \\ \left(a^{\frac{n}{2}}\right)^2 & \text{if } n > 0 \text{ and } n \text{ even} \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a & \text{if } n > 0 \text{ and } n \text{ odd} \end{cases}$$

```
//Complexity: O(logN)
int bigMod(int a, int b, int MOD) {
    if (b == 0) return 1LL;
    if (b == 1) return a;
    int res = bigMod(a, b / 2, MOD);
    res = (res * res) % MOD;
    if (a % 2 != 0) res = (res * a) % MOD;
    return res;
}
```

Listing 1.4. Binary Exponentiation

8 Counting

- Addition Rule
- Multiplication Rule
- Permutations

$$^{n}P_k = \frac{n!}{(n-k)!}$$

- Combinations

$$^{n}C_k = \frac{n!}{k!(n-k)!}$$

9 Modular Multiplicative Inverse

$$a \cdot x \equiv 1 \mod m.$$

- Exists **only if**,

$$\gcd(a, m) = 1$$

- From Euler's theorem,

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

- If m is a prime number, then

$$a^{m-1} \equiv 1 \mod m \tag{2}$$

- Multiply both sides by a^{-1}

$$a^{m-2} \equiv a^{-1} \mod m \tag{3}$$

10 Long contest - 2