Advanced Number Theory 2

- Srivaths P

Goal

To learn:

- Binary/Modular Exponentiation
- Euclidean algorithm for GCD
- Fast binomial coefficient

Binary Exponentiation

The idea of binary exponentiation is as follows:

When B is even: $A^B = A^{\frac{B}{2}} \times A^{\frac{B}{2}}$.

When B is odd: $A^B = A^{\frac{B}{2}} \times A^{\frac{B}{2}} \times A$.

(Assuming division is floored)

We can do the above using a recursive function (or iteratively).

Factorial precomputation

You can precompute the necessary factorials (and inverse factorials) to compute binomial coefficient in O(log N) using Binary Exponentiation.

Greatest Common Divisor

$GCD(A, B)$ is the Greatest Common Divisor of A and B.
$LCM(A, B)$ is the Least Common Multiple of A and B.

To calculate GCD efficiently, we can use the Euclidean Algorithm.

Euclidean Algorithm states that $GCD(A, B) = GCD(B \ \% \ A, A)$.
When $A = 0$, the solution is B.

Euclidean Algorithm – Code

Recursive:

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

Iterative:

```
int gcd_(int a, int b) {
    while (a) {
        int t = a;
        a = b % a;
        b = t;
    }

    return b;
}
```