Number Theory

1 Greatest Common Divisors

2 Congruence

2.1 Modular Inverse

A modular inverse of an integer b (modulo p) is the integer b^{-1} such that:

$$b\,b^{-1} \equiv 1 \; mod \; p$$

2.1.1 Extended Euclidean Method

if $b\,b^{-1} \equiv 1 \; mod \; p$, then we have:

$$b\,b^{-1} = py + 1$$
$$b\,b^{-1} - py = 1$$

If we can solve the indeterminate equation $bx + py = 1$, then we can get the value of b^{-1}, which is x. To get the x value, we can use extended euclidean method.

```
def extended_gcd(a, b):
    if b == 0:
        return (1, 0)
    x, y = extended_gcd(b, a % b)
    return (y, x - a // b * y)

def get_modular_inverse(x, p):
    res, _ = extended_gcd(x, p)
    while res < 0 {
        res += p
    }
    return res
```

2.2 Lucas Theorem

For non-negative integers m and n and a prime p, the following congruence relation holds:

$$\binom{m}{n} \equiv \prod_{i=0}^{k} \binom{m_i}{n_i} \; mod \; p$$

Where

$$m = m_k p^k + m_{k-1} p^{k-1} + ... + m_1 p + m_0$$
$$n = n_k p^k + n_{k-1} p^{k-1} + ... + n_1 p + n_0$$

Another form:

$$\binom{m}{n} \equiv \binom{\lfloor m/p \rfloor}{\lfloor n/p \rfloor} \binom{m \ mod \ p}{n \ mod \ p} \ mod \ p$$

2.3 Common Equation and Conclusions

if $a_1 \equiv b_1 \ mod \ m$, $a_2 \equiv b_2 \ mod \ m$ then:

$$a_1 + a_2 \equiv b_1 + b_2 \ mod \ m$$
$$a_1 - a_2 \equiv b_1 - b_2 \ mod \ m$$
$$a_1 \times a_2 \equiv b_1 \times b_2 \ mod \ m$$

if $a \equiv b \ mod \ m$, $k > 0$, and d is a common divisor of a, b and m, then:

$$ak \equiv bk \ mod \ m$$
$$\frac{a}{d} \equiv \frac{b}{d} \ mod \ \frac{m}{d}$$

3 Numeral System

3.1 Number of bits for a number N with base b

Assume there are k bits, then:

$$b^{k-1} \leq N < b^k$$
$$log_b^N < k \leq log_b^N + 1$$

If such integer k does not exist, then we must have an integer t:

$$\begin{cases} t \leq log_b^N \\ log_b^N + 1 < t + 1 \end{cases}$$

Under this condition, we have:

$$\begin{cases} N >= b^t \\ N < b^t \end{cases}$$

which is impossible. So we can conclude that k must exist, and the value of k is $\lfloor log_b^N + 1 \rfloor$ or $\lceil log_b^{N+1} \rceil$. The latter one is calculated by:

$$N \leq b^k - 1$$
$$k \geq log_b^{N+1}$$

To represent N numbers from 0 to N - 1, we need $\lfloor log_b^{N-1} + 1 \rfloor$ or $\lceil log_b^N \rceil$ bits.
Similarly, for a binary tree of N nodes, the min height (if a single node has height 1) is $\lfloor log_b^N + 1 \rfloor$ or $\lceil log_b^{N+1} \rceil$.

4 Others

4.1 Gray Code

4.1.1 Transform

For a n bits number, the index of digits from right to left is 0 to n - 1. The number can be represented as $B_{n-1}...B_1B_0$.

$$\begin{cases} G_{n-1} = B_{n-1} \\ G_i = B_i \oplus B_{i+1}, 0 \le i \le n-2 \end{cases}$$

$$\begin{cases} B_{n-1} = G_{n-1} \\ B_i = G_i \oplus B_{i+1}, 0 \le i \le n-2 \end{cases}$$

4.1.2 Construction

Method 1:
If we have two digits gray code 00 01 11 10, to construct three digits gray code, we can make a mirror symmetry: 00 01 11 10 | 10 11 01 00.

Then for the first part we append the prefix 0, for the second part, we append 1. finally, we have:
000 001 011 010 | 110 111 101 100.

Method 2:
From the starting all-zero gray code, we can construct the next two numbers using following two steps:

(1). Flip the least significant digit to get the next gray code.
(2). Flip the left bit of the rightmost 1 to get the next gray code.

4.1.3 Formula

$$G(n) = n \oplus \lfloor \frac{n}{2} \rfloor$$