Math

ZUCC ACM Group

2023.01.13

目录

目录

快速幂

Definition

- 给定二元组 $(A, *)$，其中 A 是集合，$*$ 是满足结合律的运算，求 $a*a*...*a$，$a \in A$ (k 个 a，在不引起歧义的情况下一般写作 a^k)

- 给定二元组 $(A, *)$，其中 A 是集合，$*$ 是满足结合律的运算，求 $a*a*...*a$，$a \in A$ (k 个 a，在不引起歧义的情况下一般写作 a^k)
- 通过将 k 拆解成 $\sum x_i 2^i (x_i = 0\ or\ 1)$ 的形式，使得 $O(k)$ 次运算变成 $O(log_2 k)$ 次运算

快速幂
Definition

- 给定二元组 $(A,*)$，其中 A 是集合，$*$ 是满足结合律的运算，求 $a*a*...*a$，$a\in A$ (k 个 a，在不引起歧义的情况下一般写作 a^k)
- 通过将 k 拆解成 $\sum x_i 2^i (x_i = 0 \ or \ 1)$ 的形式，使得 $O(k)$ 次运算变成 $O(log_2 k)$ 次运算
- E.g: $a^{13} = (a^8)\times(a^4)\times(a^1)$，令 $b=a^2, c=a^4, d=a^8$，则可以先用 1 次操作求出 b，再用一次操作求出 c，再用一次操作求出 d，然后两次操作求出 $d\times c\times a = a^{13}$，一共 5 次

```cpp
ll qpow(ll a, ll b, ll m){ // a ^ b % m
    ll res = 1 % m;
    while(b){
        if(b & 1) res = res * a % m;
        b >>= 1;
        a = a * a % m;
    }
    return res;
}
```

$0 \leq a, b, p \leq 10^9$, 求 $a^b \bmod p$

快速幂

Problem - 1

$0 \le a, b, p \le 10^9$，求 $a^b \bmod p$

- 模板题，复杂度 $O(logb)$

$0 \le a, b, p \le 10^9$，求 $a^b \bmod p$

- 模板题，复杂度 $O(logb)$
- 要注意一些边界情况，如 $b = 0$ 且 $p = 1$

$0 \leq a, b, p \leq 10^{18}$, 求 $a^b \bmod p$

快速幂

Problem - 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \bmod p$

- 使用 $\_\_int128$ 可以直接通过

快速幂

Problem - 2

$0 \le a, b, p \le 10^{18}$，求 $a^b \bmod p$

- 使用 $\_\_int128$ 可以直接通过
- 若不使用，那么按照之前的做法计算过程中会溢出

$0 \le a, b, p \le 10^{18}$, 求 $a^b \bmod p$

- 使用 $\_\_int128$ 可以直接通过
- 若不使用，那么按照之前的做法计算过程中会溢出
- 考虑将 $a * a$ 的这一步改成加法

快速幂
Problem - 2

$0 \le a, b, p \le 10^{18}$，求 $a^b \bmod p$

- 使用 $\_\_int128$ 可以直接通过
- 若不使用，那么按照之前的做法计算过程中会溢出
- 考虑将 $a * a$ 的这一步改成加法
- 可以将式子中的一个 a 拆分成 $\sum x_i 2^i (x_i = 0\ or\ 1)$

快速幂

Problem - 2

$0 \le a, b, p \le 10^{18}$，求 $a^b \bmod p$

- 使用 $\_\_int128$ 可以直接通过
- 若不使用，那么按照之前的做法计算过程中会溢出
- 考虑将 $a * a$ 的这一步改成加法
- 可以将式子中的一个 a 拆分成 $\sum x_i 2^i (x_i = 0 \text{ or } 1)$
-
```cpp
ll mul(ll a, ll b, ll m){ // a * b % m
    ll res = 0, x = a;
    while(b){
        if(b & 1) res = (res + x) % m;
        b >>= 1;
        x = x * 2 % m;
    }
    return res;
}
```

快速幂

Problem - 3

$1 \leq n, p \leq 10^{18}$, 求斐波那契数列的第 n 项对 p 取余的值

$1 \leq n, p \leq 10^{18}$, 求斐波那契数列的第 n 项对 p 取余的值

- 考虑矩阵，因为矩阵也满足结合律

$1 \le n, p \le 10^{18}$，求斐波那契数列的第 n 项对 p 取余的值

- 考虑矩阵，因为矩阵也满足结合律
- $\begin{bmatrix} F_n & F_{n+1} \end{bmatrix} = \begin{bmatrix} F_{n-1} & F_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

快速幂

Problem - 3

$1 \le n, p \le 10^{18}$，求斐波那契数列的第 n 项对 p 取余的值

- 考虑矩阵，因为矩阵也满足结合律
- $\begin{bmatrix} F_n & F_{n+1} \end{bmatrix} = \begin{bmatrix} F_{n-1} & F_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
- 递推一下可以得到：$\begin{bmatrix} F_n & F_{n+1} \end{bmatrix} = \begin{bmatrix} F_0 & F_1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$，其中 $F_0 = 0$，$F_1 = 1$

$1 \le n, p \le 10^{18}$，求斐波那契数列的第 n 项对 p 取余的值

- 考虑矩阵，因为矩阵也满足结合律
- $\begin{bmatrix} F_n & F_{n+1} \end{bmatrix} = \begin{bmatrix} F_{n-1} & F_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
- 递推一下可以得到：$\begin{bmatrix} F_n & F_{n+1} \end{bmatrix} = \begin{bmatrix} F_0 & F_1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$，其中 $F_0 = 0$，$F_1 = 1$
- 然后就可以用快速幂去计算矩阵的 n 次方

目录

基础概念
Definition

模与剩余系

已知正整数 p, 则任何整数 n 都可以被唯一表示为 $n = ap + b$, 其中 $b \in [0, p)$, 定义 $n \bmod p = b$, 当 $n \bmod p = 0$ 时称 p 整除 n, 写作 $p|n$, p 称为 n 的因子. 若 $A = \{n \bmod p \mid n \in S\}$, 称 A 为 S 模 p 的剩余系, 当 $A = \{x \mid x \in [0, p)\}$ 时为完全剩余系.

素数 (质数) 与唯一分解定理

当且仅当正整数 p 只有 1 和 p 两个因子时被称为素数. 任意正整数都能被**唯一**表示为任意个素数的乘积, 一般写作 $n = \prod p_i^{k_i}$.

基础概念
Definition

gcd 与 lcm

$gcd(a,b)$ 表示 a 和 b 的最大公约数，即 $gcd(a,b)=\prod {p_i}^{min(k_{a_i},k_{b_i})}$
$lcm(a,b)$ 表示 a 和 b 的最小公倍数，即 $lcm(a,b)=\prod {p_i}^{max(k_{a_i},k_{b_i})}$
显然有 $a\times b=gcd(a,b)\times lcm(a,b)$
当 $gcd(a,b)=1$ 时，称 a,b 互质。

欧拉函数

用欧拉函数 $\varphi(n)$ 表示 $[1,n]$ 内与 n 互质的数的个数。
$\varphi(n)=n\prod(1-\frac{1}{p_i})$。证明可以点击链接。

基础概念
Definition

欧拉定理

若 $gcd(a,p)=1$，则有 $a^{\varphi(p)}=1 \bmod p$，当 p 为质数时称为费马小定理。

费马小定理

若 p 为质数，且 $gcd(a,p)=1$，则 $a^{p-1}\equiv 1 \bmod p$

逆元

a 的逆元写作 a^{-1} 或 $inv(a)$，逆元的作用在于实现模意义下的除法运算，用 $b\times a^{-1} \bmod p$ 来表示 $\frac{b}{a}$
当 $gcd(a,p)=1$ 且 p 为质数时，$a^{-1}=a^{-1}\times a^{p-1}=a^{p-2} \bmod p$，可以用快速幂来求逆元

求逆元问题。

P5431

基础概念

Explanation

对 p 取模实际上是一个将变量均匀映射到模 p 意义下整数集合的方法，而逆元是为了保留整数除法 (有理数) 的某些性质而做的处理。

对 p 取模实际上是一个将变量均匀映射到模 p 意义下整数集合的方法，而逆元是为了保留整数除法 (有理数) 的某些性质而做的处理。

- 满足 $ab \equiv 1 \bmod p$ 的 b $(b = a^{-1})$ 是唯一的。

基础概念

Explanation

对 p 取模实际上是一个将变量均匀映射到模 p 意义下整数集合的方法，而逆元是为了保留整数除法 (有理数) 的某些性质而做的处理。

- 满足 $ab \equiv 1 \bmod p$ 的 b $(b = a^{-1})$ 是唯一的。
- 满足结合律： $ab^{-1}c^{-1} \equiv a(bc)^{-1} \bmod p$

对 p 取模实际上是一个将变量均匀映射到模 p 意义下整数集合的方法，而逆元是为了保留整数除法 (有理数) 的某些性质而做的处理。

- 满足 $ab \equiv 1 \bmod p$ 的 b $(b = a^{-1})$ 是唯一的。
- 满足结合律： $ab^{-1}c^{-1} \equiv a(bc)^{-1} \bmod p$
- 满足分配律： $ac^{-1} + bc^{-1} \equiv (a+b)c^{-1} \bmod p$

关于质数的一些结论 (在此不作证明)

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个
- $(n, 2 \times n]$ 内必定存在质数 (Bertrand-Chebyshev 定理)

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个
- $(n, 2 \times n]$ 内必定存在质数 (Bertrand-Chebyshev 定理)
- 相邻质数的间隔很小, 约为 ln 级别

一些式子，其中 $n = \prod p_i^{k_i}$：

一些式子, 其中 $n = \prod p_i^{k_i}$:

- n 的因子个数: $\prod (k_i + 1)$

基础概念

Practice

一些式子, 其中 $n=\prod p_i^{k_i}$:

- n 的因子个数: $\prod(k_i+1)$
- n 的因子之和: $\prod \frac{p_i^{k_i+1}-1}{p_i-1}$

如何求出 n 的所有因子？

如何求出 n 的所有因子？

- 考虑若 d 是 n 的一个因子，那么一定有 $n = d \times \frac{n}{d}$，且 d 和 $\frac{n}{d}$ 之中一定有一个小于等于 $\sqrt{n}$

如何求出 n 的所有因子？

- 考虑若 d 是 n 的一个因子，那么一定有 $n = d \times \frac{n}{d}$，且 d 和 $\frac{n}{d}$ 之中一定有一个小于等于 $\sqrt{n}$
- 那么就可以直接枚举不大于 $\sqrt{n}$ 的因子，时间复杂度 $O(\sqrt{n})$

如何求出 n 的所有因子？

```cpp
vector<int> get(int n){
    vector<int> res;
    for(int i = 1; i * i <= n; i++){
        if(n % i == 0){
            res.push_back(i);
            if(i != n / i) res.push_back(n / i);
        }
    }
    return res;
}
```

如何求出 n 的所有质因子？

如何求出 n 的所有质因子？

- 考虑将 $n = \prod p_i^{k_i}$，那么从 2 开始从小到大枚举的第一个能把 n 整除的数必然是质数

如何求出 n 的所有质因子？

- 考虑将 $n = \prod p_i^{k_i}$, 那么从 2 开始从小到大枚举的第一个能把 n 整除的数必然是质数
- 然后将其记录下来，再把 n 除以当前数直到除不尽

如何求出 n 的所有质因子？

- 考虑将 $n = \prod p_i^{k_i}$，那么从 2 开始从小到大枚举的第一个能把 n 整除的数必然是质数
- 然后将其记录下来，再把 n 除以当前数直到除不尽
-

```
vector<int> get(int n){
    vector<int> res;
    for(int i = 2; i * i <= n; i++){
        if(n % i == 0){
            while(n % i == 0) {
                n /= i;
                res.push_back(i);
            }
        }
    }
    if(n > 1) res.push_back(n);
    return res;
}
```

ABC284 D

已知 $1 \le l \le r \le 10^9$，判断区间 $[l, r]$ 中质数的个数是否大于区间长度的三分之一（2019 ICPC 徐州 C）

已知 $1 \leq l \leq r \leq 10^9$，判断区间 $[l, r]$ 中质数的个数是否大于区间长度的三分之一（2019 ICPC 徐州 C）

- 质数的分布是越来越稀疏的，n 个数中近似有 $\frac{n}{\ln n}$ 个质数

已知 $1 \le l \le r \le 10^9$，判断区间 $[l, r]$ 中质数的个数是否大于区间长度的三分之一（2019 ICPC 徐州 C）

- 质数的分布是越来越稀疏的，n 个数中近似有 $\frac{n}{\ln n}$ 个质数
- 前 100 个数约有 30 个质数

已知 $1 \le l \le r \le 10^9$，判断区间 $[l, r]$ 中质数的个数是否大于区间长度的三分之一（2019 ICPC 徐州 C）

- 质数的分布是越来越稀疏的，n 个数中近似有 $\frac{n}{\ln n}$ 个质数
- 前 100 个数约有 30 个质数
- $r - l + 1 \le 100$？暴力：NO

$1 \le n \le 10^{18}$，求 $n! \bmod 1145141919810$ 的值。

目录

素数筛

Definition

筛法 (Sieve) 一般用于求出 $[1, n]$ 内的所有素数, 我们今天只讨论 $O(nlogn)$, $O(nloglogn)$, $O(n)$ 的筛法.

筛, 即用素数将非素数筛除.

关键思想在于尽可能利用已知信息, 减少一个数被筛的次数来降低复杂度.

素数筛

Method - 1

素数显然不是任何数的倍数，所以可以用已知数将其倍数筛掉。

```cpp
bool isPrime[N];
vector<int> pri;
void sieve(int n){
    isPrime[1] = true;
    for(int i = 2; i <= n; i++){
        if(!isPrime[i]) pri.push_back(i);
        for(int j = i * 2; j <= n; j += i){
            isPrime[j] = true;
        }
    }
}
```

显然数 i 会筛掉 $\frac{n}{i}$ 个数，总循环次数为：$\sum_{i=2}^{n} \frac{n}{i} \approx n \log n$，时间复杂度为 $O(n \log n)$

进一步考虑，素数的倍数涵盖了所有非素数，可以用已知的素数将其倍数筛掉。

```
bool isPrime[N];
vector<int> pri;
void sieve(int n){
    isPrime[1] = true;
    for(int i = 2; i <= n; i++){
        if(isPrime[i]) continue;
        pri.push_back(i);
        for(int j = i * 2; j <= n; j += i){
            isPrime[j] = true;
        }
    }
}
```

该算法的循环次数为 $\sum_{p \in Prime} \frac{n}{p} \approx n \ln \ln p$
此时每个数被筛的次数等于其不同的质因子数，这也说明前 n 个数的不同质因子个数之和是 $O(n \log \log n)$ 级别的.

素数筛

Method - 2.5

回顾之前的两个算法，一个数平均会被 $O(\log n)$ 或 $O(\log\log n)$ 个数筛除，如果我们能找到一种方法，使得每个数恰好被一个数筛除，这样的筛法就是 $O(n)$ 的了．为此，我们需要对每个数确定一个易于处理的**唯一**特征——最小质因子．

素数筛

Method - 3

假设我们当前枚举到了 n, 此时 $[2, n]$ 已经被处理过了, 原始方法是用 n 筛除 n 的倍数, 但如果我们想要用 n 的倍数的最小质因子筛除他们呢?

素数筛

Method - 3

假设我们当前枚举到了 n, 此时 $[2, n]$ 已经被处理过了, 原始方法是用 n 筛除 n 的倍数, 但如果我们想要用 n 的倍数的最小质因子筛除他们呢?

- n 的倍数 kn 的最小质因子只能是 n 的最小质因子或 k 的最小质因子

假设我们当前枚举到了 n, 此时 $[2, n]$ 已经被处理过了, 原始方法是用 n 筛除 n 的倍数, 但如果我们想要用 n 的倍数的最小质因子筛除他们呢?

- n 的倍数 kn 的最小质因子只能是 n 的最小质因子或 k 的最小质因子
- 假设 n 的最小质因子为 p, $n = ap$, 枚举小于 p 的**质数** k (当 k 不小于 p 时说明 kap 已经被 p 筛过了) 作为 kn 的最小质因子

素数筛

Method - 3

```
bool isPrime[N];
vector<int> pri;
void sieve(int n){
    isPrime[1] = true;
    for(int i = 2; i <= n; i++){
        if(!isPrime[i]) pri.push_back(i);
        for(int j = 0; j < pri.size() && pri[j] * i <= n; j++){
            isPrime[pri[j] * i] = true;
            if(i % pri[j] == 0) break;
        }
    }
}
```

素数筛

Problem - 1

CF 236B

CF 27E

CF 1766D

END