Pupil's Math

Hile Meow

ZUCC ACM Group

2022.1.27

Pupil's Math

Just Pupil's Math:

- Fast Power Review
- Easy Number Theory
- Sieve
- Other

Definition

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

Definition

- 给定二元组 $(A,*)$, 其中 A 是集合,$*$ 是满足**结合律**的运算, 求 $a*a*\ldots*a,\ a\in A$(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 $\sum_i x_i 2^i (x_i = 0\ or\ 1)$ 的形式，使得 $O(k)$ 次运算变成 $O(\log_2 k)$ 次运算

Definition

- 给定二元组 $(A, *)$, 其中 A 是集合,$*$ 是满足**结合律**的运算, 求 $a * a * \ldots * a$, $a \in A$(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 $\sum_i x_i 2^i (x_i = 0 \text{ } or \text{ } 1)$ 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

Definition

- 给定二元组 $(A, *)$, 其中 A 是集合,$*$ 是满足**结合律**的运算, 求 $a * a * \ldots * a$, $a \in A$(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 $\sum_i x_i 2^i (x_i = 0 \ or\ 1)$ 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
- 准确的运算次数为 $\lfloor \log_2 k \rfloor + popcount(k) - 1$,$popcount(x)$ 为 x 的二进制表示中 1 的个数

Code

```cpp
const int P=998244353;
ll qpow(ll a,ll n=P-2,ll m=P,ll x=1){
    while(n){
        if(n&1)x=x*a%m;
        n>>=1,a=a*a%m;
    }
    return x;
}
```

Problem 1

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

Problem 1

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

- 模板题, 复杂度 $O(\log b)$

Problem 1

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

- 模板题, 复杂度 $O(\log b)$
- 有些写法对于 $a = 0$ 的情况会输出 1

Problem 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \mod p$, 不许用 __$int128$ 捏

Problem 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \mod p$, 不许用 $\_\_int128$ 捏

- 计算过程溢出了?

Problem 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \mod p$, 不许用 $\_\_int128$ 捏

- 计算过程溢出了?
- 考虑加法运算依旧满足结合律

Problem 2

$0 \leq a, b, p \leq 10^{18}$, 求 $a^b \mod p$, 不许用 $\_\_int128$ 捏

- 计算过程溢出了?
- 考虑加法运算依旧满足结合律
- 将 $a \times b$ 改写为 $\sum_i a x_i 2^i, (x_i = 0 \ or \ 1)$, 其中 $\sum_i x_i 2^i = b$

Problem 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \mod p$, 不许用 $\_\_int128$ 捏

- 计算过程溢出了?
- 考虑加法运算依旧满足结合律
- 将 $a \times b$ 改写为 $\sum_i ax_i 2^i, (x_i = 0\ or\ 1)$, 其中 $\sum_i x_i 2^i = b$
- 由于 a 在运算中最多被扩大两倍, 而 long long 最大值约为 $9.2233e18$，所以不会溢出

Problem 2

$0 \le a, b, p \le 10^{18}$, 求 $a^b \mod p$, 不许用 $\_\_int128$ 捏

- 计算过程溢出了?
- 考虑加法运算依旧满足结合律
- 将 $a \times b$ 改写为 $\sum_i ax_i 2^i, (x_i = 0 \ or \ 1)$, 其中 $\sum_i x_i 2^i = b$
- 由于 a 在运算中最多被扩大两倍, 而 long long 最大值约为 $9.2233e18$，所以不会溢出
- 复杂度一般看作 $O(\log^2 b)$

Problem 3

已知 1 到 n 的排列 $p_1, p_2, \dots, p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列

$1 \le n \le 10^5, \ 1 \le k \le 10^9$

Problem 3

已知 1 到 n 的排列 $p_1, p_2, \ldots, p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列

$1 \le n \le 10^5, \ 1 \le k \le 10^9$

- 显然可以暴力 $O(nk)$ 模拟

Problem 3

已知 1 到 n 的排列 $p_1, p_2, \ldots, p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列
$1 \le n \le 10^5,\ 1 \le k \le 10^9$

- 显然可以暴力 $O(nk)$ 模拟
- 发现排列的置换操作有结合律

Problem 3

已知 1 到 n 的排列 $p_1, p_2, ..., p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列
$1 \le n \le 10^5, \ 1 \le k \le 10^9$

- 显然可以暴力 $O(nk)$ 模拟
- 发现排列的置换操作有结合律
- 可以利用快速幂优化到 $O(n \log k)$

Problem 3

已知 1 到 n 的排列 $p_1, p_2, ..., p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列
$1 \le n \le 10^5,\ 1 \le k \le 10^9$

- 显然可以暴力 $O(nk)$ 模拟
- 发现排列的置换操作有结合律
- 可以利用快速幂优化到 $O(n \log k)$
- 快速幂的本质还是二分 (分治)

Problem 3

已知 1 到 n 的排列 $p_1, p_2, ..., p_n$ 和一个整数 k, 每次操作会将第 i 个数变为第 p_i 个数, 求 k 次操作后的排列
$1 \leq n \leq 10^5,\ 1 \leq k \leq 10^9$

- 显然可以暴力 $O(nk)$ 模拟
- 发现排列的置换操作有结合律
- 可以利用快速幂优化到 $O(n \log k)$
- 快速幂的本质还是二分 (分治)
- 本题时间复杂度可以做到 $O(n)$, 留作思考题

Summary

很多符合结合律的运算都可以 $\log$ 次快速幂优化暴力，不仅仅是 **加法**和**乘法**，常用的还有**置换**，**矩阵乘法**，**多项式乘法**...
由于我们要学的是 Pupil's Math，其他应用就不讲了捏

Definition

先来几个通用定义 (为求易于理解, 放弃了一些严谨性)
没有特殊说明的情况下, 小写字母表示整数, 大写字母表示集合

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 $n\%p = b$, 当 $n\%p = 0$ 时称 p **整除** n, 写作 $p|n$,p 称为 n 的**因子**.
若 $A = \{n\%p|n \in S\}$, 称 A 为 S 模 p 的剩余系, 当 $A = \{x|x \in [0, p)\}$ 时为完全剩余系.

素数 (质数) 与唯一分解定理

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

Definition

gcd 与 lcm

$gcd(a,b)$ 表示 a 和 b 的最大公约数, 即 $gcd(a,b)=\prod p_i^{min(k_{ai},k_{bi})}$
$lcm(a,b)$ 表示 a 和 b 的最小公倍数, 即 $lcm(a,b)=\prod p_i^{max(k_{ai},k_{bi})}$
显然有 $a\times b=gcd(a,b)\times lcm(a,b)$
当 $gcd(a,b)=1$ 时, 我们称 a,b 互质, 记作 $a\perp b$

欧拉函数

用欧拉函数 $\varphi(n)$ 来表示 $[1,n]$ 内与 n 互质的数的数量, 可以在 $O(\sqrt{n})$ 时间内求得 $\varphi(n)$

欧拉定理

若 $a\perp p$, 则有 $a^{\varphi(p)}=1 \mod p$, 当 $p\in Prime$ 时称为费马小定理

Definition

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 $a \times b^{-1} \mod p$ 表示 $\frac{a}{b}$

积性函数

若函数 f 对于任意互质正整数 a, b 都满足 $f(ab) = f(a)f(b)$, 则称 f 为积性函数
当 a, b 不需要互质时,f 称作完全积性函数.

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

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

- 对于任意 a, p, 满足 $\frac{a}{b} = 1$ 的 b $(b = a)$ 是唯一的:: 满足 $ab \equiv 1 \mod p$ 的 b $(b = a^{-1})$ 是唯一的

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

- 对于任意 a, p, 满足 $\frac{a}{b} = 1$ 的 b $(b = a)$ 是唯一的:: 满足 $ab \equiv 1 \mod p$ 的 b $(b = a^{-1})$ 是唯一的
- $a/b/c = a/(bc)::ab^{-1}c^{-1} \equiv a(bc)^{-1} \mod p$(结合律)

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

- 对于任意 a, p, 满足 $\frac{a}{b} = 1$ 的 b $(b = a)$ 是唯一的:: 满足 $ab \equiv 1 \mod p$ 的 b $(b = a^{-1})$ 是唯一的
- $a/b/c = a/(bc)::ab^{-1}c^{-1} \equiv a(bc)^{-1} \mod p$(结合律)
- $a/c + b/c = (a+b)/c::ac^{-1} + bc^{-1} \equiv (a+b)c^{-1} \mod p$(分配律)

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

- 对于任意 a, p, 满足 $\frac{a}{b} = 1$ 的 b $(b = a)$ 是唯一的:: 满足 $ab \equiv 1 \mod p$ 的 b $(b = a^{-1})$ 是唯一的
- $a/b/c = a/(bc)::ab^{-1}c^{-1} \equiv a(bc)^{-1} \mod p$(结合律)
- $a/c + b/c = (a+b)/c::ac^{-1} + bc^{-1} \equiv (a+b)c^{-1} \mod p$(分配律)
- $a/b + c/d = (ad+cb)/(bd)::ab^{-1} + cd^{-1} \equiv (ad^{-1} + cb^{-1})(bd)^{-1} \equiv ab^{-1} + cd^{-1} \mod p$

Tricks

关于质数的一些结论 (在此不作证明)

Tricks

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个

Tricks

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个
- $(n, 2*n]$ 内必存在质数 (Bertrand-Chebyshev 定理)

Tricks

关于质数的一些结论 (在此不作证明)

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

Tricks

关于质数的一些结论 (在此不作证明)

- 质数有无穷多个
- $(n, 2*n]$ 内必存在质数 (Bertrand-Chebyshev 定理)
- 相邻质数的间隔很小, 约为 $\ln$ 级别
- **素数倒数和** $\lim_{p\to\infty, p\in Prime}\sum\frac{1}{p}\approx\ln\ln p$

Practice

一些推柿子练习, 其中 $n = \prod p_i^{k_i}$

Practice

一些推柿子练习，其中 $n = \prod p_i^{k_i}$

- n 的因子个数 $d(n) = \prod(k_i + 1)$

Practice

一些推柿子练习, 其中 $n = \prod p_i^{k_i}$

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

Practice

一些推柿子练习，其中 $n = \prod p_i^{k_i}$

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

Practice

一些推柿子练习, 其中 $n = \prod p_i^{k_i}$

- n 的因子个数 $d(n) = \prod(k_i + 1)$
- n 的因子之和 $\sigma(n) = \prod \frac{p_i^{k_i+1}-1}{p_i-1}$
- $\varphi(n) = \prod p_i^{k_i-1}(p_i - 1) = n\prod(1 - \frac{1}{p_i})$
- $d(n), \sigma(n)$ 都是积性函数

Problem 2

如何求 n 的**所有因子**?

Problem 2

如何求 n 的**所有因子**?

- 考虑如果有 $d|n$, 则有 $\frac{n}{d}|n$, 二者至少有一个小于 $\sqrt{n}$

Problem 2

如何求 n 的**所有因子**?

- 考虑如果有 $d|n$, 则有 $\frac{n}{d}|n$, 二者至少有一个小于 $\sqrt{n}$
- 枚举所有不大于 $\sqrt{n}$ 的因子, 时间复杂度 $O(\sqrt{n})$

Problem 2

如何求 n 的**所有因子**?

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

Problem 3

如何求 n 的**所有质因子**?

Problem 3

如何求 n 的**所有质因子**?

- 从小到大枚举到的第一个能整除 n 的数必然是质数

Problem 3

如何求 n 的**所有质因子**?

- 从小到大枚举到的第一个能整除 n 的数必然是质数
- 如果可以整除就记录，然后把 n 除以当前数直到除不尽. 小心大于 $\sqrt{n}$ 的质数.

Problem 3

如何求 n 的**所有质因子**?

- 从小到大枚举到的第一个能整除 n 的数必然是质数
- 如果可以整除就记录, 然后把 n 除以当前数直到除不尽. 小心大于 $\sqrt{n}$ 的质数.
- 时间复杂度 $O(\sqrt{n})$

Problem 3

如何求 n 的**所有质因子**?

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

Problem 5

$1 \leq c \leq 10^{18}$, 求满足 $ab^2 = c$ 且 $gcd(a, b) = 1$ 的最小的 a.

Problem 5

$1 \le c \le 10^{18}$, 求满足 $ab^2 = c$ 且 $gcd(a, b) = 1$ 的最小的 a.

- 答案最小的 a 一定不含偶数次方因子 p^{2k}, 否则可以放到 b 上使得答案更优

Problem 5

$1 \leq c \leq 10^{18}$, 求满足 $ab^2 = c$ 且 $gcd(a, b) = 1$ 的最小的 a.

- 答案最小的 a 一定不含偶数次方因子 p^{2k}, 否则可以放到 b 上使得答案更优
- 因此只要求出所有出现偶数次的 c 的质因子之积

Problem 5

$1 \leq c \leq 10^{18}$, 求满足 $ab^2 = c$ 且 $gcd(a, b) = 1$ 的最小的 a.

- 答案最小的 a 一定不含偶数次方因子 p^{2k}, 否则可以放到 b 上使得答案更优
- 因此只要求出所有出现偶数次的 c 的质因子之积
- 细节: 考虑存在大于 $O(n^{\frac{1}{3}})$ 的素数平方的情况

Problem 5

$1 \le c \le 10^{18}$, 求满足 $ab^2 = c$ 且 $gcd(a, b) = 1$ 的最小的 a.

- 答案最小的 a 一定不含偶数次方因子 p^{2k}, 否则可以放到 b 上使得答案更优
- 因此只要求出所有出现偶数次的 c 的质因子之积
- 细节: 考虑存在大于 $O(n^{\frac{1}{3}})$ 的素数平方的情况
- 时间复杂度 $O(n^{\frac{1}{3}})$

Problem 7

已知 $1 \le l \le r \le 10^9$, 判断区间 $[l, r]$ 素数是否大于区间长度的 $\frac{1}{3}$(2019ICPC 徐州 C)

Problem 7

已知 $1 \le l \le r \le 10^9$, 判断区间 $[l, r]$ 素数是否大于区间长度的 $\frac{1}{3}$(2019ICPC 徐州 C)

- 素数分布是越来越稀疏的 $(\frac{n}{\ln n})$

Problem 7

已知 $1 \leq l \leq r \leq 10^9$, 判断区间 $[l, r]$ 素数是否大于区间长度的 $\frac{1}{3}$(2019ICPC 徐州 C)

- 素数分布是越来越稀疏的 $(\frac{n}{\ln n})$
- 前 100 个数约有 30 个左右的素数

Problem 7

已知 $1 \le l \le r \le 10^9$, 判断区间 $[l, r]$ 素数是否大于区间长度的 $\frac{1}{3}$(2019ICPC 徐州 C)

- 素数分布是越来越稀疏的 $(\frac{n}{\ln n})$
- 前 100 个数约有 30 个左右的素数
- $r - l \le 100$? 暴力:NO;

Problem 11

已知 $1 \leq n \leq 10^{18}$, 求 $n!\%1145141919810$ 的值

Problem 11

已知 $1 \le n \le 10^{18}$, 求 $n!\%1145141919810$ 的值

- 当模数不常见时要从模数的特殊性考虑解法

Problem 11

已知 $1 \leq n \leq 10^{18}$, 求 $n!\%1145141919810$ 的值

- 当模数不常见时要从模数的特殊性考虑解法
- 本地打表发现 $1145141919810 = 2 \times 3^2 \times 5 \times 7 \times 101^2 \times 178187$

Problem 11

已知 $1 \le n \le 10^{18}$, 求 $n!\%1145141919810$ 的值

- 当模数不常见时要从模数的特殊性考虑解法
- 本地打表发现 $1145141919810 = 2 \times 3^2 \times 5 \times 7 \times 101^2 \times 178187$
- 当 $n \ge 178187$ 时,$n!$ 必然能被 1145141919810 整除

Problem 11

已知 $1 \le n \le 10^{18}$, 求 $n!\%1145141919810$ 的值

- 当模数不常见时要从模数的特殊性考虑解法
- 本地打表发现 $1145141919810 = 2 \times 3^2 \times 5 \times 7 \times 101^2 \times 178187$
- 当 $n \ge 178187$ 时,$n!$ 必然能被 1145141919810 整除
- Extra: 若模数是 $10^9 + 7$ 呢?

Problem 13

已知一个不含前导零的十位数 n, 其中 n 的所有位构成 0 到 9 的排列, 对任意的 $k \in [1, 10]$ 满足前 k 位形成的数可以被 k 整除, 求 n(Online Math Contest 063F)
留作思考题.

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 $O(n\log n)$, $O(n\log\log n)$, $O(n)$ 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.

Method 1

考虑一个简单的想法:
根据素数的定义, 素数显然不是任何数的倍数, 我们可以用已知的数将其倍数筛掉:

```
bool np[N];
int pri[N],cnt;
void sieve(int n){
    np[1]=1;
    for(int i=2;i<=n;i++){
        if(!np[i])pri[++cnt]=i;
        for(int j=2*i;j<=n;j+=i){
            np[j]=1;
        }
    }
}
```

Method 1

显然第 i 个数会筛掉 $\frac{n}{i}$ 个数, 总循环次数为 $\sum_{i=1}^{n} \frac{n}{i} = n\sum_{i=1}^{n} \frac{1}{i} \approx n\log n$, 所以复杂度为 $O(n\log n)$.
此时每个数都会被他的所有因子筛一次, 这也说明前 n 个数的因子数之和是 $O(n\log n)$ 级别的.

Method 2

进一步考虑:
素数的倍数涵盖了所有非素数, 我们可以用已知的**素数**将其倍数筛掉:

```
bool np[N];
int pri[N],cnt;
void sieve(int n){
    np[1]=1;
    for(int i=2;i<=n;i++){
        if(np[i])continue;
        pri[++cnt]=i;
        for(int j=2*i;j<=n;j+=i){
            np[j]=1;
        }
    }
}
```

Method 2

用同样的方法分析, 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.

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.

Method 3

考虑在原始方法上优化:
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?

Method 3

考虑在原始方法上优化:
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

Method 3

考虑在原始方法上优化:
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

Problem 1

对 $[2, 500000]$ 质因子分解.

Problem 1

对 $[2, 500000]$ 质因子分解.

- Solution.1:$5e5$ 次 $O(\sqrt{n})$ 分解?$(5e5^{3/2} > 3e8)$

Problem 1

对 $[2, 500000]$ 质因子分解.

- Solution.1:$5e5$ 次 $O(\sqrt{n})$ 分解?($5e5^{3/2} > 3e8$)
- 可以 AC, 因为单次均摊复杂度是 $O(\log n)$

Problem 1

Solution.2: 在线性筛时记录最小质因子, 之后稳定 $O(\log n)$ 求质因子

Problem 1

Solution.3: 埃氏筛, 又短又快又优雅, **无法直接求质因子次数**

Problem 4

求 $[2, 500000]$ 的欧拉函数.

Problem 4

求 $[2, 500000]$ 的欧拉函数.

- φ 是积性函数

Problem 4

求 $[2, 500000]$ 的欧拉函数.

- φ 是积性函数
- 筛倍数时分两种情况讨论.

Problem 27

已知 $1 \le n \le 500000$, 求最小的满足因子数为 2^n 的数, 答案对 10^9+7 取模 (Project Euler 500)

Problem 27

已知 $1 \le n \le 500000$, 求最小的满足因子数为 2^n 的数, 答案对 10^9+7 取模 (Project Euler 500)

- 考虑一个数的因子数等于 (各质因子次数 $+1$) 之积

Problem 27

已知 $1 \leq n \leq 500000$, 求最小的满足因子数为 2^n 的数, 答案对 10^9+7 取模 (Project Euler 500)

- 考虑一个数的因子数等于 (各质因子次数 $+1$) 之积
- 因子数为 2^n 意味着各质因子的次数必须形如 2^k-1

Problem 27

已知 $1 \le n \le 500000$, 求最小的满足因子数为 2^n 的数, 答案对 10^9+7 取模 (Project Euler 500)

- 考虑一个数的因子数等于 (各质因子次数 $+1$) 之积
- 因子数为 2^n 意味着各质因子的次数必须形如 2^k-1
- $n \le 500000$, 意味着答案不超过前 500000 个质数的乘积

Problem 27

已知 $1 \le n \le 500000$, 求最小的满足因子数为 2^n 的数, 答案对 10^9+7 取模 (Project Euler 500)

- 考虑一个数的因子数等于 (各质因子次数 $+1$) 之积
- 因子数为 2^n 意味着各质因子的次数必须形如 2^k-1
- $n \le 500000$, 意味着答案不超过前 500000 个质数的乘积
- 筛出前 500000 个质数扔进小根堆, 每次取出堆顶, 并将其平方放回.

Combinatorics

对于 CF2100 之前的内容, 高中数学够用且不复习了, 在此只讲模意义下组合数的计算.

- $C(n, m) = C(n-1, m-1) + C(n-1, m),\ \ O(n^2) - O(1)$

Combinatorics

对于 CF2100 之前的内容, 高中数学够用且不复习了, 在此只讲模意义下组合数的计算.

- $C(n,m) = C(n-1,m-1) + C(n-1,m),\ \ O(n^2) - O(1)$
- $C(n,m) = \frac{n!}{(n-m)!m!},\ \ O(n) - O(\log n)$

Combinatorics

对于 CF2100 之前的内容, 高中数学够用且不复习了, 在此只讲模意义下组合数的计算.

- $C(n,m) = C(n-1,m-1) + C(n-1,m),\ \ O(n^2) - O(1)$
- $C(n,m) = \frac{n!}{(n-m)!m!},\ \ O(n) - O(\log n)$
- $(n!)^{-1} \equiv (n+1)!^{-1}(n+1) \mod p,\ \ O(n) - O(1)$

Combinatorics

对于 CF2100 之前的内容, 高中数学够用且不复习了, 在此只讲模意义下组合数的计算.

- $C(n, m) = C(n-1, m-1) + C(n-1, m),\ \ O(n^2) - O(1)$
- $C(n, m) = \frac{n!}{(n-m)!m!},\ \ O(n) - O(\log n)$
- $(n!)^{-1} \equiv (n+1)!^{-1}(n+1) \mod p,\ \ O(n) - O(1)$
- 存在 $O(n)$ 的预处理任意 n 个数逆元的 Trick, 但我们不讲 (

Practice

推式子练习:

- $\sum_{i=0}^{n} C(n, i) = 2^n$

Practice

推式子练习:

- $\sum_{i=0}^{n} C(n,i) = 2^n$
- $\sum_{i=0}^{n} (-1)^{n-i} C(n,i) = 0$

Practice

推式子练习:

- $\sum_{i=0}^{n} C(n, i) = 2^n$
- $\sum_{i=0}^{n} (-1)^{n-i} C(n, i) = 0$
- $\sum_{i=0}^{n} C(n, i) \times i = n2^{n-1}$

Practice

推式子练习:

- $\sum_{i=0}^{n} C(n,i) = 2^n$
- $\sum_{i=0}^{n} (-1)^{n-i} C(n,i) = 0$
- $\sum_{i=0}^{n} C(n,i) \times i = n2^{n-1}$
- $x_1 + x_2 + \ldots + x_n = m$ 非负整数解的数目 $C(n+m-1, m)$

Practice

推式子练习:

- $\sum_{i=0}^{n} C(n,i) = 2^n$
- $\sum_{i=0}^{n}(-1)^{n-i}C(n,i) = 0$
- $\sum_{i=0}^{n} C(n,i) \times i = n2^{n-1}$
- $x_1 + x_2 + \ldots + x_n = m$ 非负整数解的数目 $C(n+m-1,m)$
- $\sum_{i=0}^{m} C(n+i,i) = C(n+m+1,m)$

The End, Any Question?