范围内寻找素数。

欧拉筛法，线性筛法

- 和埃氏筛法之间有什么差别?
    - 埃氏筛法去合数的过程中。可能会重复经理某一些数字，存在一些多余的计算。

```
void init() {
  for (int i = 2; i < MAXN; ++i) {
    if (!vis[i])
      pri[cnt++] = i;
    for (int j = 0; j < cnt; ++j) {
      if (1ll * i * pri[j] >= MAXN) break;
      vis[i * pri[j]] = 1;
      if (i % pri[j] == 0)
        break;
    }
  }
}
```

- 来自jly的板

```
int mp[maxn+1];
vector<int>primes;

for (int i = 2; i <= maxn; i++)
    {
        if (!mp[i])
        {
            mp[i] = i;
            primes.push_back(i);
        }
        for (auto p : primes)
        {
            if (p * i > maxn)
                break;
            mp[i * p] = p;
            if (i % p == 0)
                break;
        }
    }
```

- 这个可以方便的拆分素数。

矩阵

- 常见应用：
  - 给定路的长度。查询路径。
- 一个n阶矩阵板子。其中，乘法由快速幂的思想优化。总体上的复杂度是$n^3log(k)$

```
const int maxn = 100 + 10;

const int mod = 1e9 + 7;

struct matrix
{
    static const int n = maxn;
    ll M[n][n];
    int N;
    matrix(int num)
    {
        N = num;
        memset(M, 0, sizeof(M));
    }
    void build()
    {
        for (int i = 1; i <= N; i++)
            M[i][i] = 1;
    }
    ll *operator[](int i)
    {
        return M[i];
    }
    matrix operator*(matrix &a)
    {
        matrix temp(N);
        for (int i = 1; i <= N; i++)
            for (int j = 1; j <= N; j++)
                for (int k = 1; k <= N; k++)
                    temp[i][j] = (temp[i][j] + (1LL * M[i][k] * a[k][j]) % mod) % mod;
        return temp;
    }
    matrix operator^(ll P)
    {
        matrix res(N);
        res.build();
        matrix A = *this;
        while (P > 0)
        {
            if (P & 1)
                res = res * A;
            A = A * A;
            P >>= 1;
        }
        return res;
    }
};
```

柯西不等式

- 柯西不等式，应用。

$$\text{二维形式：}$$
$$(a^2+b^2)\times(c^2+d^2)>=(ac+bd)^2$$
$$\text{一般形式：}$$
$$\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2 = \left(\sum_{i=1}^{n} a_i b_i\right)^2 \tag{1}$$
$$\text{众多等价形式：}$$
$$\sum_{i=1}^{n}\left(\frac{ai^2}{bi}\right) = \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{\sum_{i=1}^{n} b_i}$$

题目简介

```
给定n个整数，求众数。
```

20min

```
摩尔投票法：O（n）空间复杂度常数值
哈希方法。nlog(n)
桶排序    O（n）    O(max(ai))

摩尔投票法：
用两个变量，一个记录经过前面一大段之后的胜者，以及它剩下的可抵消的次数。
如果一个数，的出现次数大于数组的长度一半，向下取整。那么，必然会是赢家，反之。
```

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 10;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    int n;
    while (cin >> n)
    {
        int a = 0, b = 0;
        for (int i = 1; i <= n; i++)
        {
            int k;
            cin >> k;
            if (b == 0)
                a = k, b = 1;
            else if (a == k)
                b++;
```

```
            else
                b--;
        }
        cout << a << '\n';
    }
}
```

组合数板子

一共提供了四种求组合数的方法：资料来源于acwing

第一种方法：性质+递推+预处理：

将组合数得计算分解为小规模得问题实现：

$$C_r^n = C_{r-1}^n + C_{r-1}^{n-1} \tag{2}$$

最小规模得子问题如C_r^0, C_r^1已知（容易计算。）

然后就可以把所有情况求出来：

code

```
const int N_c = 3E3;
const int mod = 1E9 + 7;
int c[N_c][N_c];
void C_init() {
    for (int i = 1; i < N_c; ++i) {
        c[i][0] = c[i][i] = 1;
        for (int j = 1; j < i; ++j) {
            c[i][j] = (c[i - 1][j] + c[i - 1][j - 1]) % mod;
        }
    }
}
//注意范围
//小心me
```

tips

1. 数组范围允许。

第二种方法： 逆元+预处理+组合数定义

basic

$$C_n^a = \frac{n!}{(n-a)! \times b!} \tag{3}$$

可以考虑先将分母得阶乘法求出。然后求出其逆元（由于mod是一个质数，逆元可以通过费马小定理求出。）

综上：时间复杂度：

$n + alog(a)$:

[AcWing 886. 求组合数 II-数论-C++ - AcWing]

code

```cpp
using ll = long long;
const int N_c = 1E5 + 10;
const int mod = 1e9 + 7;

int fac[N_c] , infac[N_c];
ll quickly_pow(ll x, ll n, ll p)
{
    ll res = 1;
    while (n > 0)
    {
        if (n & 1)res = res * x % p;
        x = x * x % p;
        n >>= 1;
    }
    return res;
}
void init() {
    fac[0] = infac[0] = 1;
    for (int i = 1; i < N_c; i++)
    {
        fac[i] = 1LL * fac[i - 1] * i % mod;
        infac[i] = 1LL * infac[i - 1] * quickly_pow(i, mod - 2, mod) % mod;
    }
}
int c(int a , int b) {
    return 1LL * fac[a] * infac[b] % mod * infac[a - b] % mod;
}
/*
*记得初始化。
*/
```

[乘法逆元（inverse element）及四大相关求法详解（含证明）*NothingAtall.的博客-CSDN博客乘法逆元*]()

还得看大佬博客

定义

对于线性同余方程$ax \equiv 1(mod \quad b)$

称x为a mod b的逆元记作a^{-1}

性质

逆元存在:

1. 当$gcd(a, mod) = 1$时，逆元才有解。

逆元计算

费马小定理：）

1. 内容：

$$
\begin{aligned}
&1.\text{假如}a\text{是一个整数。}m\text{是质数。}\\
&a^m \equiv a(mod \quad m)
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
&2.\text{假设}m\text{不是质数},gcd(a,m)=1\\
&a^{m-1} \equiv 1(mod \quad m)\\
&a*a^{m-2} \equiv 1(mod \equiv m)\\
&\text{综上}a^{m-2}\text{是}a\text{的逆元。}
\end{aligned}
\tag{5}
$$

- 前提约束：mod数是质数。

其实就是写一个快速幂运算。

```
//inverse_element.
ll mod;
ll qpow( ll x , ll n , ll p = mod)
{
    ll res = 1;
    while (n > 0) {
        if (n & 1) res = res * x % p;
        x = x * x % mod;
        n >>= 1;
    }
    return res;
}

ll inv(ll x , ll p = mod)
{
    return qpow( x , p - 2);
}
/*
* 1. mod定义
* 2. 使用前提：p是质数，且x，p互质。
*/
```

其它的有机会再补。靠队友啦 qaq

chenjiuri_kuaisumi_basicproblem

```
ll quickly_pow(ll x,ll n,ll mod)
{
    ll res=1;//用来返回结果。
    while(n>0)
    {
        if(n&1)res=res*x%mod;
        x=x*x%mod;
        n>>=1;
    }
    return res;
}
```

- 分析模板
  - 当前的剩余部分是否可以被x^2（x实际意义上是多少次方）整除.
    - 如果可以不需要处理，
    - 如果不可以，余出来的部分必然是x,将它先乘在结果之上即可。

欧几里得算法（辗转相除法）

- 应用
  - 求最大公约数

```
欧几里得算法描述：
当我们要求两个数的最大公约数  a,b;
那么    设 gcd(x,y)为和y之间的最大公约数的值。


```

$$\begin{cases} g\left(x,y\right)=g(y,x\%y), & x!=0\&\&y!=0 \\ g(x,0)=x \end{cases} \tag{6}$$

```
typedef long long ll;
ll gcd(ll x,ll y){
    return y?x:gcd(y,x%y);
}
```

- 求最小公倍数

$$a\text{与}b\text{的最小公倍数为:} \\ \frac{a\times b}{gcd(a,b)} \tag{7}$$

拓展

- 多个数的最大公约数

$$gcd(a_1,a_2,a_3,a_4.\dots,a_n) \\ =gcd(a_1,gcd(a_{1.....m})) \\ gcd=(a_1....a_{n-2},gcd(a_{n-1},a_n)) \tag{8}$$

- 翡蜀定理

$$
\begin{gathered}
\text{对于两个正整数} a,b; \\
\text{有} x*a+y*b=k*gcd(a,b); \\
x\text{，}y\text{为正整数。} \\
\text{对于多个依然成立} \\
\text{关注一个数的角度是，一个两个整数之间向相乘，} \\
\text{很轻易就能看出来。} \\
\text{对于某一个变量的加一减一不过只是相差了一个} gcd(a,b);
\end{gathered} \tag{9}
$$

一个明智地追求快乐的人，除了培养生活赖以支撑的主要兴趣之外，总得设法培养其他许多闲情逸致。