数学 04

- 素数和素数判定
- 筛法，埃氏筛和欧拉筛
- 分解质因子，计算 $n!$ 中质数因子 p 出现的次数
- miller-rabin 素性检测（选讲）
- 题目选讲

素数

只能被 1 和自身整除的数叫做素数，能被除 1 和自身外的的数整除的数叫做合数

- 任何一个大于1 的正数都可以被写成其质因子的乘积。
- 任何一个合数 n 都至少有一个不超过 $\sqrt{n}$ 的质因子

第二个性质的证明

设 n 为合数，a，b 是 n 的两个因子，即 $n = a \cdot b$，显然有 $a \leq \sqrt{n}$ 或者 $b \leq \sqrt{n}$

不妨设 $a \leq \sqrt{n}$，如果 a 本身为素数，则定理成立

如果 a 本身是合数，那么 a 有不大于 a 的质因子，则定理也成立

以上两种情况均表明 n 有不大于 $\sqrt{n}$ 的质因子

素数判定

根据性质，如果 n 没有超过 $\sqrt{n}$ 的质因子，那么 n 是质数，由此可以写出素数判定的代码，时间复杂度 $O(\sqrt{n})$

```
bool isPrime(int n)
{
    if (n == 0 || n == 1) return false;
    for (int i = 2; i * i <= n; ++i) {
        if (n % i == 0) return false;
    }
    return true;
}
```

筛法

筛法指的是筛选 $1 \sim n$ 内的所有素数

埃氏筛

一开始将所有数全部标记为素数，从 2 开始枚举 i，如果 i 是素数，则将 i 的所有倍数全部标记为合数，如果 i 是合数，跳过当前循环

复杂度 $O(nloglogn)$

```cpp
const int N = 1e5 + 7;
int n, prime[N];
bool vis[N];
int Era_sieve()
{
    int cnt = 0; //length of prime table
    for (int i = 2; i < N; ++i) {
        if (!vis[i]) prime[++cnt] = i;
        for (int j = 2; j * i < N; ++j)
            vis[j * i] = 1;
    }
    return cnt;
}
```

欧拉筛

埃氏筛的缺点在于，同一个合数会被重复筛除

例如 $30 = 3 * 10 = 5 * 6$，分别在 3，5 的时候被筛除

欧拉筛用每个合数的最小质因子去筛除该数，达到“每个合数被筛且仅被筛一次”的效果，从而复杂度达到线性 $O(n)$

```cpp
const int N = 1e5 + 7;
int n, prime[N];
bool vis[N];
int Euler_sieve()
{
    int cnt = 0;
    for(int i = 2; i < N; ++i)
        vis[i] = 1;
    for(int i = 2; i < N; ++i){
        if(vis[i]) prime[++cnt] = i;
        for(int j = 1; j <= cnt && prime[j] * i < N; ++j){
            vis[prime[j] * i] = 0;
            if(i % prime[j] == 0) break;
        }
    }
    return cnt;
}
```

分解质因子

根据算数基本定理，任意一个正整数 n 可以分解成质因子指数的乘积，假设其有 j 个质因子，那么可以形式化的写为

$$n = \prod_{i=1}^{j} p_i^{k_i}$$

对于分解一个整数 n，我们可以使用试除法

先预处理质数表，然后对于所有可能的质数 p_i，用 n 不断除 p_i，直到除不尽，换下一个质数，当 p_i 比 n 大时，终止循环，如果 n 最后大于 1，那么此时的 n 也是一个质数因子

还有其他比较高级的算法处理这个问题，不过这个算法的时间复杂度也比较优秀

代码如下

```
vector<int> getPrimeFac(vector<int> prime, int n)
{
    int cnt = prime.size();
    vector<int> vec;
    for (int i = 0; i < cnt && prime[i] <= n; ++i) {
        int flag = 0;
        while (n % prime[i] == 0) {
            n /= prime[i];
            flag = 1;
        }
        if (flag) vec.push_back(prime[i]);
    }
    if (n > 1) vec.push_back(n);
    return vec;
}
```

计算 $n!$ 中质数 p 出现的次数

以 2 为例，我们知道 $1 \sim n$ 中能被 2 整除的数有 $\lfloor \frac{n}{2} \rfloor$ 个，这么多数各自贡献一个 2，能被 4 整除的数有 $\lfloor \frac{n}{4} \rfloor$，这么多数各自在贡献一个 2，所以一共有

$$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{4} \rfloor + .. + \lfloor \frac{n}{2^k} \rfloor$$

个 2，其中 $k = \lfloor log_2 n \rfloor$

更加一般的，对于质数 p，一共出现

$$\lfloor\frac{n}{p}\rfloor+\lfloor\frac{n}{p^2}\rfloor+..+\lfloor\frac{n}{p^k}\rfloor$$

个 p，其中 $k=\lfloor log_pn\rfloor$

这样处理的时间复杂度是 $O(log_pn)$

代码如下

```
int getNum(int p, int n)
{
    int num = 0, temp = p;
    while (p <= n) {
        num += n / temp;
        temp *= p
    }
    return num;
}
```

Miller-Rabin 素性检测（选讲）

输入大整数 n，进行 k 轮检测，判断 n 是否是素数，时间复杂度 $O(klogn)$

思路简述

对于给定的奇数 n，$n \geq 3$，将其分解成 $n-1=2^s \cdot t$，其中 t 为奇数

方程 $a^{n-1} \equiv 1\ (mod\ n)$ 等价于方程

$$(a^t-1)(a^t+1)(a^{2t}+1)\cdots(a^{2^{s-1}t}+1) \equiv 0\ (mod\ n)$$

等价于下面方程中**至少一个**成立

$$\begin{cases} a^t - 1 \equiv 0 \\ a^t + 1 \equiv 0 \\ a^{2t} + 1 \equiv 0 \\ \cdots \\ a^{2^{s-1}t} + 1 \equiv 0 \end{cases}$$

若上述方程均不成立，则说明 n 是合数，反之，则 n 以超过 $\frac{3}{4}$ 的概率是素数

选取不同整数 a，$2 \le a \le n-2$，进行 k 轮检测，便能够检验出 n 是否是素数

算法

- 进行 k 轮检测
- 随机生成 $2 \le a \le n-2$，利用快速幂计算 a^t，判断是否满足 $a^t \equiv \pm 1 \ (mod\ n)$，若满足，直接进入下一轮检测
- 计算 a^{2t}，判断是否满足 $a^{2t} \equiv -1 \ (mod\ n)$，若满足，直接进入下一轮检测
- ...
- 计算$a^{2^{s-1}t}$，判断是否满足 $a^{2^{s-1}t} \equiv -1 \ (mod\ n)$，若满足，直接进入下一轮检测，若不满足，说明 n 是合数，直接退出 $Miller\ \ Rabin$ 过程

时间复杂度为 $O(klog(n))$，代码见 notes.md

22. 质数生成

给定 n，生成 $1 \sim n$ 里面所有质数

保证 $1 \le n \le 100,000$

直接上欧拉线性筛或者埃氏筛

1437. 质数吗？

给定 n 个正整数 x，判断他们是不是质数

保证 $1 \leq n \leq 200,000, \ 1 \leq x \leq 1,000,000$

可以对每个质数进行 $O(\sqrt{x})$ 的判断，总的时间复杂度 $O(n\sqrt{x})$

也可以先线性筛出 $1,000,000$ 范围内的质数，总计 L 个，然后用哈希表或者二分法查询，前者时间复杂度 $O(n)$，后者时间复杂度 $O(n+nlog(L))$

闲得无聊也可以上米勒罗宾测试，代码量大一点而已

2586. 能否被整除呢?

给定 l_1，r_1，l_2，r_2，令 $a = \prod_{l_1}^{r_1}$，$b = \prod_{l_2}^{r_2}$，判断 a 能否整除 b

数据保证 $1 \leq l_1$，r_1，l_2，$r_2 \leq 10,000,000$

这类问题一般规约为求解质因子出现次数

先预处理 $1,000,000$ 范围内的质数，枚举质数 p_i，记区间 $[l_1,\ r_1]$ 和 $[l_2,\ r_2]$ 中 p_i 出现的次数分别为 $k_1,\ k_2$

若对于区间内任意质数 p_i，都满足 $k_2 \geq k_1$，那么说明 a 整除 b，反之不成立

注意数据很大，记得开 `long long int`