

AcWing 算法基础、提高课数学知识

廖涛

2022 年 4 月 15 日

目录

1	质数	5
1.1	判定质数	5
1.2	分解质因数	6
1.3	质数筛	7
1.3.1	埃氏筛法	7
1.3.2	欧拉筛法 (线形筛法)	7
2	约数	9
2.1	试除法求约数	9
2.2	约数个数	9
2.3	约数之和	11
2.4	最大公约数	11
3	欧拉函数	13
3.1	费马小定理	13
4	快速幂	15
4.1	快速幂原理	15
4.2	快速幂求逆元	15
5	扩展欧几里得算法 (粉碎机)	17
5.1	扩展欧几里得算法原理	17
5.2	扩展欧几里得算法求逆元	17
6	中国剩余定理 (孙子定理)	19

4	目 录
7 组合数	21
8 容斥原理	23
9 博弈论	25

Chapter 1

质数

$\forall n \in [(1, +\infty) \cap \mathbb{Z}^+]$, 若 n 只有 1 和 n 两个约数, 那么 n 为质数 (素数), 否则为合数。

0、1 以及负整数既不是质数也不是合数。

1.1 判定质数

```
1 bool is_prime(int n)
2 {
3     if(n < 2) return false;
4
5     for(int i = 2; i <= n / i; i++)
6     {
7         if(n % i == 0) return false;
8     }
9     return true;
10 }
```

1. 是否是大于等于 2 的整数, 2. 从 2 $n-1$ 中是否有 n 的约数。

不要写 $i * i < n$, 可能 $i * i$ 结果溢出 `int`; 也不要再在循环条件里写 $i <= \text{sqrt}(n)$, 否则每次循环都会调用 `sqrt()`。

$d | n$, 则必有 $\frac{n}{d} | n$, 故不需要枚举到 $n - 1$, 即枚举每组的约数的最小值即可。

时间复杂度为 $O(\sqrt{n})$

1.2 分解质因数

```

1 void divide(int n)
2 {
3     for(int i = 2; i <= n / i; i++)
4     {
5         int cnt = 0;
6         if(n % i == 0)
7         {
8             while(n % i == 0)
9             {
10                 n /= i;
11                 cnt++;
12             }
13             printf("%d %d\n", i, cnt);
14         }
15     }
16 }
17
18 if(n > 1) printf("%d 1\n", n);
19 puts("");
20 }
```

从 2 开始枚举到 \sqrt{n} ，每找到一个质因子则将 n 中的该质因子除尽，最后判断 n 是否大于 1，若大于 1，则此时的 n 也是所求质因子之一。

Tips1: 所有合数因子在被枚举到之前会被更小的质数因子除尽。故枚举到的合数必定不会满足 $n \bmod i = 0$ 。eg: $n = 24$, 8 这个合数因子必然会被之前出现的 2 除尽（除 3 次）。

Tips2: n 最多只存在 1 个大于等于 \sqrt{n} 的质因子，故只需要枚举到 \sqrt{n} 再判断最后的 n 情况如何即可。简单证明，假设存在

$$a \mid n, b \mid n, a > \sqrt{n}, b > \sqrt{n}$$

前两个条件可得 $a \times b \mid n$ ，最后一个条件可得 $a \times b > n$ ，相悖，故假设不成立，故 n 最多只存在 1 个大于等于 \sqrt{n} 的质因子。

Tips3: 当 $n = 2^k$ 时，第一次枚举即能把 n 除尽，此时的时间复杂度为 $\log_2 n$ ，最坏情况则是需要枚举到 \sqrt{n} ，故该方法的时间复杂度为 $O(\log_2 n)$ 与 $O(\sqrt{n})$ 之间。

1.3 质数筛

1.3.1 埃氏筛法

```

1 void get_primes(int n)
2 {
3     for(int i = 2; i <= n; i++)
4     {
5         if(st[i]) continue;
6
7         primes[cnt++] = i;
8         for(int j = 2 * i; j <= n; j += i)
9             {
10                st[j] = true;
11            }
12     }
13 }
```

从 2 枚举到 n ，即能找到 n 以内的所有质数。每次枚举到的 i 若 $st[i]$ 为 false 则说明 i 为质数，记录到 $primes[]$ 中，并筛掉其倍数。否则 i 为合数，直接跳过本次循环。

这里说明为什么不需要用合数筛掉其他数，当 $st[i]$ 为 true 时证明之前已经找到了 i 的一个或多个质因子， i 的倍数也必定是这些质因子的倍数，所以当 i 为合数时， i 的倍数已经在之前被其质因子筛掉，不需要再筛。

在枚举到 i 时， $2i-1$ 的所有数都已经枚举过，若 $st[i]$ 为 false 则说明 $2i-1$ 均不是 i 的因数，则 i 为质数（素数），否则为合数。

时间复杂度分析，当 $i = 2$ 时内层循环次数为 $\frac{n}{2}$ ，当 $i = 3$ 时为 $\frac{n}{3} \dots$ ，这里我们假设合数也进行内层循环，则内层循环的次数和为

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots + \frac{n}{n} = n \times \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} \right)$$

调和级数

$$\lim_{n \rightarrow \infty} \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = \ln n$$

故内层循环的次数小于等于 $n \ln n$ 小于 $n \log_2 n$ ，故可以认为埃氏筛法的时间复杂度为

$$O(n \log_2 n)$$

1.3.2 欧拉筛法 (线形筛法)

```
1 void get_primes(int n)
2 {
3     for(int i = 2; i <= n; i++)
4     {
5         if(!st[i]) primes[cnt++] = i;
6
7         for(int j = 0; primes[j] <= n / i; j++)
8         {
9             st[primes[j] * i] = true;
10            if(i % primes[j] == 0) break;
11        }
12    }
13 }
```

欧拉筛法的一个特点之一就是每个合数都是用其最小的质因子筛掉。

这里重点讲内层循环。首先无论 i 是否为质数, 都会进内层循环。 $primes[]$ 中小的数一定在前面。

当 $i \bmod primes[j] \neq 0$ 时, $primes[j]$ 的不是 i 的因子, 则 $primes[j] * i$ 的最小质因子就是 $primes[j]$ 。

当 $i \bmod primes[j] == 0$ 时, $primes[j]$ 是 i 的最小质因子, 也是 $primes[j] * i$ 的最小质因子。但当 $j++$ 后, $primes[j+1] * i$ 有更小的质因子 $primes[j]$ ($primes[j] \mid i$) 这里就不满足用最小质因子筛去合数的要求, 所以要 break。

Chapter 2

约数

2.1 试除法求约数

```
1 vector<int> get_divisors(int x)
2 {
3     vector<int> res;
4     for(int i = 1; i <= x / i; i++)
5     {
6         if(x % i == 0)
7         {
8             res.push_back(i);
9             if(i * i != x) res.push_back(x / i);
10        }
11    }
12    sort(res.begin(), res.end());
13    return res;
14 }
```

2.2 约数个数

若

$$n = p_1^{a_1} + p_2^{a_2} + p_3^{a_3} + \cdots + p_n^{a_n}$$

那么 n 的约数个数为

$$(a_1 + 1) * (a_2 + 1) * (a_3 + 1) * \cdots * (a_n + 1)$$

```

1  #include <iostream>
2  #include <algorithm>
3  #include <unordered_map>
4
5  using namespace std;
6
7  const int N = 110, mod = 1e9 + 7;
8  typedef long long LL;
9
10 int main()
11 {
12     int n;
13     cin >> n;
14     unordered_map<int, int> primes;
15
16     while(n --)
17     {
18         int x;
19         cin >> x;
20         for(int i = 2; i <= x / i; i ++){
21             {
22                 while(x % i == 0)
23                 {
24                     x /= i;
25                     primes[i] ++;
26                 }
27             }
28             if(x > 1) primes[x] ++;
29         }
30
31         LL res = 1;
32         for(auto t : primes)
33         {
34             LL p = t.first, s = t.second;
35             res = res * (s + 1) % mod;

```

```

36     }
37
38     cout << res << endl;
39
40     return 0;
41 }

```

2.3 约数之和

若

$$n = p_1^{a_1} + p_2^{a_2} + p_3^{a_3} + \cdots + p_n^{a_n}$$

那么 n 的约数之和为

$$(p_1^0 + p_1^1 + \cdots + p_1^{a_1}) * (p_2^0 + p_2^1 + \cdots + p_2^{a_2}) * (p_3^0 + p_3^1 + \cdots + p_3^{a_3}) * (p_n^0 + p_n^1 + \cdots + p_n^{a_n})$$

2.4 最大公约数

$(a / \gcd(a, b)) * (b / \gcd(a, b)) * \gcd(a, b)$ 即 $a / \gcd(a, b) * b$ 为 a 和 b 的最小公倍数。

```

1  int gcd(int a, int b)
2  {
3      return b ? gcd(b, a % b) : a;
4  }

```

Chapter 3

欧拉函数

n 的欧拉函数为 $[1, n-1]$ 内与 n 互质的数的个数。

若

$$n = p_1^{a_1} + p_2^{a_2} + p_3^{a_3} + \cdots + p_n^{a_n}$$

那么

$$\phi(n) = n * (1 - \frac{1}{p_1}) * (1 - \frac{1}{p_2}) * \cdots * (1 - \frac{1}{p_n}) = n * (\frac{p_1 - 1}{p_1}) * (\frac{p_2 - 1}{p_2}) * \cdots * (\frac{p_n - 1}{p_n})$$

当 n 为质数时

$$\phi(n) = n - 1$$

3.1 费马小定理

若 a 为整数, p 为质数, 那么

$$a^{\phi(p)} \equiv 1 \pmod{p}$$

即

$$a^{p-1} \equiv 1 \pmod{p}$$

Chapter 4

快速幂

4.1 快速幂原理

```
1 LL qmi(LL a, LL b, LL p)
2 {
3     LL res = 1;
4     while(b)
5     {
6         if(b & 1) res = res * a % p;
7         a = a * a % p;
8         b >>= 1;
9     }
10    return res;
11 }
```

4.2 快速幂求逆元

当模数 p 为质数时， a 的逆元即为 a^{p-2} ，故直接用快速幂计算即可。

```
1 qmi(a, p - 2, p);
```

Chapter 5

扩展欧几里得算法（粉碎机）

5.1 扩展欧几里得算法原理

5.2 扩展欧几里得算法求逆元

Chapter 6

中国剩余定理（孙子定理）

Chapter 7

组合数

Chapter 8

容斥原理

Chapter 9

博弈论