算法基础（二十八）： 数学基础 - 容斥原理

要计算几个集合并集的大小，我们要先将所有单个集合的大小计算出来，然后减去所有两个集合相交的部分，再加回所有三个集合相交的部分，再减去所有四个集合相交的部分.....依此类推，一直计算到所有集合相交的部分。（可以理解为就是先把所有单个集合全加一遍然后再去重）

`n`个集合的容斥原理的公式为：

$$\left|\bigcup_{i=1}^{n} A_i = \sum_{i}^{n} |A_i| - \sum_{i,j:1<=i<j<=n} \left|A_i \bigcap A_j\right| + \sum_{i,j,k:1<=i<j<k<=n} \left|A_i \bigcap A_j \bigcap A_k\right| \ldots + (-1)^{n-1} * \left|A_1 \bigcap \ldots \bigcap A_n\right|\right.$$

简单来说，就是奇数项是加上这个集合的元素，偶数项是减去这个集合的元素

先看看执行这个公式的时间复杂度：

我们首先得弄清楚公式中项的个数，一项的个数是C_n^1， 两项的个数就是C_n^2，三项的个数是C_n^3，从而总的项的个数就是：

$$C_n^1 + C_n^2 + C_n^3 + C_n^4 + \ldots . C_n^n$$

我们先看看以下这个公式：

$$C_n^0 + C_n^1 + C_n^2 + C_n^3 + C_n^4 + \ldots . C_n^n$$

这个公式代表的是从`n`个方案中挑选任意个方案组合的总数，相当于从`n`个元素中选择`0，1，2，……，n`个元素的所有情况，他等价于分步乘法原理，每个元素都有选或者不选两种情况，共2^n种情况，即：

$$C_n^0 + C_n^1 + C_n^2 + C_n^3 + C_n^4 + \ldots . C_n^n = 2^n$$

所以就有：

$$C_n^1 + C_n^2 + C_n^3 + C_n^4 + \ldots . C_n^n = 2^n - 1$$

于是容斥原理的公式共$2^n - 1$项，从而执行这个公式的时间复杂度就是2^n级别，`n`是集合的个数

下面来证明这个等式：

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i}^{n} |A_i| - \sum_{i,j:1<=i<j<=n} \left|A_i \bigcap A_j\right| + \sum_{i,j,k:1<=i<j<k<=n} \left|A_i \bigcap A_j \bigcap A_k\right| \ldots + (-1)^{n-1} * \left|A_1 \bigcap \ldots \bigcap A_n\right|$$

左边是所有元素的并集所含的元素的个数，如果任意一个元素在右边只被加了一次，那么右边的结果显然就是所有并集的元素总数了，我们假设有一个元素`a`在`k`个`Ai`集合中，也就是说有`k`个集合含有元素`a`，那么我们来证明在等式的右边这个元素只被加了一次

在右边的第一项，只有一个集合的情况下，这个元素被加了`k`次，也就是C_k^1，对于每一个单个集合，这个元素所贡献的个数都是`1`

在右边的第二项，两个集合取交集的情况下，交集中含有这个元素的个数是从`k`个集合中任意取出两个集合取交集的个数，即C_k^2

在右边的第三项，三个集合取交集的情况下，同理可得C_k^3

.....

在右边的第`k`项，`k`个集合取交集，这个元素只被加了一次，但是最后的负号取值是$(-1)^{k-1}$

当集合的个数大于k的时候显然，除了那k个集合之外还会有其他的集合，这时取交集，这个元素贡献的个数为0所有之后的情况不再考虑

所以对于这个元素来说，他在右边等式中贡献的个数为：

$$T = C_k^1 - C_k^2 + C_k^3 - \ldots . (-1)^{i-1} C_k^i + \ldots + (-1)^{k-1} C_k^k$$

由二项式定理，我们可以把他变成：

$$(1-x)^k = C_k^0 - C_k^1 x + C_k^2 x^2 - C_k^3 x^3 + \ldots + (-1)^k C_k^k x^k$$

令x = 1，从而得到：

$$T = C_k^1 - C_k^2 + C_k^3 - \ldots . (-1)^{i-1} C_k^i + \ldots + (-1)^{k-1} C_k^k = 1$$

所以，对于左边集合中的任意一个元素，他对于右边的个数贡献都是1，所以右边就代表左边所有集合并集的总的元素个数，证明完毕

容斥原理的一个简单应用：

给定一个整数 n 和 m 个不同的质数 $p_1, p_2, \ldots, p_m$。

请你求出 $1 \sim n$ 中能被 $p_1, p_2, \ldots, p_m$ 中的至少一个数整除的整数有多少个。

输入格式

第一行包含整数 n 和 m。

第二行包含 m 个质数。

输出格式

输出一个整数，表示满足条件的整数的个数。

数据范围

$1 \leq m \leq 16$,
$1 \leq n, p_i \leq 10^9$

输入样例：

```
10 2
2 3
```

输出样例：

```
7
```

这个题目朴素的想法是遍历每一个整数，然后看其能否被m个质数整除，时间复杂度就是mn级别，乘一下可以发现是16*10^9，但是C++每秒只能处理10^7 ~10^8的数据，这里显然超时，所以本题得换一种思路，使用容斥原理的方法来处理

令集合Ap表示1~n中是质数p的倍数的数，那么显然Ap中的数都满足条件，于是我们得到了集合Ai，i = p1 ~ pm，那么题目的结果就是这些所有的集合的并集的中元素的个数了，使用容斥原理即可求解，时间复杂度是2^m级别，这里显然在时间范围内

|Ai|的元素的个数就是n / i，由于每个质数都是互素的所以$|A_{pi} \bigcap A_{pj}|$的元素个数就是n / pi * pj

在计算元素个数的时候需要将k个元素乘在一起，然后计算个数，所以每一项的时间复杂度就是O(k)级别，这里k就是这一项的集合的个数，所以总的时间复杂度就是O(2 ^ m * m)，仍然在10^8以内

在遍历集合的时候我们使用位运算的方式

代码如下：

```cpp
#include<iostream>
#include<algorithm>
using namespace std;

typedef long long LL;
const int N = 20;

int n, m;
int p[N];

int main()
{
    cin >> n >> m;
    //读入m个质数
    for(int i = 0; i < m; i ++) cin >> p[i];

    //res表示最后的结果
    int res = 0;

    //容斥原理一共有2^m - 1项，循环遍历每一项
    //从1 ~ 2^m - 1每一个数的二进制位可以表示这一项中的集合交集交不交这个位代表的集合
    for(int i = 1; i < 1 << m; i ++)
    {
        //t表示当前所有质数的乘积
        //cnt表示当前i的二进制位中包含多少1
        int t = 1, cnt = 0;
        //i从右往左的第k位表示质数pk，i的第k位为1表示集合Ak被选上取交集
        for(int j = 0; j < m; j ++)
            if(i >> j & 1)
            {
                //如果为1表示需要交上这一位代表的集合
                //集合相交的个数++
                cnt ++;
                //开始将这些集合对应的质数乘起来
                if((LL) t * p[j] > n)
                {
                    //如果乘起来大于n很显然n中没有共同的乘积的倍数了，直接退出循环
                    t = -1;
                    break;
                }
                t *= p[j];
            }
        if(t != -1)
        {
            //如果t不为-1，说明这些质数的乘积是小于n，可以计算对应的所有交集的元素个数
            if(cnt % 2) res += n / t;
            else res -= n / t;
        }
    }

    cout << res <<endl;

```

```
53      return 0;
54  }
```

我们再次看看下面这个公式：

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i}^{n} |A_i| - \sum_{i,j:1<=i<j<=n} \left|A_i \bigcap A_j\right| + \sum_{i,j,k:1<=i<j<k<=n} \left|A_i \bigcap A_j \bigcap A_k\right| \ldots + (-1)^{n-1} * \left|A_1 \bigcap \ldots \bigcap A_n\right|$$

等式的右边一共有`2^n - 1`项，总项数代表的含义是`n`个集合，取任意集合一起相交的个数，也就是这`n`个集合组成的一个大集合，其非空子集的个数，也可以看作是`n`位二进制数每位取`0`或者`1`可以构成的非`0`数的个数，所以对于每一项，可以看作对应一个二进制数，这一项中包含集合`i，j，k`可以看作，这个二进制数中的第`i，j， k`位为`1`，其符号可以看作是二进制中`1`的个数减一作为`-1`的幂，二进制数从`1`到`2^n - 1`刚好可以对应上面等式右边的所有的项，所以我们可以用位运算来解决取质数，以及将质数乘积。