容斥原理及其应用

张渊昌
西北工业大学ACM基地

2015 年 2 月 14 日

1 容斥原理

1.1 例1：计数1,2,3,4,...,n的排列$i_1, i_2, i_3, ..., i_n$,中1不在第一个位置上的那些排列的数目

解：方法一：加法原理，第一个位置上可以放2,3,...,n,有n-1种选择，剩下的n-1个数字放在n-1个位置上共有 $(n-1)!$种方式，所以总共有（n-1)*$(n-1)!$种方式。

方法二：减法原理，1在第一个位置上的排列数总共有$(n-1)!$中，所以总共有$n!$-$(n-1)!$=(n-1)*$(n-1)!$种方式。

1.2 例2：计数1到600之间不能被6整除的整数个数。

解：减法原理，每连续的6个数的第六个数都能被6整除，所以1到600之间总共有100个数能被6整除，不能被6 整除的数的个数为600-100=500个。

减法原理是容斥原理的最简单示例。我们将用一种便于应用的方式陈述容斥原理。

设S是对象的有限集合，设P_1,P_2是S中每一个集合有或者没有的两个"性质"。我们希望计数集合S 中既不具有性质P_1 又不具有性质P_2的对象个数。扩展减法原理其内在的理由，我们就可以完成这个计数：首先计数集合S 中的所有对象，然后排除具有性质P_1的所有对象，再排除具有性质P_2的所有对象，注意此时我们已经把具有性质P_1和P_2的对象派出了两次，因此，需要把这样的对象在重新加入一次。

这样的性质用公式表示为:

$$|\bar{A}_1 \cap \bar{A}_2| = |S| - |A_1| - |A_2| + |A_1 \cap A_2|$$

1.3 定理1：集合S中不具有性质P_1,P_2,...,P_m的对象个数由下面的交错表达式给出：

$$|\bar{A}_1 \cap \bar{A}_2 \cap ...\bar{A}_m| = |S| - \sum |A_i| + \sum |A_i \cap A_j| - \sum |A_i \cap A_j \cap A_k| + ... + (-1)^m |A_1 \cap A_2 \cap A_3 \cap ... \cap A_m|.$$

1.4 定理2：集合S中至少具有性质$P_1, P_2, ..., P_m$之一的对象个数由下式给出：

$|A_1 \cup A_2 \cup ... \cup A_m|$=$\sum|A_i|$-$\sum|A_i \cap A_j|$+$\sum|A_i \cap A_j \cap A_k|$ -...+$(-1)^{m+1}|A_1 \cap A_2 \cap ... \cap A_m|$.

证明：$|A_1 \cup A_2 \cup ... \cup A_m|$=$|S|$-$|\overline{A_1 \cup A_2 \cup ... \cup A_m}|$

由德.摩根法则：

$\overline{A_1 \cup A_2 \cup ... \cup A_m}$=$\bar{A}_1 \cap \bar{A}_2 \cap \bar{A}_3 \cap ... \cap \bar{A}_m$

所以有：$|A_1 \cup A_2 \cup ... \cup A_m|$=$|S|$-$|\bar{A}_1 \cap \bar{A}_2 \cap ... \cap \bar{A}_m|$

1.5 例3：求1到1000之间不能被5，6和8整除的整数个数。

解：设P_1表示能被5整除的性质，P_2表示能被6整除的性质，P_3表示能被8整除的性质。S是有前1000个正整数组成的集合。对于i=1，2，3，设A_i是S中那些具有性质P_i的整数组成的集合。求出$\bar{A}_1 \cap \bar{A}_2 \cap \bar{A}_3$中的整数个数。

$|A_1|$=$\lfloor \frac{1000}{5} \rfloor$=200;

$|A_2|$=$\lfloor \frac{1000}{6} \rfloor$=166;

$|A_3|$=$\lfloor \frac{1000}{8} \rfloor$=125;

$|A_1 \cap A_2|$=$\lfloor \frac{1000}{30} \rfloor$=33

$|A_1 \cap A_3|$=$\lfloor \frac{1000}{40} \rfloor$=25

$|A_3 \cap A_2|$=$\lfloor \frac{1000}{24} \rfloor$=41

$|A_1 \cap A_2 \cap A_3|$=$\lfloor \frac{1000}{120} \rfloor$=8

因此：$\bar{A}_1 \cap \bar{A}_2 \cap \bar{A}_3$=1000-(200+166+125)+(33+25+41)-8=600

2 带重复的组合

2.1 基本概念

1.n个不同元素的集合的r子集的数目为：$\binom{n}{r}$ =$\frac{n!}{r!*(n-r)!}$

2.k种不同对象且每种对象都有无限重数的多重集合的r组合的个数等于：$\binom{r+k-1}{r}$

3.设T是多重集合，而x是T中某种类型的对象，其重数大于r。T的r组

合数目等于这样一个多重集合的r组合数目：即把T 的x重数换成r而得到的多重集合。

2.2 例1：确定多重集合$\{3*a,4*b,5*c\}$的10组合的数目

把容斥原理运用到多重集合T^*=$\{\infty*a,\infty*b,\infty*c\}$的所有10组合的集合S 上。设$P_1$ 是T^* 的10组合中a出现多于3次的性质,P_2是T^*的10组合中b出现多于4次的性质,P_2是T^*的10 组合中c 出现多于5次的性质。此时,T的10组合数就是T^*的10组合数中不具有性质P_1，P_2，P_3的10 组合的数目。同样，设A_i是由T^* 的10组合当中具有性质P_i的那些10组合组成。确定集合$\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3$的大小。

根据容斥原理：

$|\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3|$=$|S|$-($|A_1|$+$|A_2|$+$|A_3|$)+($|A_1\cap A_2|$+$|A_2\cap A_3|$+$|A_1\cap A_3|$)-$|A_1\cap A_2\cap A_3|$

根据定理：

$|S|$=$\binom{10+3-1}{10}$=66

$|A_1|$=$\binom{6+3-1}{6}$=28

$|A_2|$=$\binom{5+3-1}{5}$=21

$|A_3|$=$\binom{4+3-1}{4}$=15

$|A_1\cap A_2|$=$\binom{1+3-1}{1}$=3

$|A_1\cap A_3|$=$\binom{0+3-1}{0}$=1

$|A_2\cap A_3|$=0

$|A_1\cap A_2\cap A_3|$=0

将所有的结果放到容斥原理中，得到：

$|\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3|$=66-(28+21+15)+(3+1+0)-0=6;

2.3 例2：满足1⩽x_1⩽5,-2⩽x_2⩽4,0⩽x_3⩽5,3⩽x_4⩽9 的方程$x_1+x_2+x_3+x_4=18$的整数解的数目是多少？

解：引入一些新的变量：y_1=x_1-1,y_2=x_2+2,y_3=x_3,y_4=x_4-3

这样方程变为：y_1+y_2+y_3+y_4=18

关于x_i的不等式成立当且仅当

0⩽y_1⩽4,0⩽y_2⩽6,0⩽y_3⩽5,0⩽y_4⩽6

设S是方程的所有非负整数解的集合。S的大小为

$|S|=\binom{16+4-1}{16}=969$

设P_1是$y_1\geqslant 5$的性质,设P_2是$y_2\geqslant 7$的性质,设P_3是$y_3\geqslant 6$的性质,设P_4是$y_4\geqslant 7$的性质。设A_i表示S中满足性质P_i的解组成的子集。现在要计算集合$\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3\cap\bar{A}_4$的大小，集合$A_1$由S中满足$y_1\geqslant 5$的解组成。做变量代换($z_1=y_1$-5,$z_2=y_2$,$z_3$,$y_3$,$z_4=y_4$),我们可以看到，$A_1$的解得个数与 $z_1+z_2+z_3+z_4=11$ 的非负整数解的个数相同。因此：$|A_1|=\binom{14}{11}=364$

以类似的方式得到

$|A_2|=\binom{12}{9}=220$, $|A_3|=\binom{13}{10}=286$, $|A_4|=\binom{12}{9}=220$

集合$A_1\cap A_2$是由S中满足$|y_1|\geqslant 5$和$|y_2|\geqslant 7$的那些解组成的。进行变量代换($|u_1|=|y_1|$-5, $|u_2|=|y_2|$-7,$|u_3|=|y_3|$,$|u_4|=|y_4|$),我们看出$A_1\cap A_2$的解得个数与$|u_1|+|u_2|+|u_3|+|u_4|=4$的非负整数解的个数相同。因此:

$|A_1\cap A_2|=\binom{7}{4}=35$

以类似的方式得到:

$|A_1\cap A_3|=\binom{8}{5}=56$, $|A_1\cap A_4|=\binom{7}{4}=35$

$|A_2\cap A_3|=\binom{6}{3}=20$, $|A_2\cap A_4|=\binom{5}{2}=10$

$|A_3\cap A_4|=\binom{6}{3}=20$

集合A_1，A_2，A_3，A_4中任意三个的交集都是空集。

应用容斥原理得到:

$\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3\cap\bar{A}_4$=969-(364+220+286+220)+(35+56+35+20+10+20)=55

3 POJ 1173

Bar Codes Time Limit: 1000MS Memory Limit: 10000K Total Submissions: 893 Accepted: 317 Description

A bar-code symbol consists of alternating dark and light bars, starting with a dark bar on the left. Each bar is a number of units wide. Figure 1 shows a bar-code symbol consisting of 4 bars that extend over 1+2+3+1=7 units.

Figure 1: Bar-code symbol over 7 units (see top) with 4 bars (see bottom)

In general, the bar code BC(n,k,m) is the set of all symbols with k bars that together extend over exactly n units, each bar being at most m units wide. For instance, the symbol in Figure 1 belongs to BC(7,4,3) but not to BC(7,4,2).

0: 1000100 — 8: 1100100

1: 1000110 — 9: 1100110

2: 1001000 — 10: 1101000

3: 1001100 — 11: 1101100

4: 1001110 — 12: 1101110

5: 1011000 — 13: 1110010

6: 1011100 — 14: 1110100

7: 1100010 — 15: 1110110

Figure 2: All symbols of BC(7,4,3)

Figure 2 shows all 16 symbols in BC(7,4,3). Each '1' represents a dark unit, each '0' a light unit. The symbols appear in lexicographic (dictionary) order. The number on the left of the colon (':') is the rank of the symbol. The symbol in Figure 1 has rank 4 in BC(7,4,3). Input

Your program is to read from standard input. The first line contains the numbers n, k, and m (1 ¡= n,k,m ¡= 33). On the second line is a number s (0 ¡= s ¡= 100). The following s lines each contain some symbol in BC(n,k,m), represented by '0's and '1's as in Figure 2. Output

Your program is to write to standard output. On the first line your program should write the total number of symbols in BC(n,k,m). On each of the s following lines, it should write the rank of the corresponding symbol in the input. Sample Input

7 4 3 5 1001110 1110110 1001100 1001110 1000100 Sample Output

16 4 15 3 4 0

3.1 题目大意

输入 n（代表二进制位数） k（代表黑条白条总共有几条，条形码是以黑条开始的，再白黑交替出现） m（代表每条最多占多少个二进制位） 输出这种模式的条形码的有多少个?

输入s，再输入s个二进制形式的条形码 输出每个条形码在该模式中的序号，序号是根据二进制条形码的十进制数值排序，序号从0开始。

3.2 解题思路

动态规划+组合数学。 我们举一个例子：n=7，k=4，m=3。

1.计算个数 要计算此模式条形码的数量，那么我们只要分别计算出n=6，k=3，m=3。 n=5，k=3，m=3。 n=4，k=3 ，m=3。 再讲他们相加即可。 扩展到一般，得公式 [n，k] =∑[n-i，k-1]（i = 1，2・・・m-1，m）根据推算，我们可以将公式化简成 [n,k] = [n-1,k] + [n-1,k-1] - [n-m-1,k-1] 我们令[0,0]=1, 令 [0,1]至[0,k] 和 [1,0]至[k,0] =0，其余的值都可以通过递推得到

2.计算序号 首先将 长度为n的二进制条形码 转换成 长度为k的向量，例如 1101110 -》 2131（2个1，1个0，3个1，1个0） 2131之前的条形码可以分为四部分：

1??? [6,3] = 7

22?? [3,2] = 1

23?? [2,2] = 2

211? [3,1] = 1

212? [2,1] = 1

合计为12 对照下面题目给出的表 确是如此 至于究竟上面是如何弄出来的，需要分黑条和白条分别考虑，不是很好说明，看代码应该能懂。 还有一点，我们都知道0在二进制位里面越前，1越后，则该数值会越小（即序号越小）。

0: 1000100 — 8: 1100100

1: 1000110 — 9: 1100110

2: 1001000 — 10: 1101000

3: 1001100 — 11: 1101100

4: 1001110 — 12: 1101110

5: 1011000 — 13: 1110010

6: 1011100 — 14: 1110100

7: 1100010 — 15: 1110110

3.3 代码

```
#include <iostream>
```

```
using namespace std;

int n,k,m;
int dp[40][40]={0};
int s;
char bin[40];
int vec[40];
int count[102];

void table()
{
    dp[0][0]=1;
    for(int j=1;j<=k;j++)
        for(int i=1;i<=n;i++)
        {
            dp[i][j]=dp[i-1][j]+dp[i-1][j-1];
            if(i-m-1>=0)
                dp[i][j]-=dp[i-m-1][j-1];
        }
}

void binToVec()
{
    int c=0;
    char first;
    for(int i=0,j=0;i<n;i--)
    {
        c=1;
        first=bin[i++];
        while(bin[i++]==first)
        {
            c++;
        }
        vec[j++]=c;
    }
}

int countOrder()
{
    int count = 0;
    for(int i=0,u=n;i<k-1;i++)
    {
        if(i%2==0)
        {
            for(int j=1;j<vec[i];j++)
            {
                if(u>=j)
```

```
                    count+=dp[u-j][k-i-1];
            }
        }
        else
        {
            for(int j=m;j>vec[i];j--)
            {
                if(u>=j)
                    count+=dp[u-j][k-i-1];
            }
        }
        u-=vec[i];
    }
    return count;
}

int main()
{
    cin>>n>>k>>m;
    table();
    cin>>s;
    for(int i=0;i<s;i++)
    {
        cin>>bin;
        binToVec();
        count[i] = countOrder();
    }
    cout<<dp[n][k]<<endl;
    for(int i=0;i<s;i++)
    {
        cout<<count[i]<<endl;
    }
}
```

4 POJ 2773

Happy 2006 Time Limit: 3000MS Memory Limit: 65536K Total Submissions: 9974 Accepted: 3426 Description

Two positive integers are said to be relatively prime to each other if the Great Common Divisor (GCD) is 1. For instance, 1, 3, 5, 7, 9...are all relatively prime to 2006.

Now your job is easy: for the given integer m, find the K-th element which is relatively prime to m when these elements are sorted in ascending order. Input

The input contains multiple test cases. For each test case, it contains two integers m (1 ¡= m ¡= 1000000), K (1 ¡= K ¡= 100000000). Output

Output the K-th element in a single line. Sample Input

2006 1 2006 2 2006 3 Sample Output

1 3 5 Source

POJ Monthly–2006.03.26,static

4.1 题目大意

题目是给出m，k。找到跟第k个跟m互素的数是多少。

4.2 解题思维

思路是二分枚举[1,2^{64}]范围内所有的数x，找到1到x范围内与m不互素的数的个数y（用容斥原理）。然后用x - y，如果等于k就是结果。

找到1到x范围内与m不互素的数的个数y：这个过程可以先把m分解质因子，记录m所有的质因子。f[i]表示含有i个质因子的数的个数。ans = m - f(1) + f(2) - f(3) ....

4.3 代码

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <vector>
#include <cstring>
#include <algorithm>

typedef long long LL;
const double eps = 1e-8;
const double pi = acos(-1.0);
const double inf = ~0u>>2;


using namespace std;

const int N = 1000010;

int p[N], cnt;
```

```
void init(int m) {
    cnt = 0;
    for(int i = 2; i*i <= m; ++i) {
        if(m%i == 0) {
            p[cnt++] = i;
            while(m%i == 0) {m /= i;}
        }
    }
    if(m != 1)  p[cnt++] = m;
}

LL cal(LL n) {
    int i, j, bit;
    LL res = 0, sum;
    for(i = 1; i < 1<<cnt; ++i) {
        bit = 0; sum = 1;
        for(j = 0; j < cnt; ++j) {
            if(i&(1<<j)) {
                bit++;
                sum *= p[j];
            }
        }
        if(bit&1)   res -= n/sum;
        else    res += n/sum;
    }
    return n + res;
}

int main() {
    int m, k;
    while(cin >> m >> k, !cin.eof()) {
        init(m);
        LL l = 1, r = (1LL<<60), mid, tmp;
        while(r - l > 0) {
            mid = MID(l, r);
            tmp = cal(mid);
            if(tmp >= k)    r = mid;
            else    l = mid + 1;
        }
        printf("%lld\n", l);
    }
    return 0;
}
```