# 容斥原理


原作：e-maxx(Russia)

翻译：vici@cust


2011年8月25日

## 译者序

这篇文章发表于http://e-maxx.ru/algo/inclusion_exclusion_principle，原文是俄语的。由于文章确实很实用，而且鉴于国内俄文资料翻译的匮乏，我下决心将其翻译之。但是俄语对我来说如同乱码，而用Google直接翻译中文的话又变得面目全非，所以只能先用Google翻译成英语，再反复读，慢慢理解英语的意思，实在是弄得我头昏脑胀。因此在理解文章意思然后翻译成中文的时候，中文都不知道如何表述了。而又由于我对容斥原理知识的匮乏，很可能有些地方我的表述是错误的。而原文中也有一些错误，我（自作聪明？）将它们进行了一些改动和注释，当然我都保留了原文。如果你对这篇文章有什么不理解的地方，可以去网站论坛的Feedback版(http://e-maxx.ru/forum/viewforum.php?id=6)发问。不过这可是俄语的，所以直接问我吧:)

（QQ：573525822，E-mail：573525822@qq.com或veecci@gmail.com）

# 目录

容斥原理是一种重要的组合数学方法，可以让你求解任意大小的集合，或者计算复合事件的概率。

# 1 对容斥原理的描述

## 1.1 描述

对容斥原理可以描述如下：

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**。

## 1.2 关于集合的原理

上述描述的公式形式可以表示如下：

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

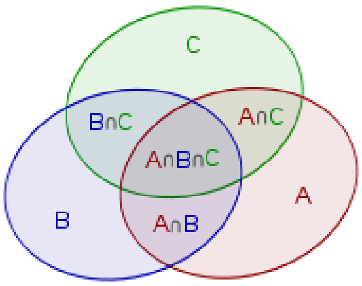
它可以写得更简洁一些，我们用$B$代表所有$A_i$的集合，那么容斥原理就变成了：

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{C\subseteq B} (-1)^{size(C)-1} \left|\bigcap_{e\in C} e\right| .$$

这个公式是由De Moivre (Abraham de Moivre)提出的。

## 1.3 关于维恩图的原理

用维恩图来表示集合A、B和C：




那么$A\cup B\cup C$的面积就是集合A、B、C各自面积之和减去$A\cap B$, $A\cap C$, $B\cap C$的面积，再加上$A\cap B\cap C$的面积。

$$S(A\cup B\cup C)=S(A)+S(B)+S(C)-S(A\cap B)-S(A\cap C)-S(B\cap C)+S(A\cap B\cap C).$$

由此，我们也可以解决n个集合求并的问题。

### 1.4 关于概率论的原理

设事件$A_i(i=1\ldots n)$, $P(A_i)$代表发生某些事件的概率（即发生其中至少一个事件的概率），则：

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

这个公式也可以用$B$代表$A_i$的集合：

$$P\left(\bigcup_{i=1}^{n}A_i\right)=\sum_{C\subseteq B}(-1)^{size(C)-1}\cdot P\left(\bigcap_{e\in C}e\right).$$

## 2 对容斥原理的证明

我们要证明下面的等式：

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{C\subseteq B} (-1)^{size(C)-1} \left|\bigcap_{e\in C} e\right|.$$

其中$B$代表全部$A_i$的集合。

我们需要证明在$A_i$集合中的任意元素，都由右边的算式被正好加上了一次（注意如果是不在$A_i$集合中的元素，是不会出现在右边的算式中的)。

假设有一任意元素在$k$个$A_i$集合中（$k \geq 1$)，我们来验证这个元素正好被加了一次：

- 当$size(C) = 1$时，元素$x$被加了$k$次。
- 当$size(C) = 2$时，元素$x$被减了$C_k^2$次，因为在$k$个集合中选择2个，其中都包含$x$。
- 当$size(C) = 3$时，元素$x$被加了$C_k^3$次。
- ...
- 当$size(C) = k$时，元素x被加/减了$C_k^k$次，符号由$sign(-1)^{(k-1)}$决定。
- 当$size(C) > k$时，元素x不被考虑。

然后我们来计算所有组合数的和。

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

由二项式定理，我们可以将它变成$(1-x)^k$:

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

我们把x取为1，这时$(1-x)^k$表示$1-T$（其中T为x被加的总次数），所以$T=1-(1-1)^k=1$，证明完毕。

# 3 容斥原理的应用

容斥原理的理论需要通过例子才能很好的理解。

首先，我们用三个简单的例子来阐释这个理论。然后会讨论一些复杂问题，试看如何用容斥原理来解决它们。

其中的"寻找路径数"是一个特殊的例子，它反映了容斥问题有时可以在多项式级复杂度内解决，不一定需要指数级。

## 3.1 一个简单的排列问题

由0到9的数字组成排列，要求第一个数大于1，最后一个数小于8，一共有多少种排列？

我们可以来计算它的逆问题，即第一个元素$\leq 1$或者最后一个元素$\geq 8$的情况。

我们设第一个元素$\leq 1$时有X组排列，最后一个元素$\geq 8$时有Y组排列。那么通过容斥原理来解决就可以写成：

$$|X|+|Y|-|X\cap Y|$$

经过简单的组合运算，我们得到了结果：

$$2\cdot 9!+2\cdot 9!-2\cdot 2\cdot 8!$$

然后被总的排列数10!减，就是最终的答案了。

## 3.2 （0,1,2）序列问题

长度为$n$的由数字0，1，2组成的序列，要求每个数字至少出现1次，这样的序列有

多少种？

同样的，我们转向它的逆问题。也就是不出现这些数字的序列。

我们定义$A_i\,(i=0\dots 2)$表示不出现数字$i$的序列数，那么由容斥原理，我们得到该逆问题的结果为：

$$|A_0|+|A_1|+|A_2|-|A_0\cap A_1|-|A_0\cap A_2|-|A_1\cap A_2|+|A_0\cap A_1\cap A_2|$$

可以发现每个$A_i$的值都为$2^n$（因为这些序列中只能包含两种数字）。而所有的两两组合$A_i\cap A_j$ 都为1（它们只包含1种数字）。最后，三个集合的交集为0。(因为它不包含数字，所以不存在)

要记得我们解决的是它的逆问题，所以要用总数减掉，得到最终结果：

$$3^n-3\cdot 2^n+3\cdot 1-0.$$

## 3.3 求方程整数解的个数

给出一个方程：

$$x_1+x_2+x_3+x_4+x_5+x_6=20.$$

其中$0\le x_i\le 8$。

求这个方程的整数解有多少组。

我们先不去理会$x_i\le 8$的条件，来考虑所有正整数解的情况。这个很容易用组合数来求解，我们要把20个元素分成6组，也就是添加5块”夹板”，然后在25个位置中找5块”夹板”的位置。

$$N_0=C_{25}^5$$

然后通过容斥原理来讨论它的逆问题，也就是$x_i\ge 9$时的解。

我们定义$A_k$为$x_k \geq 9$并且其他$x_i \geq 0$时的集合，同样我们用上面的添加"夹板"法来计算$A_k$的大小，因为有9个位置已经被$x_k$所利用了，所以：

$$|A_k| = C_{16}^5$$

然后计算两个这样的集合$A_k$、$A_p$的交集：

$$|A_k \cap A_p| = C_7^5$$

因为所有$x$的和不能超过20，所以三个或三个以上这样的集合时是不能同时出现的，它们的交集都为0。最后我们用总数剪掉用容斥原理所求逆问题的答案，就得到了最终结果：

$$C_{25}^5 - C_6^1 \cdot C_{16}^6 + C_6^2 \cdot C_7^5.$$

（译注：这里似有错误，$C_{16}^6$应为$C_{16}^5$）

## 3.4 求指定区间内与n互素的数的个数

给出整数$n$和$r$。求区间$[1;r]$中与$n$互素的数的个数。

去解决它的逆问题，求不与$n$互素的数的个数。

考虑$n$的所有素因子$p_i\ (i = 1 \ldots k)$

在$[1;r]$中有多少数能被$p_i$整除呢？它就是：

$$\left\lfloor \frac{r}{p_i} \right\rfloor$$

然而，如果我们单纯将所有结果相加，会得到错误答案。有些数可能被统计多次（被好几个素因子整除）。所以，我们要运用容斥原理来解决。

我们可以用$2^k$的算法求出所有的$p_i$组合，然后计算每种组合的$p_i$乘积，通过容斥原

理来对结果进行加减处理。

**程序实现：**

```
int solve (int n, int r) {
	vector<int> p;
	for (int i=2; i*i<=n; ++i)
		if (n % i == 0) {
			p.push_back (i);
			while (n % i == 0)
				n /= i;
		}
	if (n > 1)
		p.push_back (n);
	int sum = 0;
	for (int msk=1; msk<(1<<p.size()); ++msk) {
		int mult = 1,
			bits = 0;
		for (int i=0; i<(int)p.size(); ++i)
			if (msk & (1<<i)) {
				++bits;
				mult *= p[i];
			}

		int cur = r / mult;
		if (bits % 2 == 1)
			sum += cur;
		else
			sum -= cur;
	}

	return r - sum;
}
```

算法的复杂度为$O\left(\sqrt{n}\right)$。

## 3.5 求在给定区间内，能被给定集合至少一个数整除的数的个数

给出$n$个整数$a_i$和整数$r$。求在区间$[1;r]$中，至少能被一个$a_i$整除的数有多少。

解决此题的思路和上题差不多，计算$a_i$所能组成的各种集合（这里将集合中$a_i$的最小公倍数作为除数）在区间中满足的数的个数，然后利用容斥原理实现加减。

此题中实现所有集合的枚举，需要$2^n$的复杂度，求解Lcm需要$O\left(n\log r\right)$的复杂度。

## 3.6 求能满足一定数目匹配的字符串的个数

给出$n$个匹配串，它们长度相同，其中有一些'?'表示待匹配的字母。然后给出一个整数$k$，求能正好匹配$k$个匹配串的字符串的个数。更进一步，求至少匹配$k$个匹配串的字符串的个数。

首先我们会发现，我们很容易找到能匹配所有匹配串的字符串。只需要对比所有匹配串，去在每一列中找出现的字母（或者这一列全是'?'，或者这一列出现了唯一的字母，否则这样的字符串就存在），最后所有字母组成的单词即为所求。

现在我们来学习如何解决**第一个问题**：能正好匹配$k$个匹配串的字符串。

我们在$n$个匹配串中选出$k$个，作为集合$X$，统计满足集合$X$中匹配的字符串数。求解这个问题时应用容斥原理，对$X$的所有超集进行运算，得到每个$X$集合的结果：

$$ans\left(X\right)=\sum_{Y\supseteq X}\left(-1\right)^{|Y|-k}\cdot f\left(Y\right)$$

此处$f\left(Y\right)$代表满足匹配集合$Y$的字符串数。

如果我们将所有的$ans\left(X\right)$相加，就可以得到最终结果：

$$ans=\sum_{X:|X|=k}ans\left(X\right).$$

这样，就得到了一个复杂度$O\left(3^k\cdot k\right)$的解法。

这个算法可以作一些改进，因为在求解$ans\left(X\right)$时有些$Y$集合是重复的。

回到利用容斥原理公式可以发现，当选定一个Y时，所有$C_{|Y|}^{k}$中$X$的结果都是相同的，其符号都为$(-1)^{|Y|-k}$。所以可以用如下公式求解：

$$ans = \sum_{Y:|Y|\geq k} (-1)^{|Y|-k} \cdot C_{|Y|}^{k} \cdot f\left(Y\right).$$

这样就得到了一个复杂度$O\left(2^{k} \cdot k\right)$的解法。

现在我们来求解**第二个问题**：能满足至少$k$个匹配的字符串有多少个。

显然的，我们可以用问题一的方法来计算满足$k$到$n$的所有结果。问题一的结论依然成立，不同之处在于这个问题中的$X$不是大小都为$k$的，而是$\geq k$的所有集合。

如此进行计算，最后将$f\left(Y\right)$作为另一个因子：将所有的$ans$作和，有点类似二项式展开：

$$(-1)^{|Y|-k} \cdot C_{|Y|}^{k} + (-1)^{|Y|-k-1} \cdot C_{|Y|}^{k+1} + (-1)^{|Y|-k-2} \cdot C_{|Y|}^{k+2} + ... + (-1)^{|Y|-|Y|} \cdot C_{|Y|}^{|Y|}.$$

在《具体数学》(Graham, Knuth, Patashnik. "Concrete Mathematics" [1998])中，介绍了一个著名的关于二项式系数的公式：

$$\sum_{k=0}^{m} (-1)^{k} \cdot C_{n}^{k} = (-1)^{m} \cdot C_{n-1}^{k}.$$

根据这个公式，可以将前面的结果进行化简：

$$(-1)^{|Y|-k} \cdot C_{|Y|-1}^{|Y|-k}.$$

那么，对于这个问题，我们也得到了一个$O\left(2^{k} \cdot k\right)$的解法：

$$ans = \sum_{Y:|Y|\geq k} (-1)^{|Y|-k} \cdot C_{|Y|-1}^{|Y|-k} \cdot f\left(Y\right).$$

## 3.7 路径的数目问题

在一个$n \times m$的方格阵中，有$k$个格子是不可穿越的墙。一开始在格子$(1,1)$（最左下角的格子）中有一个机器人。这个机器人只能向上或向右行进，最后它将到达位于格子$(n,m)$的笼子里，其间不能经过障碍物格子。求一共有多少种路线可以到达终点。

为了方便区分所有障碍物格子，我们建立坐标系，用$(x,y)$表示格子的坐标。

首先我们考虑没有障碍物的时候：也就是如何求从一个点到另一个点的路径数。如果从一个点在一个方向要走$x$个格子，在另一个方向要走$y$个格子，那么通过简单的组合原理可以得知结果为：

$$C_{x+y}^{x}$$

现在来考虑有障碍物时的情况，我们可以利用容斥原理：求出至少经过一个障碍物时的路径数。

对于这个例子，你可以枚举所有障碍物的子集，作为需要要经过的，计算经过该集合障碍物的路径数（求从原点到第一个障碍物的路径数、第一个障碍物到第二个障碍物的路径数...最后对这些路径数求乘积），然后通过容斥原理，对这些结果作加法或减法。

然而，它是一个**非多项式的解法**，复杂度$O\left(2^k \cdot k\right)$。下面我们将介绍一个**多项式的解法**。

我们运用**动态规划**：令$d[i][j]$代表从第$i$个点到第$j$个点，不经过任何障碍物时的路径数（当然除了$i$和$j$）。那么我们总共需要$k+2$个点，包括$k$个障碍物点以及起点和终点。

首先我们算出从$i$点到$j$点的所有路径数，然后减掉经过障碍物的那些"坏"的路线。让我们看看如何计算"坏"的路线：枚举$i$和$j$之间的所有障碍物点$i < l < j$，那么从$i$到$j$的"坏"路径数就是所有$d[i][l]$和$d[l][j]$的乘积最后求和。再被总路径数减掉就

是$d[i][j]$的结果。

我们已经知道计算总路径数的复杂度为$O(k)$，那么该解法的总复杂度为$O\left(k^3\right)$。

（译注：当然也有$O(nm)$的$DP$解法，根据$n,m,k$的值可以采取适当的解法）

## 3.8 求素数四元组的个数

给出$n$个数$a_1,a_2,\ldots,a_n$，从其中选出4个数，使它们的最大公约数为1，问总共有多少种取法。

我们解决它的逆问题：求最大公约数$d>1$的四元组的个数。

运用容斥原理，将求得的对于每个$d$的四元组个数的结果进行加减。

$$ans=\sum_{d\geq 2}(-1)^{\deg(d)-1}\cdot f(d).$$

其中$\deg(d)$代表$d$的质因子个数，$f(d)$代表四个数都能被$d$整除的四元组的个数。

求解$f(d)$时，只需要利用组合方法，求从所有满足被$d$整除的$a_i$中选4个的方法数。

然后利用容斥原理，统计出所有能被一个素数整除的四元组个数，然后减掉所有能被两个素数整除的四元组个数，再加上被三个素数整除的四元组个数…

## 3.9 求和睦数三元组的个数

给出一个整数$n\leq 10^6$。选出$a,b,c(2\leq a<b<c\leq n)$，组成**和睦三元组**，即：

- 或者满足$\gcd(a,b)>1$，$\gcd(a,c)>1$，$\gcd(b,c)>1$
- 或者满足$\gcd(a,b)=\gcd(a,c)=\gcd(b,c)=1$

首先，我们考虑它的逆问题：也就是**不和睦三元组**的个数。

然后，我们可以发现，在每个不和睦三元组的三个元素中，我们都能找到正好两个元素满足：它与一个元素互素，并且与另一个元素不互素。

所以，我们只需枚举2到$n$的所有数，将每个数的与其互素的数的个数和与其不互素的数的个数相乘，最后求和并除以2，就是要求的逆问题的答案。

现在我们要考虑这个问题，如何求与2到$n$这些数互素（不互素）的数的个数。虽然求解与一个数互素数的个数的解法在前面已经提到过了，但在此并不合适，因为现在要求2到$n$所有数的结果，分别求解显然效率太低。

所以，我们需要一个更快的算法，可以一次算出2到$n$所有数的结果。

在这里，我们可以使用**改进的埃拉托色尼筛法**。

- 首先，对于2到$n$的所有数，我们要知道构成它的素数中是否有次数大于1的，为了应用容斥原理，我们还有知道它们由多少种不同的素数构成。

  对于这个问题，我们定义数组$deg[i]$：表示i由多少种不同素数构成，以及$good[i]$：取值$true$或$false$，表示$i$包含素数的**次数小于等于1**是否成立。

  再利用埃拉托色尼筛法，在遍历到某个素数$i$时，枚举它在2到$n$范围内的所有倍数，更新这些倍数的$deg[]$值，如果有倍数包含了多个$i$，那么就把这个倍数的$good[]$值赋为$false$。

- 然后，利用容斥原理求出2到$n$每个数的$cnt[i]$：在2到$n$中不与$i$互素的数的个数。

  回想容斥原理的公式，它所求的集合是不会包含重复元素的。也就是如果这个集合包含的某个素数多于一次，它们不应再被考虑。

  所以只有当一个数$i$满足$good[i] = true$时，它才会被用于容斥原理。枚举$i$的所有倍数$i \times j$，那么对于$i \times j$，就有$\frac{N}{i}$个与$i \times j$同样包含$i$（素数集合）的数。将这些结果进行加减，符号由$deg[i]$（素数集合的大小）决定。如果$deg[i]$为奇数，那么我们要用加号，否则用减号。

**程序实现：**

```
bool good[MAXN];
int deg[MAXN], cnt[MAXN];

long long solve() {
   memset (good, 1, sizeof good);
   memset (deg, 0, sizeof deg);
   memset (cnt, 0, sizeof cnt);

   long long ans_bad = 0;
   for (int i=2; i<=n; ++i) {
      if (!good[i]) continue;
      if (deg[i] == 0)  deg[i] = 1;
      for (int j=1; i*j<=n; ++j) {
         if (j > 1 && deg[i] == 1)
            if (j % i == 0)
               good[i*j] = false;
            else
               ++deg[i*j];
         cnt[i*j] += (n / i) * (deg[i]%2==1 ? +1 : -1);
      }

      //ans_bad += (g[i] - 1) * 1ll * (n - g[i]);
      ans_bad += (cnt[i] - 1) * 1ll * (n - cnt[i] - 1);
   }

   //return n * 1ll * (n-1) * (n-2) / 6 - ans_bad / 2;
   return (n-1) * 1ll * (n-2) * (n-3) / 6 - ans_bad / 2;
}
```

*(译注：原文的程序有一些问题，我对它作了一些改动。改动之前的代码为注释部分。)*

最终算法的复杂度为$O(n \log n)$，因为对于大部分i都要进行$\frac{n}{i}$次枚举。

## 3.10 错排问题

我们想要证明如下的求解长度为$n$序列的错排数的公式：

$$n! - C_n^1 \cdot (n-1)! + C_n^2 \cdot (n-2)! - C_n^3 \cdot (n-3)! + \cdots \pm C_n^n \cdot (n-n)!.$$

它的近似结果为：

$$\frac{n!}{e}$$

（此外，如果将这个近似式的结果向其最近的整数舍入，你就可以得到准确结果）

我们定义$A_k$：在长度为$n$的序列中，有一个不动点位置为$k(1 \le k \le n)$时的序列集合。

现在我们运用容斥原理来计算至少包含有一个不动点的排列数，要计算这个，我们必须先算出所有$A_k$、以及它们的交集的排列数。

$$|A_p| = (n-1)!.$$

$$|A_p \cap A_q| = (n-2)!.$$

$$|A_p \cap A_q \cap A_r| = (n-3)!.$$

因为我们知道当有$x$个不动点时，所有不动点的位置是固定的，而其它点可以任意排列。

用容斥原理对结果进行带入，而从$n$个点中选$x$个不动点的组合数为$C_n^x$，那么至少包含一个不动点的排列数为：

$$C_n^1 \cdot (n-1)! - C_n^2 \cdot (n-2)! + C_n^3 \cdot (n-3)! + \cdots \pm C_n^n \cdot (n-n)!.$$

那么不包含不动点（即错排数）的结果就是：

$$n! - C_n^1 \cdot (n-1)! + C_n^2 \cdot (n-2)! - C_n^3 \cdot (n-3)! + \cdots \pm C_n^n \cdot (n-n)!.$$

化简这个式子，我们得到了错排数的准确式和近似式：

$$n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + ... \pm \frac{1}{n!}\right) \approx \frac{n!}{e}$$

（因为括号中是$e^{-1}$的泰勒展开式的前$n+1$项）

用这个式子也可以解决一些类似的问题，如果现在求有$m$个不动点的排列数，那么我们可以对上式进行修改，也就是将括号中的累加到$\frac{1}{n!}$改成累加到$\frac{1}{(n-m)!}$。

# 4 在OJ的相关题目

这里列出了一些可以用容斥原理解决的习题。

- UVA #10325 ”The Lottery” [难度：简单]
- UVA #11806 ”Cheerleaders” [难度：简单]
- TopCoder SRM 477 ”CarelessSecretary” [难度：简单]
- TopCoder TCHS 16 ”Divisibility” [难度：简单]
- SPOJ #6285 NGM2 ”Another Game With Numbers” [难度：简单]
- TopCoder SRM 382 ”CharmingTicketsEasy” [难度：中等]
- TopCoder SRM 390 ”SetOfPatterns” [难度：中等]
- TopCoder SRM 176 ”Deranged” [难度：中等]
- TopCoder SRM 457 ”TheHexagonsDivOne” [难度：中等]

- SPOJ #4191 MSKYCODE ”Sky Code” [难度：中等]

- SPOJ #4168 SQFREE ”Square-free integers” [难度：中等]

- CodeChef ”Count Relations” [难度：中等]

# 5 参考文献

Debra K. Borkovitz. **”Derangements and the Inclusion-Exclusion Principle”**