排列与组合

排列与组合的知识是组合计数的基础，今天开始，我们将系统地介绍组合计数理论。

组合计数所要解决的中心问题是求满足某些条件的方案总数。

比如:

(l)在全系 100 名学生中选出 3 名学生去参加ACM 比赛，有多少种方案?

(2)一棵有n个结点的二叉树共有多少种不同构的形式?

(3)将 8 个皇后放在一张棋盘上使得没有两个处于互相攻击的位置，共有多少种放置方法?

排列与组合的知识就是解答诸如这类问题的。

组合计数理论最初的应用是概率的计算。

Laplace (1749-1827)首先定义了概率的概念。他说，如果一个确定的集合中总共有T 个对象，并且有一个具有 F 个对象的有利子集。那么，随机地从这个全集中选取出有利对象的概率是 F/T。

为了从这个定义计算概率，我们必须能够计算出有利对象的数目与可能情况的总数。这就涉及到组合计数问题。

1 基本概念

在学习排列与组合时我们首先引入两个非常直

观的原理。

加法原理 若事件 X 能以 x 种方式发生，另一个不同的事件 Y 能以 y 种不同的方式发生，则事件X 或事件 Y 能以(x+y)种方式发生。

例 1 全系共三个班级，这三个班级准备参加 ACM 竞赛的学生人数分别为 3、4、5，则全系准备参加 ACM 竞赛的人数为 3+4+5=12。

乘法原理 若事件 X 能以 x 种方式发生，另一个不同的事件 Y 能以 y 种不同的方式发生，则事件 X 和事件 Y 能以 xy 种方式发生。

例 2 求小于 10 亿且包含数字 l 的正整数的个数。

解首先计算不包含数字 1 的数的个数，从 0，2，3，4，5，6，7，8，9 中寻找 9 个符号的字符串。第一个数字有 9 种选择，第二个数字有 9 种选择，等等。

然后，根据乘法原理共有$9^9 = 387420489$个这样的数，其中包括 000000000。如果我们从 1000000000 中减去这样的数，得到答案为 612579511。

定义 1 从 n 个不同的元素中，有次序地选取 r 个元素，称为从 n 中取 r 个的排列，其排列数记为

到 P(n,r)。当 r=n 时，称此排列为全排列。

比如从 A, B, C,D 四个英文字母中有次序地选取出两个字母，则可以得到如下的 12 种情况:

AB AC AD BA BC BD CA CB CD DA DB DC

我们记为 P(4,2)=12。这就是从 4 个元素中取出 2 个元素的例子。

定理 3.1

$$P(n,r)=\frac{n!}{(n-r)!}$$

证明：在 n 个不同的元素中，有次序地选取出 r 个元素，可以采取以下方式：首先取一个元素，有 n 种选择;然后取第二个元素,有 n-1 个种选择;,,,,;最后取第 r 个元素，有(n-r+1)种选择。由乘法原理得

$$P(n,r)=n\cdot(n-1)\cdot(n-2)\cdot\cdots\cdot(n-r+1)$$

$$=\frac{n!}{(n-r)!}$$

例 3 从 n 个不同元素中选取出 r 个元素围成一个圆，求选取的方案总数。

解:从 n 个不同元素中选取出r 个元素的排列数为 $P(n,r)$ 。设自 $a_1,a_2,\cdots,a_r$ 为一个排列。将其加以变

换

$a_2, a_3, \cdots, a_r, a_1;\ a_3, a_4, \cdots, a_r, a_1, a_2;\ a_r, a_1, a_2 \cdots, a_{r-1}$

一共有 r 个排列。但是这 r 个排列对应同一个圆排列，也就是说每一个圆排列对应着 r 个线排列，所以从 n 个不同元素中选取出 r 个元素组成的圆排列总数为

$$\frac{P(n,r)}{r}$$

定义 2 从 n 个不同元素中，选取 r 个元素而不考虑其次序时，称为从 n 个中取出 r 个的组合，其组合数记为 C(n,r)，或

$$\begin{bmatrix} n \\ r \end{bmatrix}$$

比如从{A,B,C,D}中选取出 2 个元素的组合，则有如下情况:

{A,B} {A,C} {A，D} {B,C} {B,D} {C,D}

我们记作 C(4,2)=6 或

$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} = 6$$

对于一个从 n 个元素中选取出 r 个元素的每一个组合，我们都对这 r 个元素进行全排列，便得到了一个从 n 个元素中选取 r 个元素的排列。

所以可以得到下面的定理:

定理 2

$$\begin{bmatrix} n \\ r \end{bmatrix} = \frac{P(n,r)}{r!} = \frac{n!}{r!\,(n-r)!}$$

例 4 在如图 1 所示的棋盘中，若从左下角走到右上角，并且规定只能向右走或者向上走，问共有多少种方案?

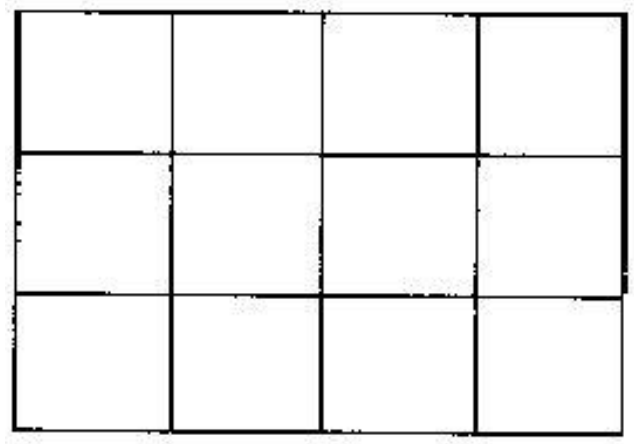

图 1 一个 4x3 的棋盘

解：我们看到图 1 是一个 4x3 的棋盘。如果从左下角向右走 4 步，然后再向上走3 步就可以到达右上角。并且发现:如果从左下角开始，无论怎么走，只要向右走 4 步和向上走 3 步都可以到达右上角。所以我们用 0 代表向右走一步，用1 代表向上走一步。那么从左下角到右上角的一种可能的走法就唯一地对应了一个由 4 个 0 和 3 个 1 组成的 7 位 01 字符串。反之，给定一个由4 个 0 和 3 个 1 组成的 7 位 01 字符串。我们规定 0 代表向右走，1 代表向上走，则这个 01 字符串又惟一地对应着一种可能的走法。这样便建立了一种一一对应关系。所以我们所求的方案数就是由 4 个 0 和 3 个 1 组成的 7 位

01 字符串的个数，即

$$\begin{bmatrix} 7 \\ 4 \end{bmatrix} = 35$$

有一些数学问题可以使用不同的方法来得到答案。在组合数学中这些问题是大量存在的。不同的人由于对于问题的理解不同，解决问题的思路必然会有差异。下面的问题我们给出两种方法求解。

例 5 假定有 B 个男孩子和 G 个女孩子，希望选出不同性别的一对孩子。问共有多少不同的方案。

解：我们可以从B 个男孩子中选出一个，再从 G 个女孩子中选出一个。根据乘法原理，共有 BG 种不同的方案。

我们也可以用其他的方法说明这个问题。从全部 B+G 个人的集合中用

$$\begin{bmatrix} B+G \\ 2 \end{bmatrix}$$

种方法选出一对，这些方案中包含了

$$\begin{bmatrix} B \\ 2 \end{bmatrix}$$

种全部都为男孩子的情况和

$$\begin{bmatrix} G \\ 2 \end{bmatrix}$$

种全部都为女孩子的情况。因此选出不同性别的一

对孩子的方案数为

$$\begin{bmatrix} B+G \\ 2 \end{bmatrix} - \begin{bmatrix} B \\ 2 \end{bmatrix} - \begin{bmatrix} G \\ 2 \end{bmatrix}$$

$$= \frac{(B+G)(B+G-1)}{2} - \frac{B(B-1)}{2}$$

$$- \frac{G(G-1)}{2} = BG$$

2 排列与组合的生成算法

在许多场合中，我们经常生成一些元素的特定排列与组合。本节给出一些常见的排列组合生成算法。

2.1 r-排列生成算法

我们采用回溯法来生成从 n 个元素中取出 r 个元素的所有排列情况。其中 n 个元素用 $1,2,\cdots,n$ 表示。在程序中，过程 done 递归的层数 i 表示当前正在生成排列中第 i 个位置的数。过程 done(i)执行时，首先判断 j 是否在该排列以前的几个位置上出现过。若已经出现过，那么则说明 j 不可能出现在当前位置上，此时 j 值增 1 重复以上判断，j=n 时回溯；若 j 没有在该排列以前的位置上出现，则该位置上的值就是 j，这时判断递归的层数 i 与 r 的值是否相等。若 i=r,输出一个新的排列并回溯；若 i<r，则继续进行递归。

参考程序 1:　Example2-1

```
program examples 2_1;
var
    n，r:integer;
    flag:set of byte;{记录在生成一个排列的过程中出现过的元素}
    data:array of [1..20] of integer;{记录相应位置上的元素的值}
procedure out;{输出一个新的排列}
var
    i:integer;
begin
    for i:=1 to r do write(datat[i],’ ‘);
writeln;
end;
    procedure done(i:integer);
                                    {递归的层数i 表示正在生成第 i 个位置上的元素}
    Var
        j:integer;
    begin
```

```
for j:=1 to n do
    if not (j in flag) then
    begin
        flag:=flag+[j];
        data[i]:=j;
        if (i=r) then out
                 else done(i+1);
        flag:=flag-[j];
    end;
end;

begin
        read (n,r);
        done(1);
end.
```

2.1 错位排列生成算法

错位排列生成算法与 r- 排列生成算法的不同之处在于:在生成错位排列第 i 个位置上的元素时，必须保证该元素不等于 i。

下面给出生成错位排列的程序。

参考程序 2：Example2_2

```
program example2_2;
```

```
var
    n,r:integer;
    flag:set  of  byte;记录在生成一个排列的过程中出现过的元素}
    data:array[1..20]  of  integer;{记录相应位置上的元素的值}

    procedure out;{输出一个新的排列}
    var
        i:integer;
    begin
    for i:=1 to r do    write(data[i],' ');
    writeln;
    end;

    procedure done(i:integer);
                                {递归的层数i 表示正在生成第i 个位置上的元素}
        Var
            j:integer;
        begin
            for j:=1 to n do
                if not (j in flag) and (i<>j) then
```

```
{必须保证该元素不等于 i}
        begin
            flag:=flag+[j];
            data[i]:=j;
            if(i=r) then out
                    else done(i+1);
            flag:=flag-[j];
        end;
    end;

    begin
        read(n,r);
        done(1);
    end.
```

2.2 r-组合生成算法

这里所介绍 r-组合生成算法与 r-排列生成算法非常相似。我们知道，从 n 个元素中取出 r 个元素的一个组合情况恰对应了 r!个 r-排列情况。所以当我们按照元素的递增次序放置相应位置上的元素时，就可以产生从 n 个元素中取出 r 个元素的所有组合情况。

下面给出 r--组合生成算法的程序。

参考程序 3:　Example2_3

```
program example2_ 3;
Var
    n,x:integer;
    flag :set of byte;{记录在生成一个组合的过程中出现过的元素}
    data:array[1..20] of integer;{记录相应位置上的元素的值}

    procedure out;{输出一个新的组合}
    var
        i:integer;
    begin
        for i:=1 to r do write(data[i],' ');
        writeln;
    end;

    procedure done(i:integer);
                                {递归的层数i 表示正在生成第 i 个位置上的元素}
    Var
        j:integer;
```

```
    beqin
        for j:=data[i-1]+1 to n do   {按照元素递增的次序生成所有组合情况}
            if not (j in flag) then
            begin
                flag:=flag+[j];
                data[i]:=j;
                if(i=r) then out
                        else done(i+1);
                flag:=flag-[j];
            end;
    end;

begin
    data[0]:=0;
    read(n,r);
    done(1);
end.
```

3 购票问题

3.1 问题描述

农夫 John 和他的朋友们一同去参加 Cownty 展览会。Cownty 展览会的门票为$50。John 发现一个奇

怪的现象:在排队购票的 2n 个人中，总有 n 个人拿的是面值为$100 的钞票，而另外的 n 个人拿的是面值为$50 的钞票。农夫 John 想知道的是在这种情况下这 2n 个人共有多少种排队方式，使售票处不至出现找不开钱的局面 (假设售票处原来没有零钱的情况)?

3.2 试题分析

本题将采用五种算法分析这道题目，并对各种算法的效率加以比较。从中可以看出组合数学理论在算法优化方面所起到的显著作用。

1.算法 1:搜索策略

我们用回溯法来直观地枚举所有情况。算法中指定一变量 k 记录售票处有$50 钞票的张数，初始时令 k=0，若某人手持$100 钞票且 k=0 时则回溯，否则继续递归。若第 2n 个人购完票即递归进行到第 2n 层时计数器累加 1。递归结束后，计数器中的值便为排队方案总数。

参考程序 4:　example3-1

```
program example3_ 1;
var
    n,k,m:integer;
    total:longint;
```

```
procedure DFS(i:integer);    {i 为递归层数}
var
    j:integer;
begin
    for j:=0 to 1 do
    begin
        if (j=0) then
        begin
    inc(k); {k 表示计数器}
    inc(m) ; {m 表示有多少人手持$50 的钞票}
    if (m=n) then ins(total)
                        else dfs(i+1);
    dec(k);
    dec(m);
end
else begin
    if k>0 then
        begin
            dec(k);
            dfs(i+1);
            inc(k);
        end;
end;
```