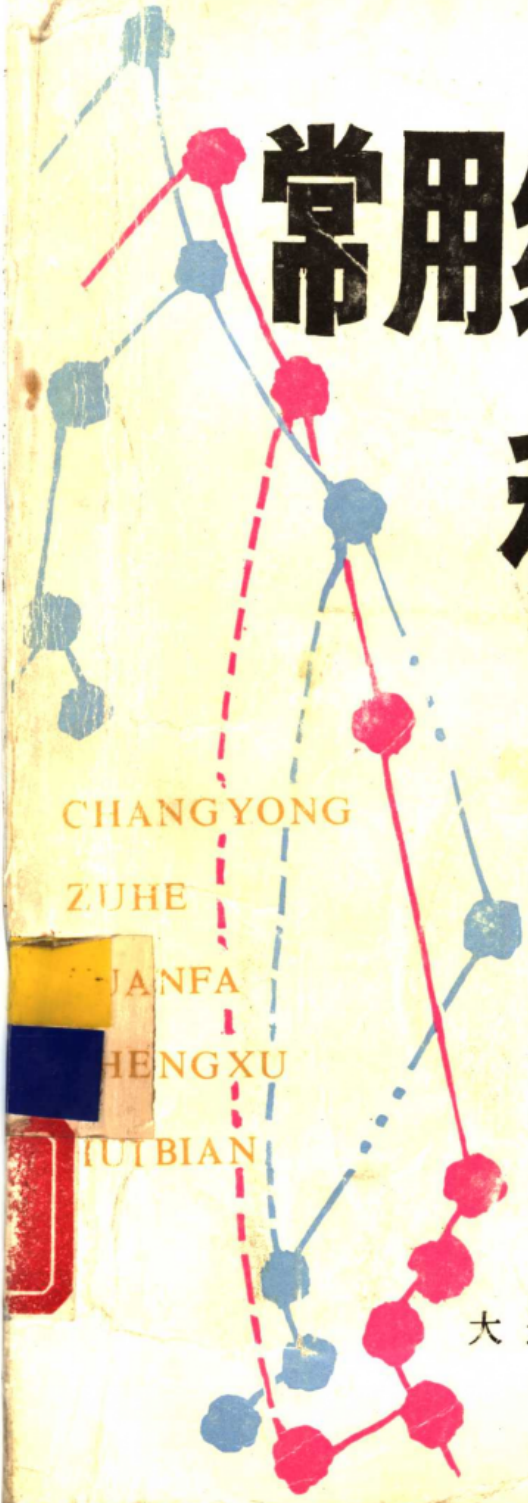
常用组合算法程序汇编



CHANGYONG
ZUHE
JANFA
HENGXU
IUIBIAN

大连工学院出版社

常用组合算法程序汇编

迟忠先　左　垲　李宪廷　等编

大连工学院出版社

内 容 简 介

本书介绍了组合理论图论中常用算法程序的功能及使用方法，对算法本身也作了简要说明。全书共分九章，前两章是介绍组合数学中的枚举计数和随机抽样的有关算法。第三、四、五、六四章是分别介绍图论中图、树、路和流的有关算法。最后三章是从算法设计的角度分别介绍三种常用算法设计方法——动态规划法、回溯法和启发式方法的应用。

本书可供非数值和数值应用领域中计算工作者编程时参考，也可作为大专院校网络图论、组合数学、算法分析、数据结构等课程的实习教材和参考书。

常用组合算法程序汇编

CNANG YONG ZUHE SUAN FA
CHENG XU HUI BIAN

迟忠先 左 垲 李宪廷 等编

大连工学院出版社出版发行
（大连市甘井子区凌水桥）
辽宁省新华书店经销
大连船舶生产服务公司印刷厂印刷

开本：787×1092 1/32 印张：14 3/4 字数：319.1千字
1987年5月第1版 1987年5月第1次印刷
印数：1—4000册

责任编辑：尉迟洁斐 封面设计：羊 戈
责任校对：金 戈

统一书号：15400·7 ISBN 7 5611 0028 0/TP·2

定价：2.42元

前　言

组合学是一门历史悠久、应用广泛的数学分支，它所包括的组合数学与图论两部份内容已经发展成为现代离散数学重要的组成部份。由于这门学科主要是分析研究离散性对象在系统中的结构关系，它所涉及的问题具有普遍性，因而它对于多种学科技术和社会经济系统有着广泛的应用价值。例如，组合理论在计算机科学、网络优化、空间技术、人工智能、通讯网络、信息编码、土木工程、交通运输、物质结构、实验设计、生物学、遗传学、心理学、人类学、语言学、管理科学、社会经济学以及物理学，化学和数学的其它分支都有一系列的具体应用。

到目前为止，组合理论的成果已相当丰富，特别是电子计算机的广泛应用，使组合学获得了新的生命力，相应产生了“算法组合学”，“组合算法”等新颖的学科分支。“组合算法”把理论成果转化为算法使之有可能在计算机上实现，因此，它是理论联系实际的桥梁，是应用组合理论来解决实际问题的关键。目前已提出的大量组合算法已成为解决许多领域问题的共同工具。特别是在非数值应用领域中，组合算法的应用更为广泛。

然而，在目前已提出的大量算法中，有许多还未程序化，有些虽然有了程序但比较孤立、分散，未成系统。这就大大地阻碍了组合算法的实际推广应用。为此，我们收集了有关组合数学、图论及组合优化等方面的常用算法程序六十余个，编写了本书。书中的算法程序采用 FORTRAN 或

PASCAL语言编写，都已在CROMEMCO，PC STM,IBM PC XT等微机上调试通过。全书共分九章，前两章介绍组合数学中枚举计数和随机抽样的有关算法。第三，四，五，六章分别介绍网络图论中有关图、树、路和流的算法。最后三章从算法设计的观点出发分别介绍三种常用算法设计方法——动态规划法，回溯法和启发式方法的应用。

参加本书编写工作的有：大连工学院迟忠先，李宪廷，许宏，韩杰；北京邮电学院舒贤林，徐志才，高元华，霍锡真，王练辉；安徽大学迟成文，汪世铭，张良震；中国科技大学研究生院左垲等同志，全书由迟忠先，左垲，李宪廷审校。在本书编写过程中，得到清华大学常迥教授、北京邮电学院胡健栋教授和大连工学院王众讬教授的热情关怀和支持，在此表示衷心感谢！

书中难免存在错误和缺点，恳望读者惠于批评，指正。

编　者

1986年10月

目　　录

第一章　枚举计数

组合数学中，组合对象的枚举与计数是最基本的问题之一。本章将给出有关枚举的一些基本算法及相应的FORTRAN子程序。这些子程序有如下特点：能够识别出最后一个组合对象，能够保存前趋的足够信息，从而在生成后继时可以节省大量的计算。

这类子程序的典型调用结构如下：

```
        {设置参数}
        MTC=.FALSE.
10      CALL NEX…
        {处理输出的组合对象}
        IF (MTC) GOTO 10
…
```

需要说明的是，在处理输出的组合对象时，注意不要修改参数，因为这些参数保存着为生成后继所需要的信息。

1.1　生成所有子集(NEXSUB/LEXSUB)

（1）功能

子程序NEXSUB按格雷码规定的次序生成{1，2，…，n}的子集。

子程序LEXSUB按字典序生成{1，2，…，n}的子集，且子集的基数不超过给定值。

（2）调用方式

为用NEXSUB生成{1，2，…，n}的所有子集，可采用如下的调用结构：

```
        {设置参数}
        MTC=.FALSE.
10      CALL NEXSUB (N, IN, MTC, NCARD, J)
        {处理输出子集}
        IF (MTC) GOTO 10
        ...
```

其中，N为整型变量，集合的基数。IN为整型一维数组，它有N个元素。当I属于输出的子集时，IN(I)=1，否则为0。MTC为逻辑变量，每当子程序返回时，它都被置为.TRUE.或.FALSE.，如果是.TRUE.，则说明当前返回的子集不是最后一个，如果是.FALSE.，则当前返回的子集是最后一个。NCARD为输出子集的基数。J为整型变量，是当前子集与其前趋不同的元素下标。

为用LEXSUB生成{1，2，…，n}的所有子集，可采用如下的调用结构：

```
        {设置参数}
        K=0
10      CALL LEXSUB (N, K, IN, JMP, NDIM)
        IF (K.EQ.0) GO TO 20
        {处理输出子集}
        GO TO 10
20      {结束}
        ...
```

其中，N为整型变量，集合的基数。K为整型变量，输出子集的基数。IN为整型一维数组，它有NDIM个元素，IN(I)是按增序给出的子集的第I个成员（I=1，…，K）。JMP为

逻辑变量，用来控制某些集合的产生，当JMP=.TRUE.时，跳过所有基数超过限定值的子集。NDIM为整型变量，子集的最大长度。

（3）算法说明：

格雷代码法

一般说来，按格雷码顺序所生成的子集序列对应着n维立方体的一条Hamilton路径。例如集合{1，2，3}的所有子集序列为：

ϕ，{1}，{1，2}，{2}，{2，3}，{1,2，3}，{1，3}，{3}

若用格雷码给出，即为：

(0，0，0)，(1，0，0)，(1，1，0)，(0，1，0)

(0，1，1)，(1，1，1)，(1，0，1)；(0，0，1)

这恰好是如图1.1.1所示的立方体上的一条Hamilton路径：

而我们的目的就是用算法来刻划这样一条路径。

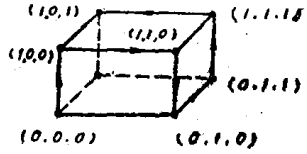


图1.1.1

设Ln是n维立方体上的一条Hamilton路径，$\overline{Ln}$表示其反向路径，则有

$L_n = L_{n-1}\otimes 0$，$\overline{L_{n-1}}\otimes 1$，（$n\geqslant 1$，$L_0$为空）

这即是说，在L_{n-1}中每个向量的右边加一个0，然后在L_{n-1}的反向表中，将每个向量的右边加一个1就可以得到L_n。例如：

L_1	L_2	L_3
0	00	000
1	10	100
	11	110
	01	010

L_1	L_2	L_3
		011
		111
		101
		001

假定我们已得到了L_n中的某个向量（当然对应一集合）S，而现在想得到S的直接后继，亦即，想要找出为生成后继而需改变的坐标a_j的下标j。对L_3来说，这个下标序列为：

1，2，1，3，1，2，1

一般地，若令S的基数为k，则有如下规则：

k为偶数时，j=1；当k为奇数时，j是S的第一个“1”位之后的坐标的下标。

因此，只要知道集合S及其基数的奇偶性，就能应用上面的规则来确定其直接后继。如果在生成表列的同时，还要识别出最后的子集（$a_n=k$），则需要的是基数本身而不仅仅是它的奇偶性。

NEXSUB算法

(a) 〔第一次入口〕

$a_i \leftarrow 0$ $(i=1, \cdots, n)$；$k \leftarrow 0$；转出口；

(b) 〔以后各次入口〕

$t \leftarrow mod(k, 2)$；$j \leftarrow 1$；如果$t \neq 0$，转(d)

(c) 〔改变第j位〕

$a_j \leftarrow 1-a_j$；$k \leftarrow k+2a_j-1$；

如果$k=a_n$，转最后出口；转出口。

(d) 〔找第一个“1”位〕

$j \leftarrow j+1$；如果$a_{j-1}=1$，转（c）；否则转（d）。

字典编序法

假定S由其k个成员的表$(a_1, a_2, \cdots, a_k)$来表示，其中$a_1<a_2<\cdots<a_k$。令$S'=(a'_1, a'_2, \cdots, a'_{k'})$，若存在$j\leqslant \min(k, k')$，使得对$i<j$有$a_i=a'_i$，而$a_j<a'_j$，或者是对$i\leqslant \min(k, k')$有$a_i=a'_i$，而$k<k'$，则按字典序S先于S′。这种编序法有一个重要性质：对任一集合$S=(a_1, a_2, \cdots, a_k)$，位于S之前且与S最接近的i元集合$(i<k)$为$(a_1, a_2, \cdots, a_i)$，而为生成S所需要的信息总是由其前趋的第$k-1$个坐标提供。

LEXSUB算法

(a) 〔输入n，k，J，$(a_1, a_2, \cdots, a_k)$；排除平凡情况；置初值〕
如果$k\neq 0$，转（b）；如果J=.TRUE.，转出口；
$s\leftarrow 0$；转（c）

(b) 〔测试a_k和k是否最大值〕
如果$a_k=n$，转（e）；$s\leftarrow a_k$；如果J=.TRUE.，转（d）。

(c) $k\leftarrow k+1$

(d) $a_k\leftarrow s+1$；转出口。

(e) 〔执行$a_k=n$时的任务〕
$k\leftarrow k-1$；如果$k=0$，转出口；
$s\leftarrow a_k$；转（d）。

需要说明的是，对某些应用来说，要跳过按字典序排列的一部分集合，也就是说基数可能有一个界限，为了适应这种情况，算法中有一个参数J，若J为真，就跳过基数超过界

限的所有子集。此外，该算法不具有记忆功能，它在最后一个子集产生之后又给出空集（K = 0），这也是用户必须提供的开始点。

（4）子程序清单

```
      SUBROUTINE NEXSUB (N,IN,MTC,NCARD,J)
      INTEGER IN (N)
      LOGICAL MTC
      IF (MTC) GOTO 20
      DO 11 I=1,N
11    IN (I) =0
      NCARD=0
      MTC=.TRUE.
      RETURN
20    J=1
      IF(MOD(NCARD,2) .EQ.0) GOTO 30
40    J=J+1
      IF (IN(J-1) .EQ.0) GOTO 40
C     IF (J.GT.N) J=N
30    IN (J) =1-IN (J)
      NCARD=NCARD+2*IN (J) -1
      MTC=NCARD.NE.IN(N)
      RETURN
      END

      SUBROUTINE LEXSUB (N,K,IN,JMP,NDIM)
      DIMENSION IN (NDIM)
      LOGICAL JMP
10    IF (K.NE.0) GOTO 40
20    IF (JMP) RETURN
30    IS=0
100   IF (.NOT.JMP) K=K+1
```

```
110   IN(K) =IS+1
      RETURN
40    IF (IN(K) .EQ.N) GOTO 50
80    IS=IN(K)
90    GOTO 100
50    K=K-1
60    IF (K.EQ.0) RETURN
      IS=IN (K)
      GOTO 110
      RETURN
      END
```

(5) 实例

对N=5，我们从 MTC=.FALSE.开始反复调用子程序 NEXSUB，直到 MTC再次变为.FALSE.时终止。下面是所用的主程序及输出结果。输出结果的每一行分别是 IN(1)，IN(2)，…，IN(5)，跟在后面的是被改变坐标的下标 J 及子集的基数 NCARD。

```
      PROGRAM MAIN
      DIMENSION IN (5)
      LOGICAL MTC
      MTC=.FALSE.
10    CALL NEXSUB (5, IN, MTC, NCARD, J)
      WRITE (*, 20) IN, J, NCARD
20    FORMAT (1X, 5I2, I4, I2)
      IF (MTC) GO TO 10
      STOP
      END
```

```
0 0 0 0 0  0 0
1 0 0 0 0  1 1
1 1 0 0 0  2 2
0 1 0 0 0  1 1
0 1 1 0 0  3 2
1 1 1 0 0  1 3
1 0 1 0 0  2 2
0 0 1 0 0  1 1
0 0 1 1 0  4 2
1 0 1 1 0  1 3
1 1 1 1 0  2 4
0 1 1 1 0  1 3
0 1 0 1 0  3 2
1 1 0 1 0  1 3
1 0 0 1 0  2 2
0 0 0 1 0  1 1
0 0 0 1 1  5 2
1 1 0 1 1  1 3
1 1 0 1 1  2 4
0 1 0 1 1  1 3
0 1 1 1 1  3 4
1 1 1 1 1  1 5
1 0 1 1 1  2 4
0 0 1 1 1  1 3
0 0 1 0 1  4 2
1 0 1 0 1  1 3
1 1 1 0 1  2 4
0 1 1 0 1  1 3
0 1 0 0 1  3 2
1 1 0 0 1  1 3
1 0 0 0 1  2 2
0 0 0 0 1  1 1
```

集合{1，2，3，4，5，6}有41个基数不超过3的非空子集，下面的主程序可用来生成这些子集。

主程序可用来生成这些子集.

```
      PROGRAM MAIN
      DIMENSION IN(3)
      LOGICAL JMP
      K=0
10    JMP=K.EQ.3
      CALL LEXSUB (6, K, IN, JMP, 3)
      IF (K.EQ.0) GO TO 30
      WRITE (*, 20) (IN(I) , I=1, K)
20    FORMAT (1X, 3I2)
      GO TO 10
30    STOP
      END
```

1	1 5 6	3 4
1 2	1 6	3 4 5
1 2 3	2	3 4 6
1 2 4	2 3	3 5
1 2 5	2 3 4	3 5 6
1 2 6	2 3 5	3 6
1 3	2 3 6	4
1 3 4	2 4	4 5
1 3 5	2 4 5	4 5 6
1 3 6	2 4 6	4 6
1 4	2 5	5
1 4 5	2 5 6	5 6
1 4 6	2 6	6
1 5	3	

1.2 生成所有k一子集 (NEXKSB/NXKSRD)

(1) 功能

子程序 NEXKSB 按字典序生成{1，2，…，n}的k-子集。

子程序NXKSRD按RD顺序生成{1，2，…，n}的k -子集（参见算法说明部分）。

其实，{1，2，…，n}的k-子集与从n中取k个元素的组合是一一对应的，所以这两个子程序也可用来枚举所有从n中取k个元素的组合。

(2) 调用方式

为用NEXKSB生成{1，2，…，n}的所有k-子集，可采