离散数学实验报告

学号：1120221704　　姓名：　　戴尚轩

目录

1 求命题的主范式 1
1.1 概述 1
1.2 步骤流程 1
1.3 程序实现 1
1.3.1 PriorityofOperator 2
1.3.2 evalRPN 2
1.3.3 calculate 3
1.3.4 Print 4
2 消解算法 4
2.1 概述 4
2.2 步骤流程 5
2.3 程序实现 5
2.3.1 alphacount 6
2.3.2 Init 6
2.3.3 connect 7
2.3.4 dispel 7
3 求关系的传递闭包 8
3.1 概述 8
3.2 步骤流程 8
3.3 程序实现 8
3.3.1 Get 9
3.3.2 Walshell 9
3.3.3 Output 9
4 求偏序集中的极大元与极小元 10
4.1 概述 10
4.2 步骤流程 10
4.3 程序实现 10
4.3.1 search 11
4.3.2 find 11
4.3.3 input 12
4.3.4 output 12
5 代数系统算律的判断 12
5.1 概述 12
5.2 步骤流程 13
5.3 程序实现 13
5.3.1 find 14
5.3.2 Input 14
5.3.3 Commutativecal 14
5.3.4 Associativecal 14
5.3.5 Idemponentcal 14
5.3.6 Output 14
5.3.7 Indentityfind 14
5.3.8 Zerofind() 15

6 求 Zn 群中元素的阶 15
6.1 概述 15
6.2 步骤流程 15
6.3 程序实现 15
6.3.1 Rank 16
7 二部图的判定 16
7.1 概述 16
7.2 步骤流程 17
7.3 程序实现 17
7.3.1 Input 17
7.3.2 DFS 17
7.3.3 find 18
7.3.4 Output 19
8 有向图连通性的判定 19
8.1 概述 19
8.2 步骤流程 20
8.3 程序实现 20
8.3.1 Input 20
8.3.2 justify 20
8.3.3 Output 21

表目录

表 1　求命题的主范式函数说明表 2
表 2　消解算法函数说明表 6
表 3　求关系的传递闭包函数说明表 8
表 4　求偏序集中的极大元与极小元函数说明表 10
表 5　代数系统算律函数说明表 13
表 6　求 Zn 群中元素的阶函数说明表 15
表 7　二部图的判定函数说明表 17
表 8　有向图连通性的判定函数说明表 20

图目录

图 1 求命题的主范式流程图 1
图 2 evalRPN 函数执行流程 3
图 3 calculate 函数执行流程 4
图 4 消解算法流程图 5
图 5 Init 函数执行流程 6
图 6 dispel 函数执行流程 7
图 7 求关系的传递闭包流程图 8
图 8 Walshall 函数执行流程 9
图 9 求偏序集中的极大元与极小元流程图 10
图 10 search 函数执行流程 11
图 11 find 函数执行流程 12
图 12 代数系统算律的判断流程图 13
图 13 求 Zn 群中的节程序流程图 15
图 14 Rank 函数执行流程 16
图 15 二部图的判定程序流程图 17
图 16 DFS 函数执行流程 18
图 17 find 函数执行流程 19
图 18 有向图连通性的判定程序流程图 20
图 19 Walshell 函数执行流程 21

1求命题的主范式

1.1概述

输入命题公式的合式公式，求出公式的真值表，并输出该公式的主合取范式和主析取范式。

1.2步骤流程

程序运行主逻辑如图 1 所示。

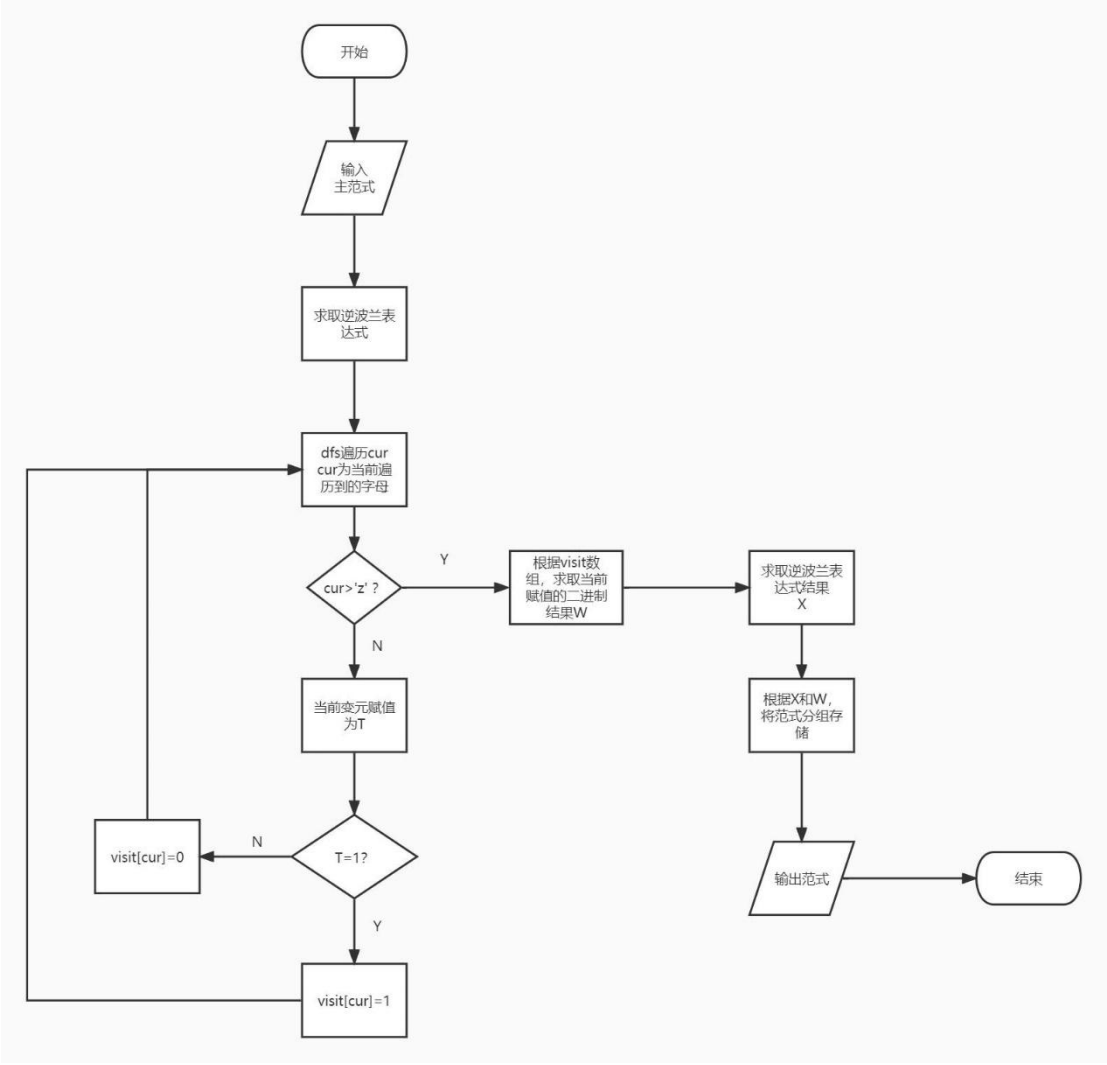

图 1 求命题的主范式流程图

1.3程序实现

表 1 求命题的主范式函数说明表

序号	名称	说明
1.	PriorityofOperator	用于归定和处理运算符的优先级
2.	evalRPN	用于将输入的字符串转成逆波兰表达式
3.	calculate	用于计算逆某一赋值下的结果
4.	Print	用于输出最终结果

1.3.1PriorityofOperator

用于归定和处理运算符的优先级，其中 ！ 优先级最高，其次是&、|、-、+、（）、最后 # 为 哨兵运算符，用于强制栈内运算符弹出，维护计算结果的正确性。

1.3.2evalRPN

用于将输入的字符串转成逆波兰表达式,其中就要使用函数一作为辅助。依次遍历整个字符串，如果为字母，则直接输出到存储结果的数组中，如果为运算符&、|、+、-，则需要与栈顶运算符的优先级进行比较，处理类似 ！插队（优先级更高）的现象，如果为（ 则直接入栈，如果为），则一直弹出栈顶元素直到找到与之匹配的左括号。

求取最终表达式过程中将读入并维护出现不同变元的总种类数，同时更新SumofVar，用于统计最终范式和迭代的数量。 关于本函数的执行流程，如所示。

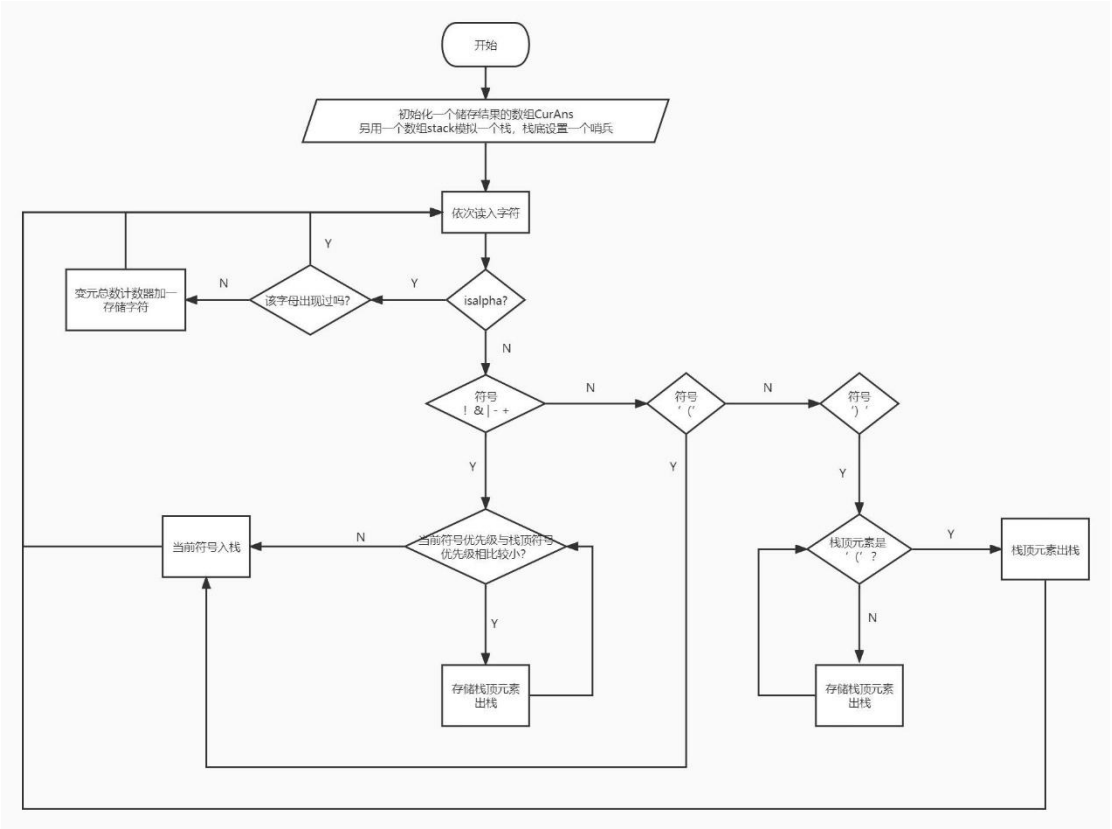

图 2 evalRPN 函数执行流程

1.3.3calculate

用于计算逆波兰表达式在某一赋值下的结果。函数参数传入目前每个字母对应的赋值的数组和逆波兰表达式的字符串。设置一个数值栈，读到字母的时候推入字母对应的值，读到!则将栈顶值取反，读到&、|、-、+取两个栈顶元素计算。最后将栈顶元素（此时栈内仅一个元素）作为函数返回值，即是本次计算的结果。

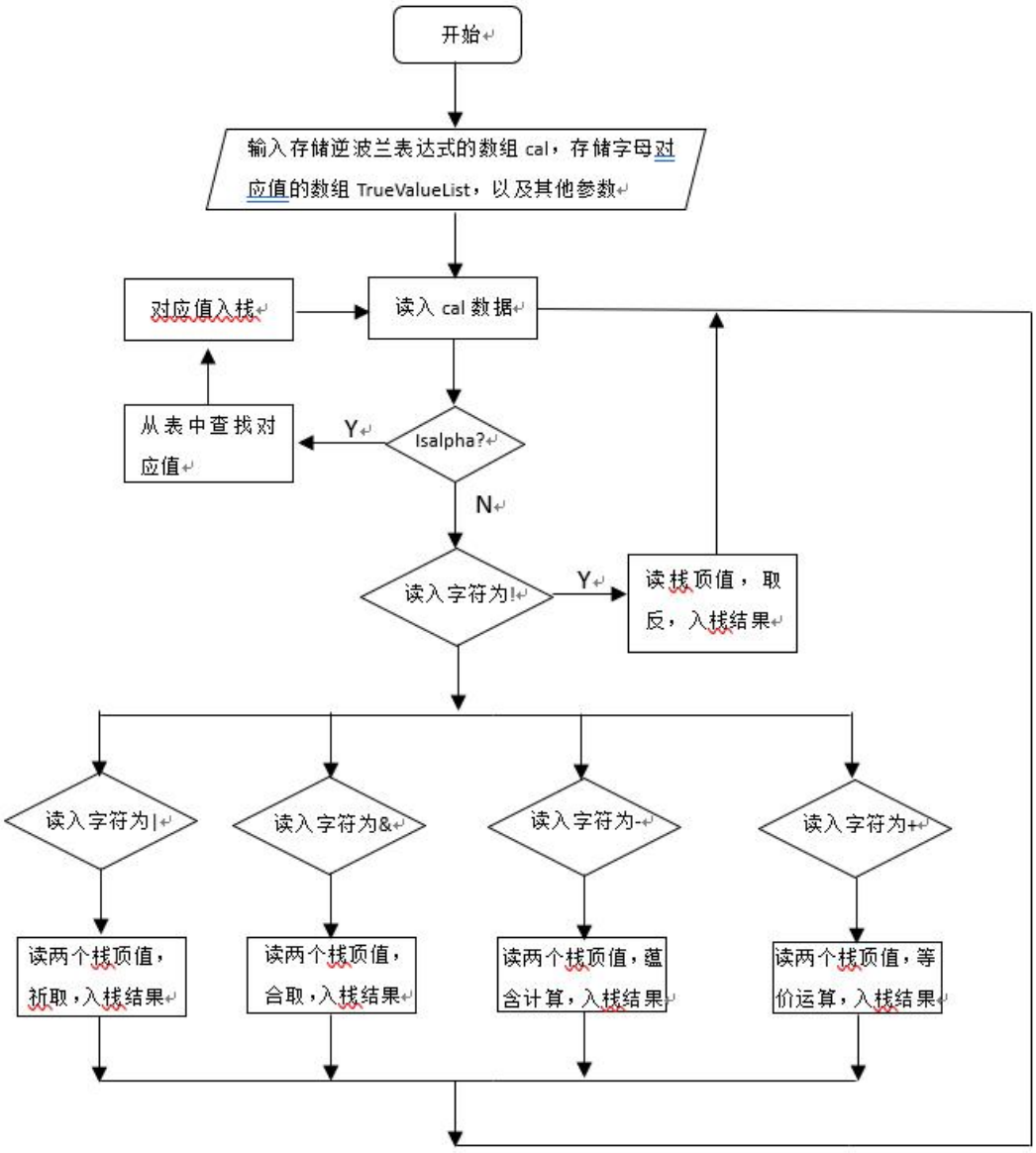

图 3 calculate 函数执行流程

1.3.4Print

每次计算得到的结果，为真存到 m 数组中，为假存到 M 数组中，记录两个数组存入的数据数；如果 length_m 为 0 为矛盾式，length_M 为 0 为永真式，最后按照顺序输出即可。

2消解算法

2.1概述

输入：合式公式 A 的合取范式，当 A 是可满足时，回答“YES ”；否则回

答“NO”。

2.2步骤流程

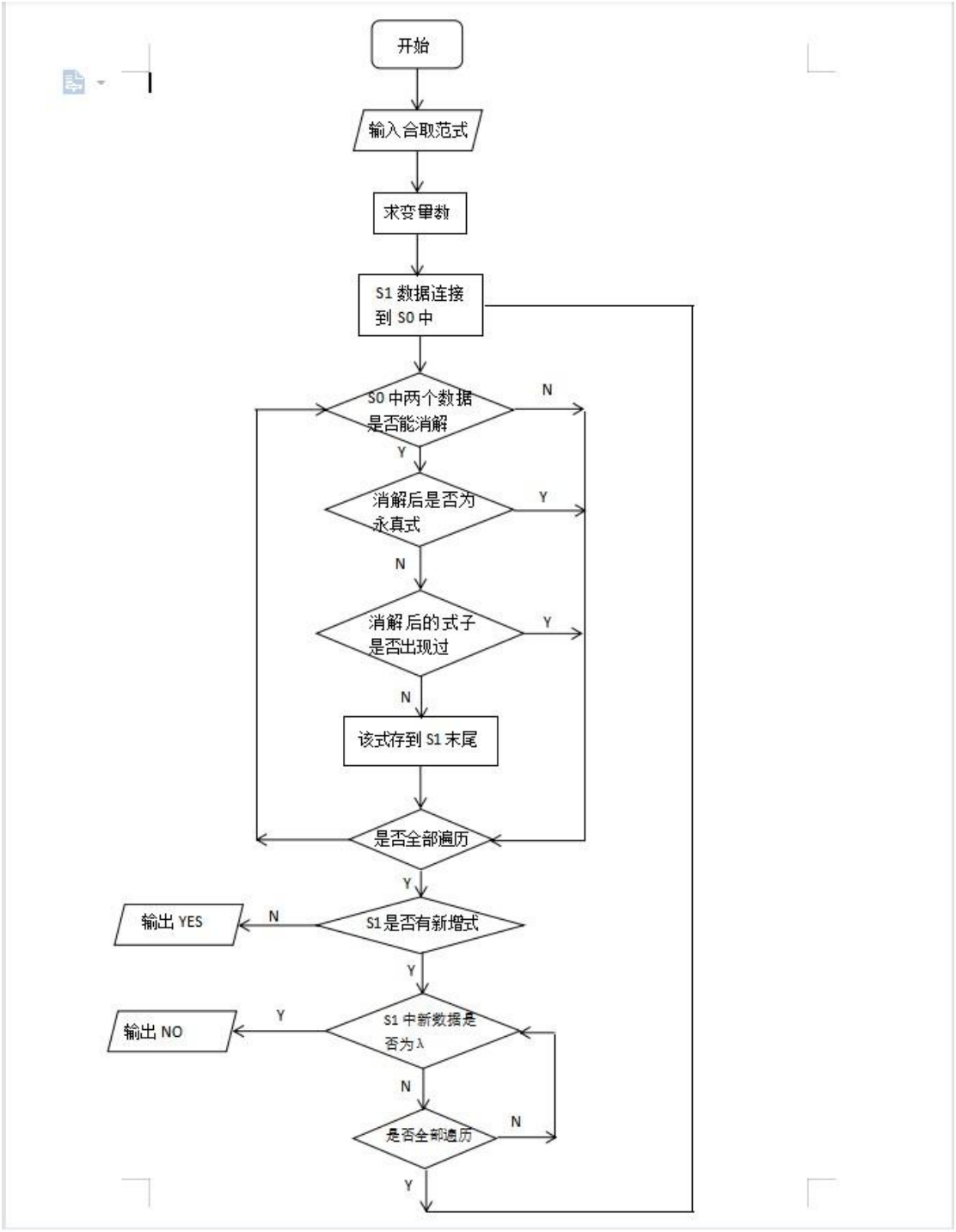

图 4 消解算法流程图

2.3程序实现

表 2 消解算法函数说明表

序号	名称	说明
1.	alphacount	用于计算不同的字母个数
2.	Init	用于初始化 S0
3.	connect	用于将 S1（新生成式）连接到 S0 中
4.	dispel	用于实现具体消解和输出

2.3.1alphacount

用于计算不同的字母个数，读取整个需要消解的式子，遇到字母则与已有的字母表 alphalist 进行对比，如果没有出现过该字母则将其顺次存到 alphalist 中，最后并返回 alphalist 的串长。

2.3.2Init

用于初始化 S0。读取输入的主和取范式，将每一个析取式存入 S0 中。如果该式是永真式，则忽略。

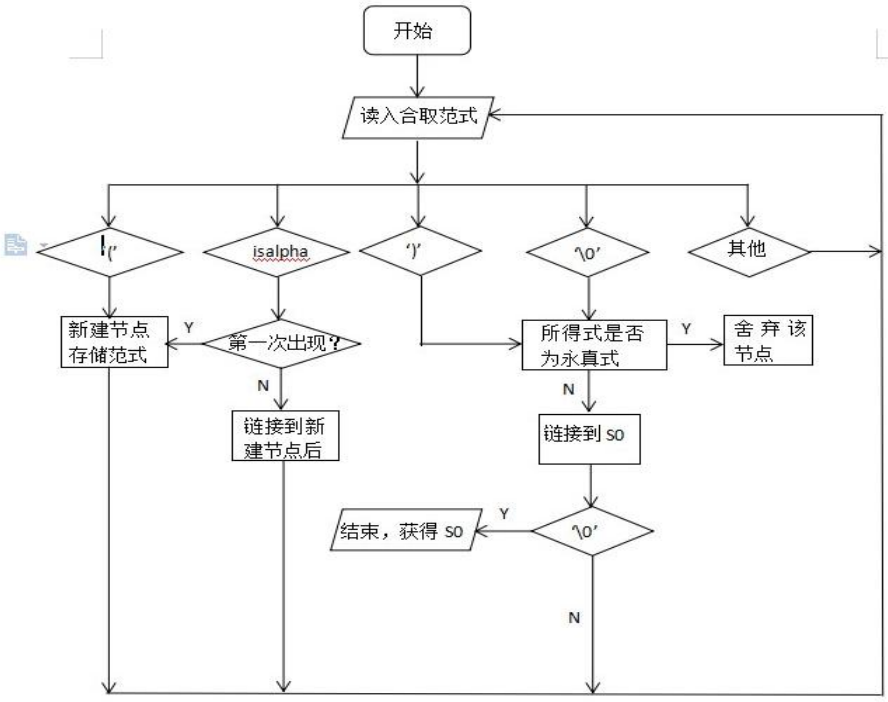

图 5 Init 函数执行流程

2.3.3connect

用于将 S1（新生成式）连接到 S0 中。由于采用的是链表的链接，直接将 S1 的数据接到 S0 后面，并将 S1 置 NULL 即可

2.3.4dispel

用于实现具体消解和输出。每次将 S0 内所得合取式依次互相比较，能消解且消解所得非永真式的即存到 S1 中。对于 S1，本次没有获得新的消解式即输出 YES，否则读取所有 S1 内式子，有 λ 的输出 NULL，没有的进行 connect 并再次消解.

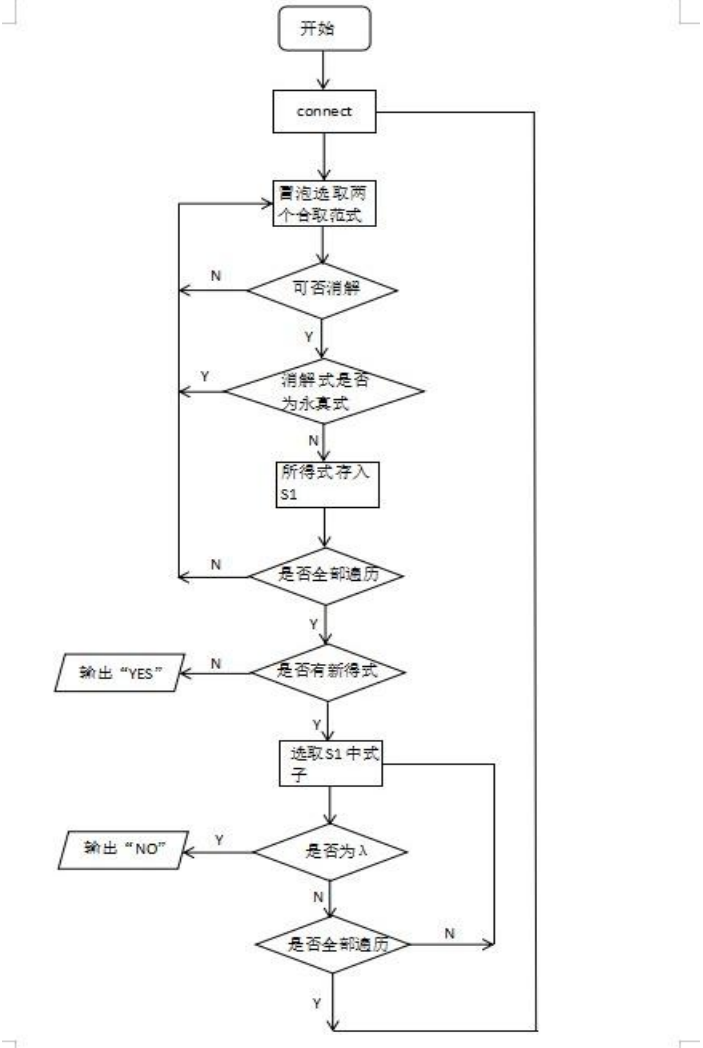

图 6 dispel 函数执行流程

3求关系的传递闭包

3.1概述

输入：一次输入一个关系矩阵，每一行两个相邻元素之间用一个空格隔开，输入元素的行与列分别对应关系矩阵的航宇列。关系的基数小于 12。输出：给关系的传递闭包所对应的关系矩阵。

3.2步骤流程

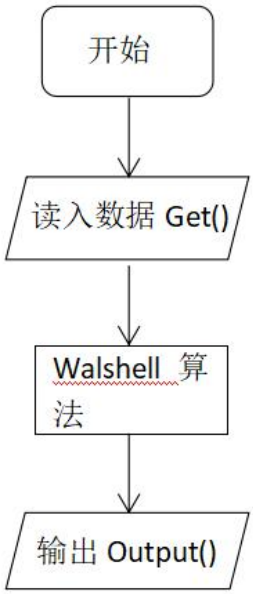

图 7 求关系的传递闭包流程图

3.3程序实现

表 3 求关系的传递闭包函数说明表

序号	名称	说明
1.	Get	用于输入数据
2.	Walshell	用于进行关系闭包实际计算
3.	Output	用于输出关系闭包矩阵

3.3.1Get

用于读入输入的数据。将所有的数据存入一个一维数组，记录总数据量；总数据量开根号即关系矩阵的阶数；最后再重新存入矩阵中。

实际上可以使用 getline()函数读取一行数据来记录矩阵阶数。

3.3.2Walshell

用于实现具体的计算关系闭包。首先建立图的关系矩阵。对于每一个点对(i,j)进行 n 次判定，第 k 步判定时，如果已经联通（M[i,j]=1）则忽略，未联通则考虑经过 xk 的通路，即 M[i,k]=M[k,j]=1 时，M[i,j]改写为 1。

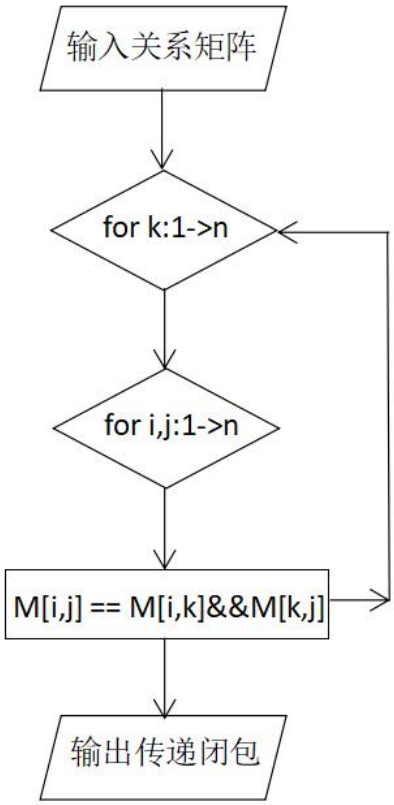

图 8 Walshall 函数执行流程

3.3.30utput

按照 Warshell 算法得到的传递闭包，直接按照顺序输出新的关系矩阵即可。

4求偏序集中的极大元与极小元

4.1概述

输入偏序集<A, £>，A 中的元素数不超过 20 个，分别用单个小写的英文字母表示。输入的第一行给出 A 中的各个元素，两个相邻的元素之间用逗号隔开，第二行给出偏序关系£，用有序对的形式给出(只给出哈斯图中的满足覆盖的两个元素形成的有序对)，如<a,b>,<c,a>等等，两个相邻的有序对之间用逗号隔开。

输出 A 的极小元与极大元。输出的第一行给出各个极小元，两个相邻元素之间用逗号隔开，输出的元素要求按照英文字母的自然顺序排列输出，第二行给出各个极大元，两个相邻元素之间用逗号隔开，输出的元素要求按照英文字母的自然顺序排列输出。

4.2步骤流程

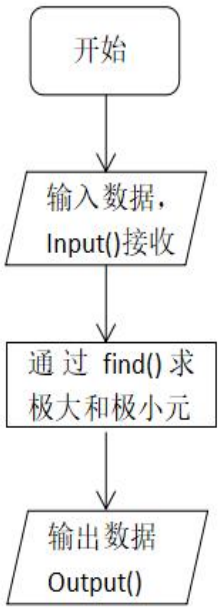

图 9 求偏序集中的极大元与极小元流程图

4.3程序实现

表 4 求偏序集中的极大元与极小元函数说明表

序号	名称	说明

序号	名称	说明
1.	search	用于将数字和字母转换,字母转数字需考虑输入顺序问题
2.	find	用于进行极大和极小的实际计算
3.	input	用于在输入字母表和偏序关系时转换
4.	output	用于输出极大和极小字母序列

4.3.1search

考虑到首个输入字母不一定为 a，转换时不能简单的用 ch-‘0’。search 用于数字和字母之间相互转换。使用时设置判定符 type, type = 1 为字母转数字，type = 2 为数字转字母。输入时用 vector<char> list 动态存储字母表。数字转字母时直接输出 list[num]即可；字母转数字时需要在 list 中寻找字母所在位置对应的数字。

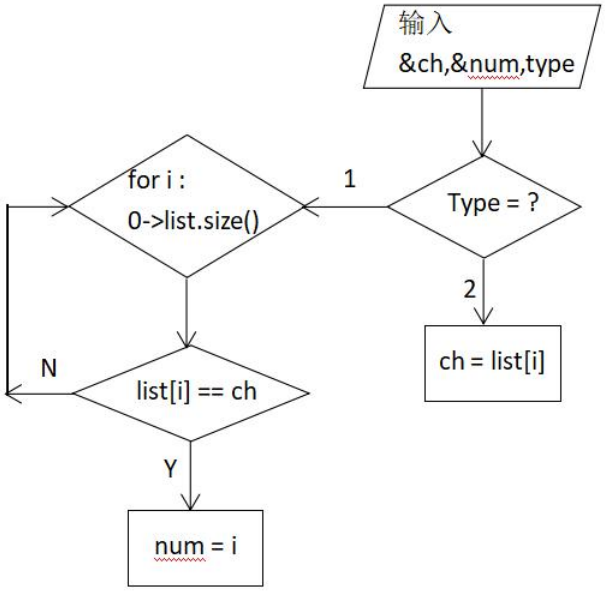

图 10 search 函数执行流程

4.3.2find

对于每个字母，如果关系矩阵中该字母所在行没有 1，则为极大；所在列没有 1，则为极小。

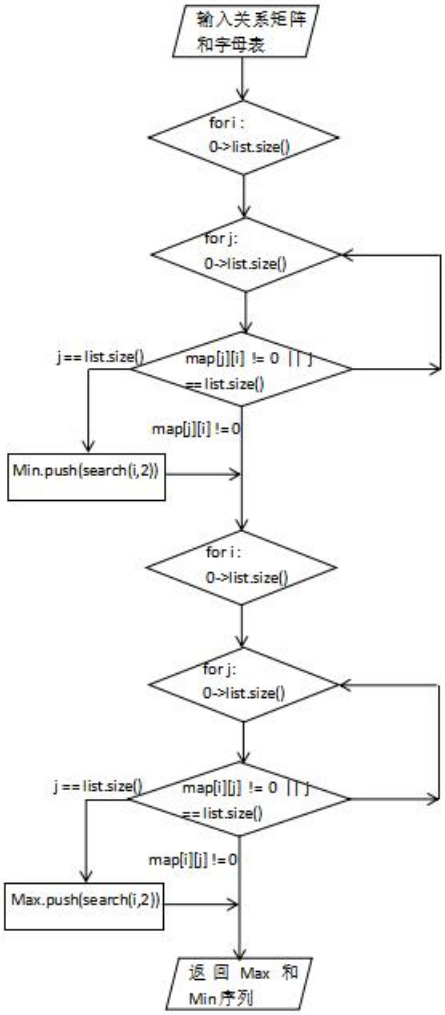

图 11 find 函数执行流程

4.3.3input

输入的字母表存到 list 当中，输入的偏序关系存入到 map 当中。其中将用到 search()函数。

4.3.4output

按照 find()函数得到的 Min 和 Max 序列，顺序输出字母。

5代数系统算律的判断

5.1概述

假设代数系统<A, *>中，A 为有限集合，*为二元运算。请判断 A 中*运算是

否满足交换律、结合律、幂等律；是否有幺元和零元，若有请输出，若无请输出n。

5.2步骤流程

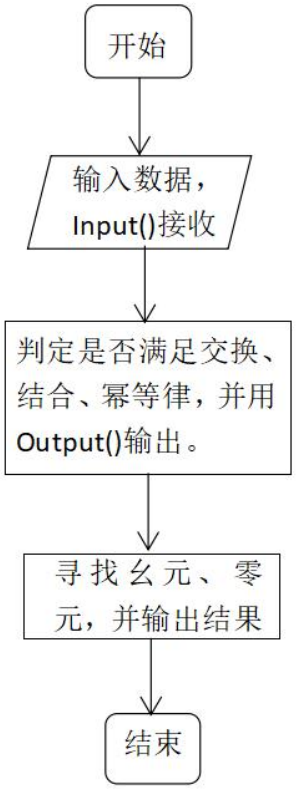

图 12 代数系统算律的判断流程图

5.3程序实现

表 5 代数系统算律函数说明表

序号	名称	说明
1.	find	用于将字母转换为对应位置的数字
2.	Input	用于输入关系列表
3.	Commutativecal	用于判定是否满足交换律
4.	Associatecal	用于判定是否满足结合律
5.	Idemponentcal	用于判定是否满足幂等律
6.	Output	用于输出前三种判定的结果
7.	Identityfind	用于寻找和输出关系的单位元

序号	名称	说明
8.	Zerofind	用于寻找和输出关系的零元

5.3.1find

用于将字母转换为位置数字。传入参数 ch，遍历记录了输入字符的数组 list，若 list[i]==ch，返回 i。对于非法值返回-1。

5.3.2Input

用于将输入的字符存入 list,将运算关系矩阵存入 map。

5.3.3Commutativecal

用于进行交换律的验证。对于关系矩阵任意非对角元素 i，j，若 map[i][j] != map[j][i]，则返回 false。若能够完成全部循环，返回 true。

5.3.4Associativecal

用于进行结合律的验证。对于关系矩阵任意三个不同元素 i,j,k,若 map[find(map[i][j])][k] != map[i][find(map[j][k])]，则返回 false。若能完成全部循环则返回 true。

5.3.5Idemponentcal

用于进行幂等律的验证。对于关系矩阵任意元素 i,若 list[i] != map[i][i]（即 a != a*a），则返回 false。若能完成全部循环，则返回 true。

5.3.6Output

用于将上述三种性质验证的结果输出。若验证结果为 false 打印 n，true 则打印 y。

5.3.7Indentityfind

用于寻找单位元。定义判断元素 int flag = 0。对于每个元素 i，如果其对任意 j，都满足 map[i][j] == list[j]，则输出该元素，并将 flag 置为 1。最后如果 flag 为 0 则输出 n 表示无单位元。

5.3.8Zerofind()

用于寻找零元。定义判断元素 int flag = 0。对于每个元素 i，如果其对任意 j，都满足 map[i][j] == list[i]，则输出该元素，并将 flag 置为 1。最后如果 flag 为 0 则输出 n 表示无零元。

6求 Zn 群中元素的阶

6.1概述

设 Zn 为模 n 整数加群(n>1)，求 Zn 中元素的阶。

输入：n，x（x 是 Zn 中的元素）

输出：x 的阶

6.2步骤流程

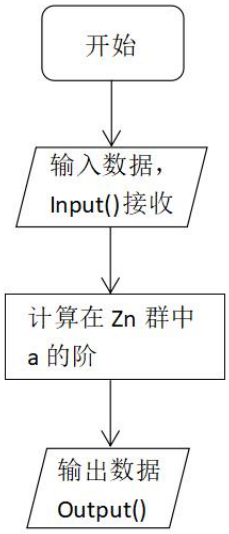

图 13 求 Zn 群中的节程序流程图

6.3程序实现

表 6 求 Zn 群中元素的阶函数说明表

序号	名称	说明
1.	Rank	用于计算 a 的阶

6.3.1Rank

用于计算 a 的阶并输出。对于给定的 a 和 Z，用 init 记录 a 的初始值，设置 list[Z]数组用于辅助计算，初始化为 0 值。然后进入循环：如果 list[a]==0，则令 list[a] = 1, a = (a + init) % Z，同时用 ans 记录进行的循环次数。循环退出后打印 ans 的值。

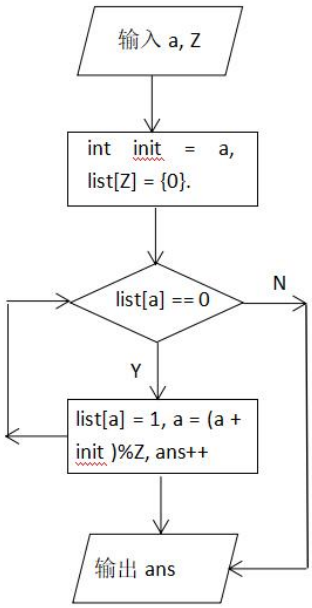

图 14 Rank 函数执行流程

7二部图的判定

7.1概述

输入：

正整数 n，代表无向图 G 的阶数；随后的 n 行代表 G 的邻接矩阵，每行有 n 个数据，每个数据以空格分隔。其中每个数据表示顶点 vi 邻接顶点 vj 边的条数。

输出：

若为二部图，输出 yes；否则，输出 no。

7.2步骤流程

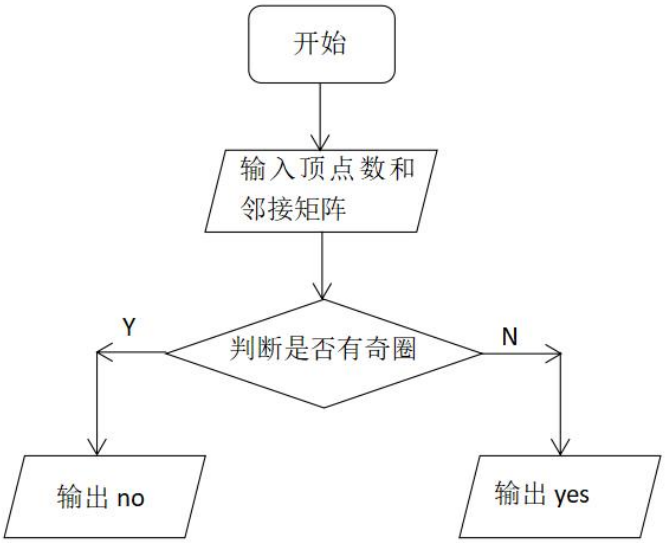

图 15 二部图的判定程序流程图

7.3程序实现

表 7 二部图的判定函数说明表

序号	名称	说明
1.	Input	用于输入数据的存储
2.	DFS	用于遍历从某一点开始的所有回路并判定是否为奇圈
3.	find	用于执行具体的判定流程
4.	Output	用于执行输出

7.3.1Input

用于输入数据的存储。将输入的顶点数存入 n，输入的邻接矩阵存入 map 二维数组。

7.3.2DFS

用于遍历从某一点开始的所有回路并判定是否为奇圈。对于传入的顶点编号（设为 i），DFS 寻找能够回到 i 点的圈。每次遍历时，用传入的参数 length 记录已经走过的距离，用 vertex 数组记录已经走过的点，用 begin 记录本次应该寻找的终点。如果存在不经过重复顶点且回

到了 i 点的偶圈，则直接返回 false。如果遍历完整个图都没有偶圈返回 true。

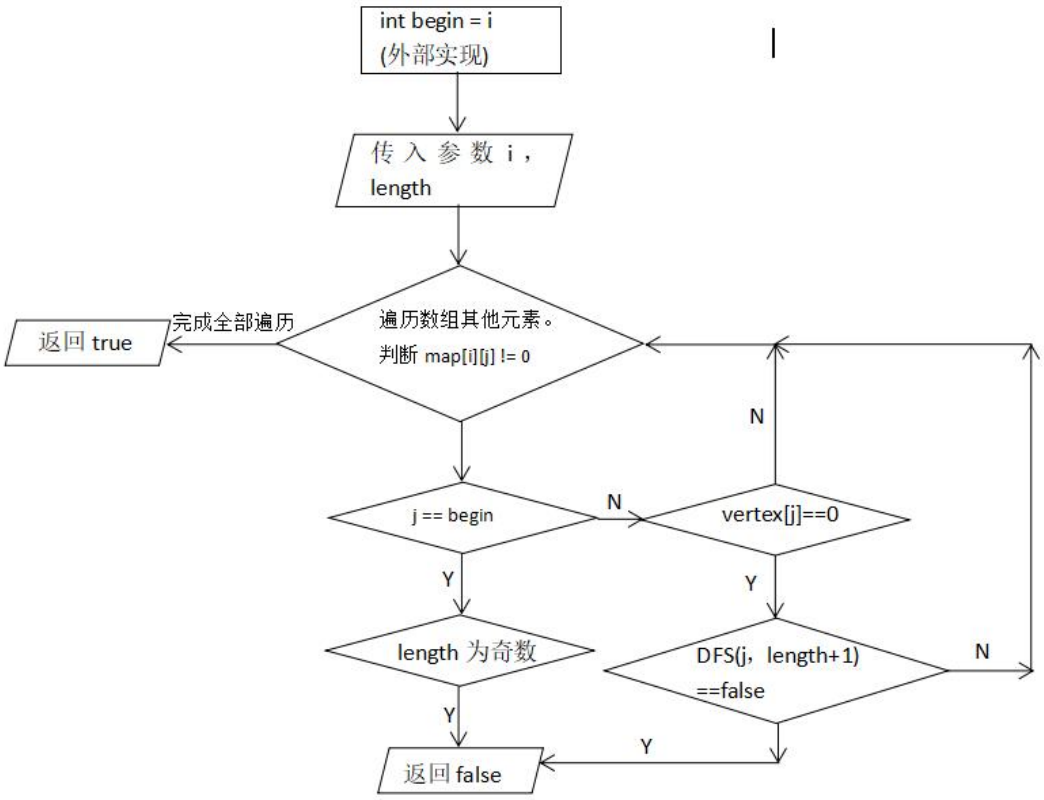

图 16 DFS 函数执行流程

7.3.3find

用于执行具体的判定流程。对于函数是否有奇圈的判定，首先检查是否存在长度为 1 的圈，即 map[i][i]==0 的情况，有则直接返回 false；而后遍历顶点，用 begin 记录每次判定时的初始起点，进行 DFS 判定，若有一个顶点返回值为 false，则返回 false。如能完成全部遍历，返回 true。

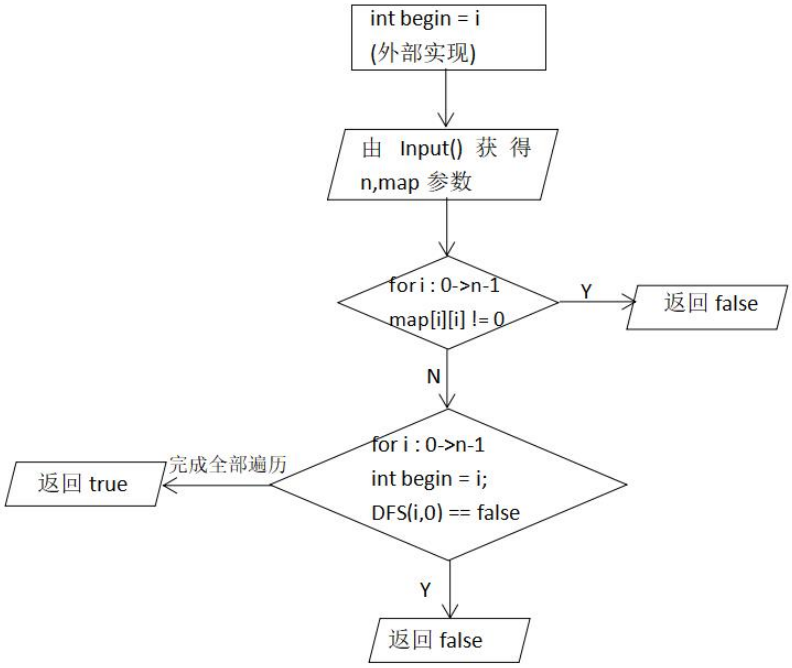

图 17 find 函数执行流程

7.3.40utput

用于执行输出。根据 find 函数的返回值，输出 yes 或 no。

8有向图连通性的判定

8.1概述

题目：判断一个图是否为强连通图、单向连通图、弱连通图。输入为有向图的邻接矩阵。

输入

第一行为正整数 N（0<N<=100），代表图中点的个数。接下来 N 行，每行有 N 个数据，每个数据以空格分隔，代表邻接矩阵。注意：输入的都是连通图。

输出

输出有一行，字母 A，B，C。A 代表强连通图，B 代表单向连通图，C 代表弱连通图。

8.2步骤流程

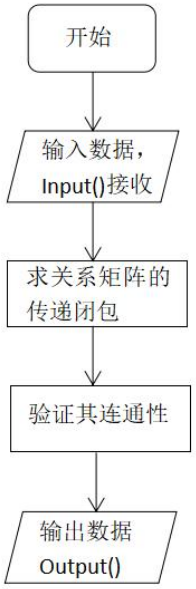

图 18 有向图连通性的判定程序流程图

8.3程序实现

表 8 有向图连通性的判定函数说明表

序号	名称	说明
1.	Input	用于输入数据的存储
2.	justify	用于计算传递闭包并验证连通性
3.	Output	用于执行输出

8.3.1Input

用于输入数据的存储。将顶点数存入 n 中，邻接矩阵存入 map 中。

8.3.2justify

用于计算传递闭包并验证连通性，该函数返回值为 1 时是强连通图，返回 2 时是单向连通图，返回 3 时是弱连通图。利用 Walshell 算法计算矩阵的传递闭包以计算每个顶点之间的连通性。然后遍历每个顶点并设置变量 flag = 1。若存在两个点 i，j 之间完全不连通，直接返回 3；i，j 之间单向联通，flag 改写为 2；如

果没有在循环过程就直接返回，函数返回 flag 的值。

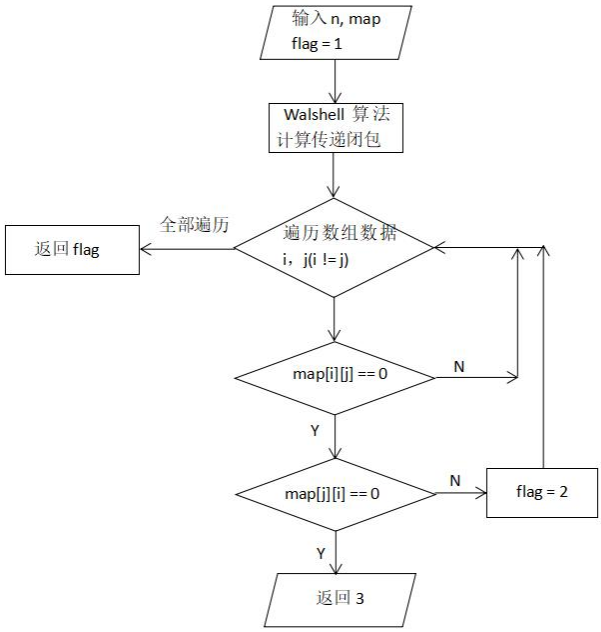

图 19 Walshell 函数执行流程

8.3.30utput

用于执行输出：接收 justify 函数返回值，返回 1 输出“A”，返回 2 输出“B”，返回 3 输出“C”。