|  |
| --- |
| 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](#_Toc3444)

[1.1 概述 1](#_Toc2359)

[1.2 步骤流程 1](#_Toc11926)

[1.3 程序实现 1](#_Toc7922)

[1.3.1 PriorityofOperator 2](#_Toc27942)

[1.3.2 evalRPN 2](#_Toc8868)

[1.3.3 calculate 3](#_Toc26633)

[1.3.4 Print 4](#_Toc21428)

[2 消解算法 4](#_Toc21716)

[2.1 概述 4](#_Toc24064)

[2.2 步骤流程 5](#_Toc21000)

[2.3 程序实现 5](#_Toc26552)

[2.3.1 alphacount 6](#_Toc12289)

[2.3.2 Init 6](#_Toc24332)

[2.3.3 connect 7](#_Toc3685)

[2.3.4 dispel 7](#_Toc3677)

[3 求关系的传递闭包 8](#_Toc23070)

[3.1 概述 8](#_Toc29386)

[3.2 步骤流程 8](#_Toc22571)

[3.3 程序实现 8](#_Toc28752)

[3.3.1 Get 9](#_Toc3002)

[3.3.2 Walshell 9](#_Toc6111)

[3.3.3 Output 9](#_Toc31350)

[4 求偏序集中的极大元与极小元 10](#_Toc5898)

[4.1 概述 10](#_Toc10227)

[4.2 步骤流程 10](#_Toc26172)

[4.3 程序实现 10](#_Toc10632)

[4.3.1 search 11](#_Toc12698)

**表目录**

[表 1 求命题的主范式函数说明表 2](#_Toc21912)

[表 2 消解算法函数说明表 6](#_Toc8350)

[表 3 求关系的传递闭包函数说明表 8](#_Toc21207)

[表 4 求偏序集中的极大元与极小元函数说明表 10](#_Toc21121)

**图目录**

[图 1 求命题的主范式流程图 1](#_Toc18711)

[图 2 evalRPN 函数执行流程 3](#_Toc32538)

[图 3 calculate 函数执行流程 4](#_Toc8769)

[图 4 消解算法流程图 5](#_Toc26826)

[图 5 Init函数执行流程 6](#_Toc18318)

[图 6 dispel函数执行流程 7](#_Toc27784)

[图 7 求关系的传递闭包流程图 8](#_Toc14968)

[图 8 Walshall函数执行流程 9](#_Toc14001)

[图 9 求偏序集中的极大元与极小元流程图 10](#_Toc8795)

[图 10 search函数执行流程 11](#_Toc679)

# 求命题的主范式

## **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。

## **步骤流程**

程序运行主逻辑如图1所示。

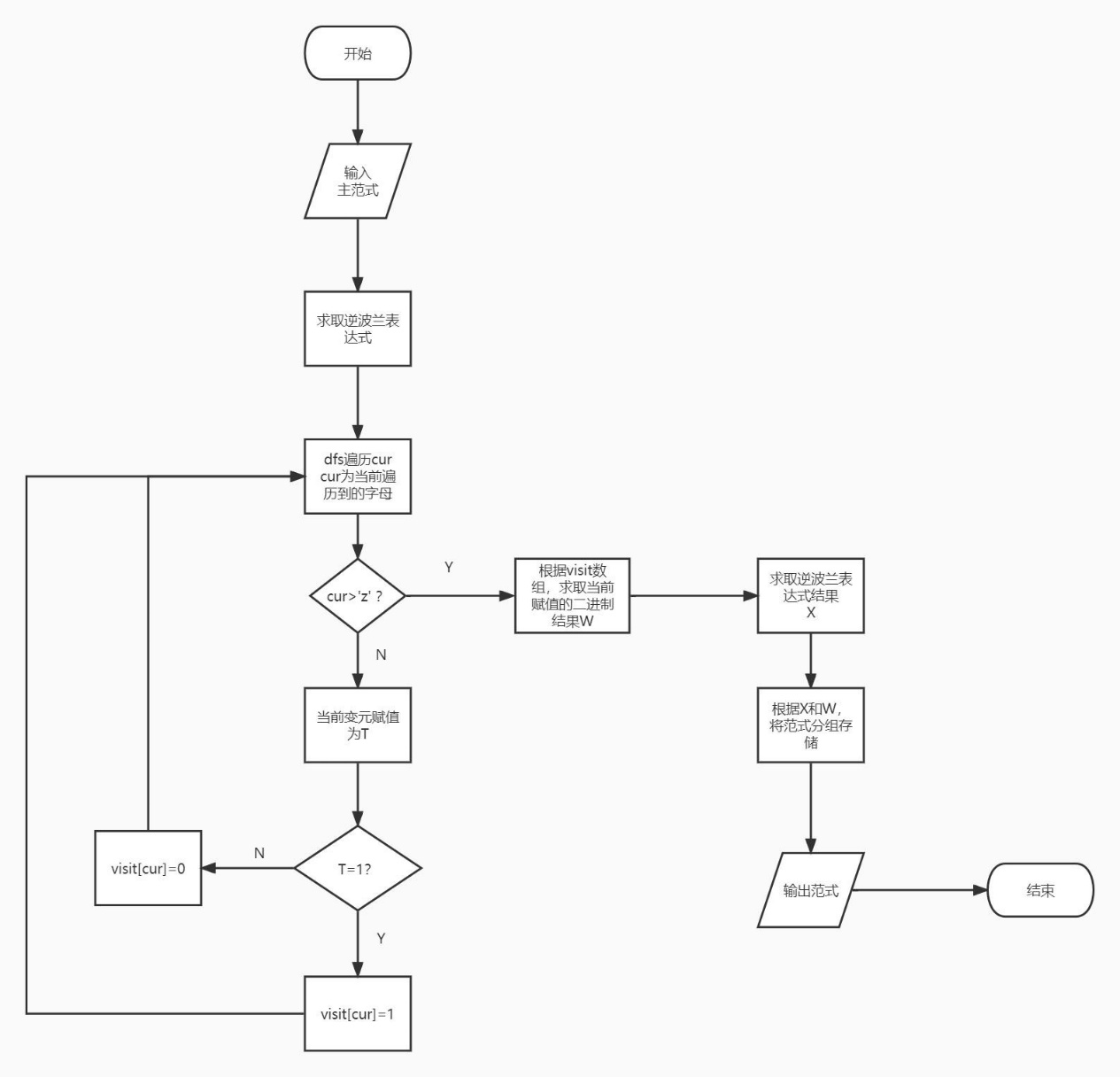


图 1 求命题的主范式流程图

## **程序实现**

表 1 求命题的主范式函数说明表

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

### **PriorityofOperator**

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。

### **evalRPN**

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。

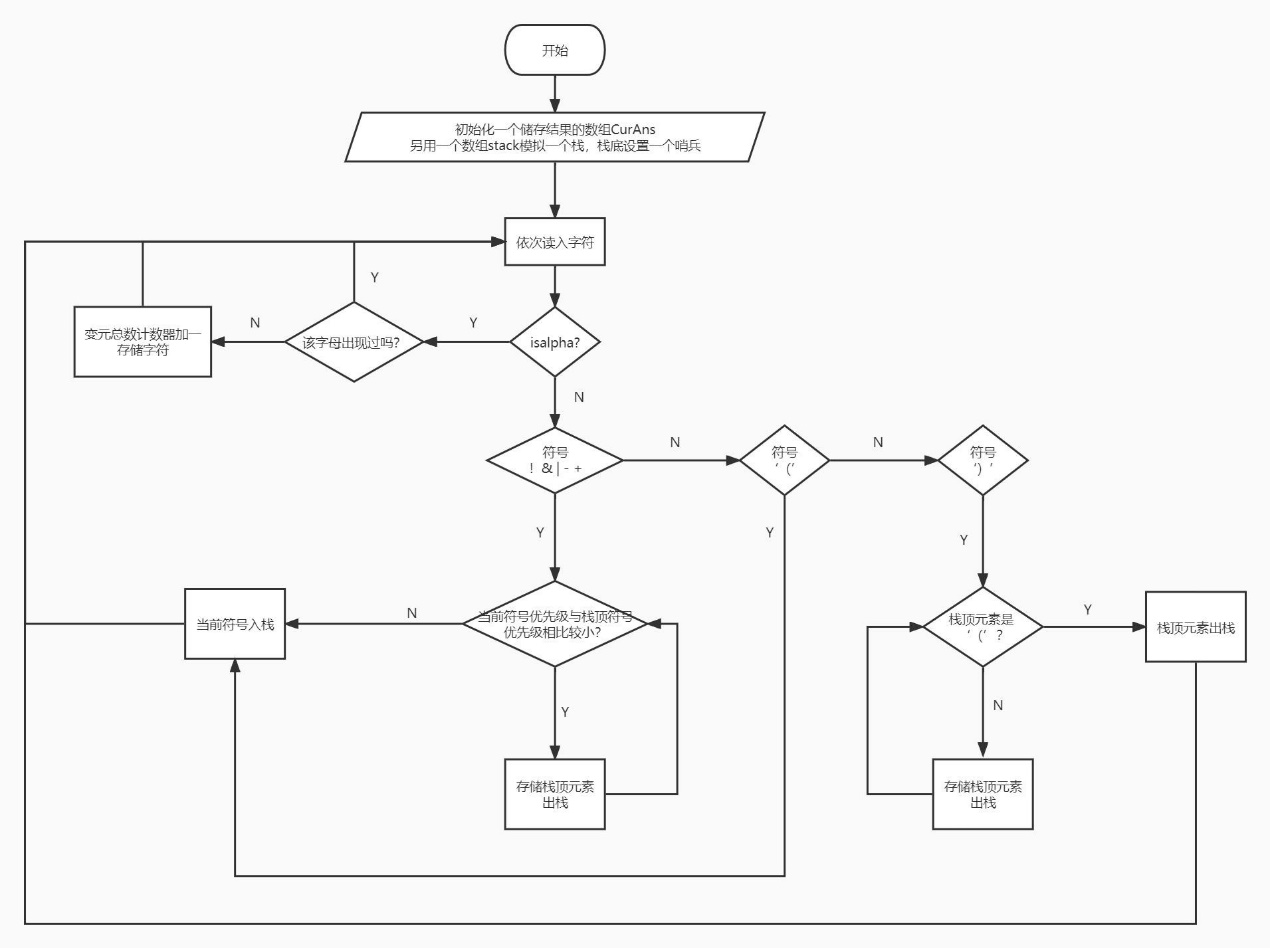


图 2 evalRPN 函数执行流程

### **calculate**

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。

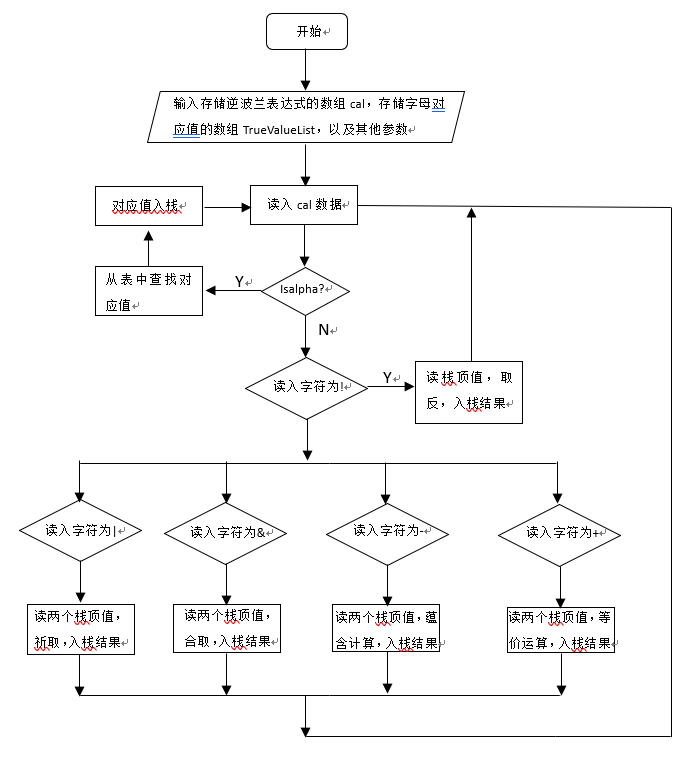


图 3 calculate 函数执行流程

### **Print**

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。

# 消解算法

## 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”。

## 步骤流程

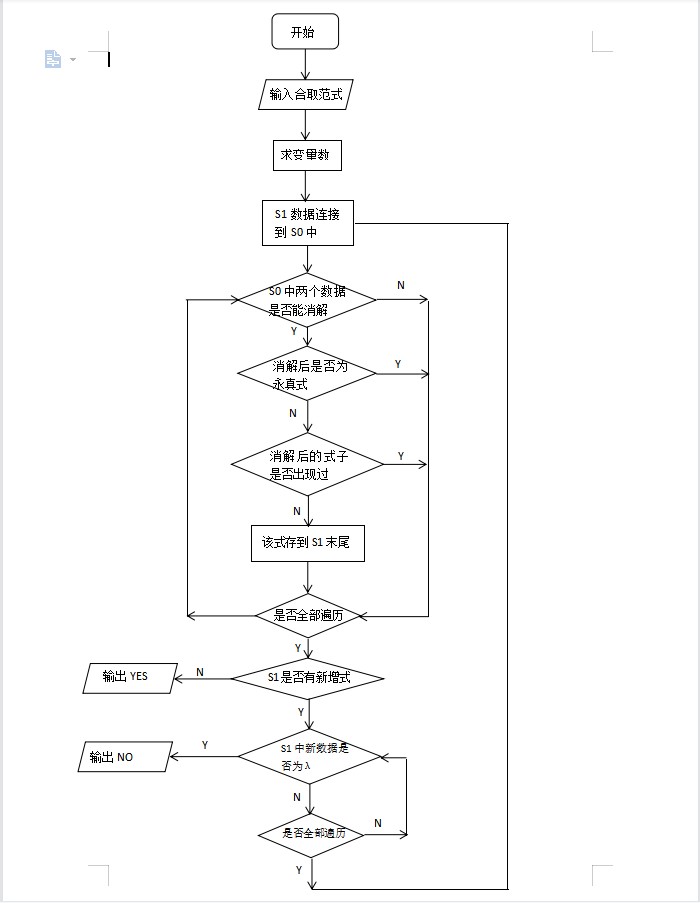


图 4 消解算法流程图

## 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 |

### **alphacount**

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。

### **Init**

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

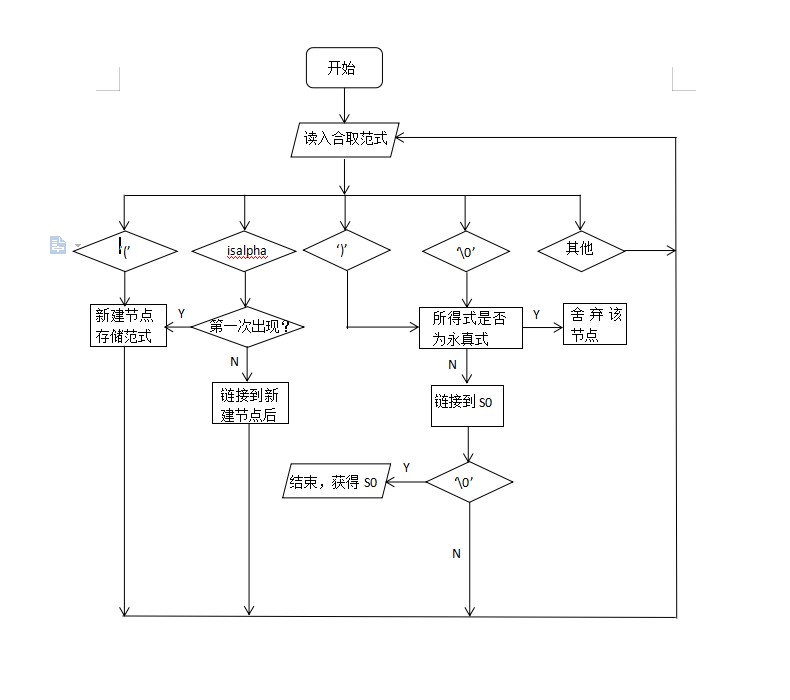


图 5 Init函数执行流程

### **connect**

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

### **dispel**

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.

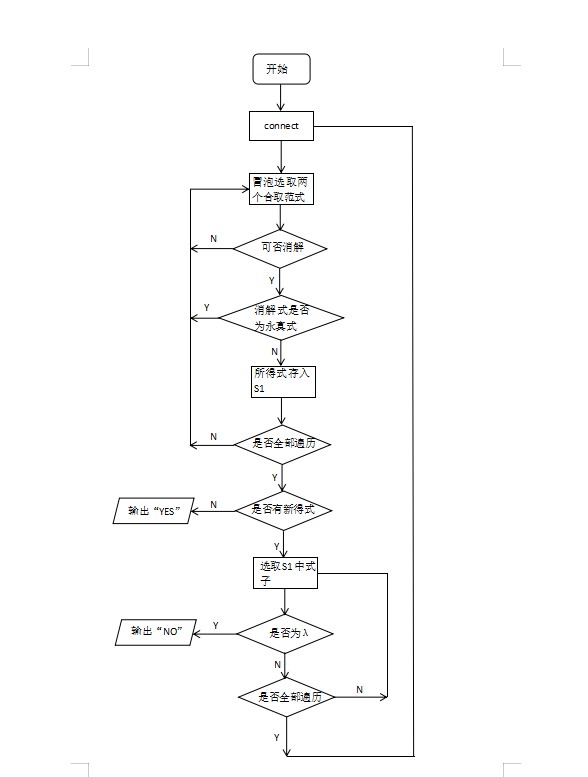


图 6 dispel函数执行流程

# 求关系的传递闭包

## **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。

## **步骤流程**

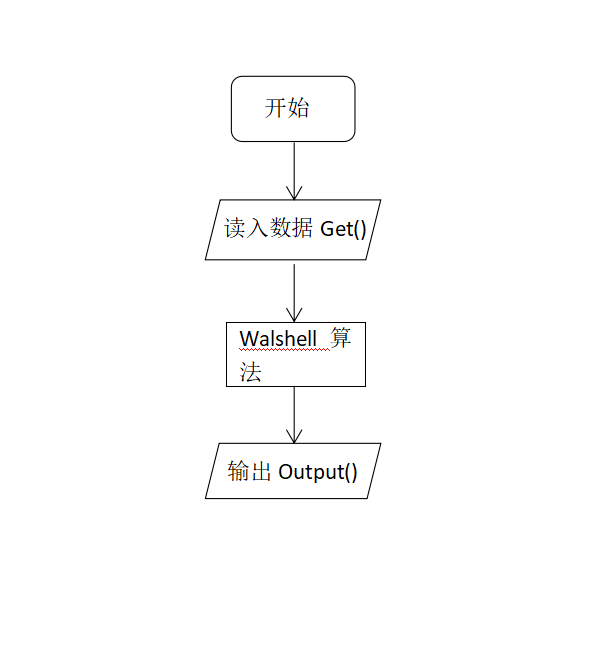


图 7 求关系的传递闭包流程图

## **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 |

### **Get**

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。

### **Walshell**

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。

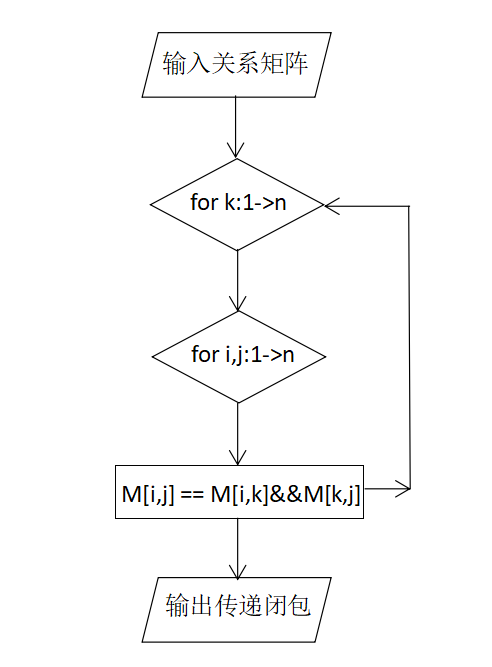


图 8 Walshall函数执行流程

### **Output**

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。

# 求偏序集中的极大元与极小元

## 概述

首先输入字母序列和偏序关系序列，建立关系矩阵。对于每个偏序关系<ch1,ch2>,将map[ch1- ‘0’][ch2- ‘0’]改写为1。然后遍历每一行和列。对于每个ch1,ch2,有map[ch1- ‘0’][ch2- ‘0’]==1，则ch1不可能为极小，ch2不可能为极大。但对于一行均为0的ch1，其为极大；一列均为零的ch2，其为极小。最后输出极大和极小序列即可。

## 步骤流程

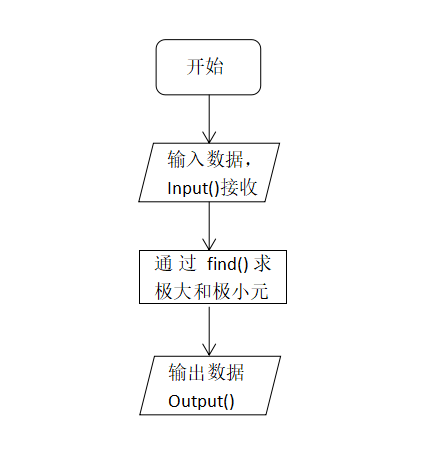


图 9 求偏序集中的极大元与极小元流程图

## 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 |
| --- | --- | --- |
| 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 |

### search

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。

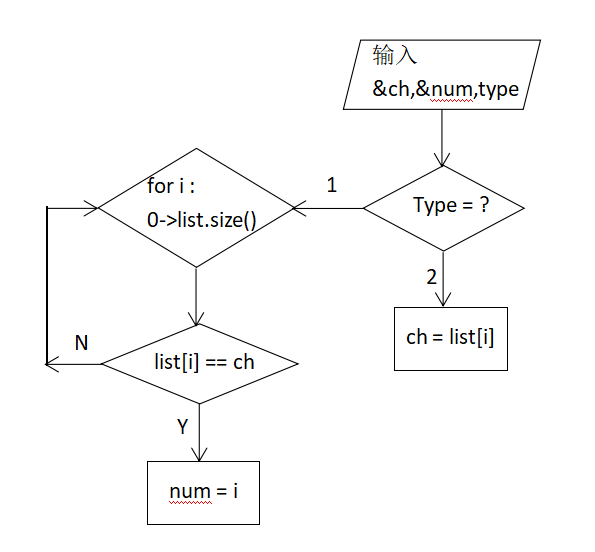


图 10 search函数执行流程

### find