* **题目：先序递归过程建立二叉树**
* **班级：2021211312 姓名：杨天宇 学号：2021211281** 分工： **1,2**
* **班级：2021211312 姓名：王昊钰 学号：2021211282** 分工： **3,4**
* **班级：2021211312 姓名：李志茂 学号：2021211289** 分工： **5,6**

1. 需求分析

本程序旨在实现以三元组表作为存储结构的稀疏矩阵的加法和乘法运算，并用另一个三元组表存放运算结果，程序的功能分为两部分：第一部分为判断两矩阵是否可加或可乘，第二部分为计算两矩阵相加或相乘的结果。输入内容为两个完整的三元组表；若两矩阵可加，则输出两矩阵的和矩阵的三元组表，否则输出“They can’t add together”，若两矩阵可乘，则输出两矩阵的乘积矩阵的三元组表，否则输出“They can't multiple together”，输入、输出的数据不超过int类型的存储范围。

1. 概要设计

解决问题的大致步骤（主程序的流程）为：输入两个以三元组表形式表示的矩阵，然后以此判断它们是否可加，若是则进行相加的运算，然后以此判断它们是否可乘，若是则进行相乘的运算，最后输出结果。程序的数据结构类型主要有：输入两矩阵的三元组表、和矩阵和乘积矩阵的三元组表，主程序通过调用三元组表输入、三元组表输出、判断是否可加、判断是否可乘、矩阵相加和矩阵相乘六个子程序完成任务。

3、详细设计

数据类型：单个元素element ，三元组表sparmattp

struct element

{

int i, j; // i行j列

int value; // 存储的值

};

struct sparmattp

{

int m, n, t; // 矩阵m行n列,t个非0元素

element \*data; // 实际矩阵所在位置

};

Add：//两矩阵相加

{ l = 0, m = 0, n = 0;

while (m < a->t && n < b->t)

{

if (a->data[m].i == b->data[n].i && a->data[m].j==b->data[n].j) // a,b在同一个位置

{

c->data[l].i = a->data[m].i;

c->data[l].j = a->data[m].j;

c->data[l].value = a->data[m].value + b->data[n].value; //两个值相加

l++, m++, n++;

}

else if (a->data[m].i \* m + a->data[m].j > b->data[n].i \* m + b->data[n].j) // a的位置在b的后面

{

c->data[l] = b->data[n];

l++, n++;

}

else // a的位置在b前面

{

c->data[l] = a->data[m];

l++, m++;

}

}

while (m < a->t) //将a表中剩余部分接入c

{

c->data[l] = a->data[m];

l++, m++;

}

while (n < b->t) //将b表中剩余部分接入c

{

c->data[l] = b->data[n];

l++, n++;

}

c->t = l;

c->m = a->m;

c->n = a->n;//获得相加后的数组

}

multiple//两矩阵相乘

{

while (m < a->t)

{

n = 0;

while (n < b->t)

{

if (a->data[m].j == b->data[n].i)

{

d[a->data[m].i - 1][b->data[n].j - 1] += ((a->data[m].value) \* (b->data[n].value));

// 列号= 行号 ，对应位置+=

}

n++;

}

m++;

}

for (i = 0; i < a->m; i++)

{

for (j = 0; j < b->n; j++)

{

if (d[i][j] != 0)

{

c->data[l].i = i + 1;

c->data[l].j = j + 1;

c->data[l].value = d[i][j];

l++;

}

}

}//将非0位放入三元组表

c->t = l;

c->m = a->m;

c->n = b->n;

}

judge\_add //判断能否相加

{

int flag = 0;

if (a->m != b->m)

flag = 1;

if (a->n != b->n)

flag = 1;

return flag; //若行列号相同则可以相加

}

judge\_multiple/判断能否相乘

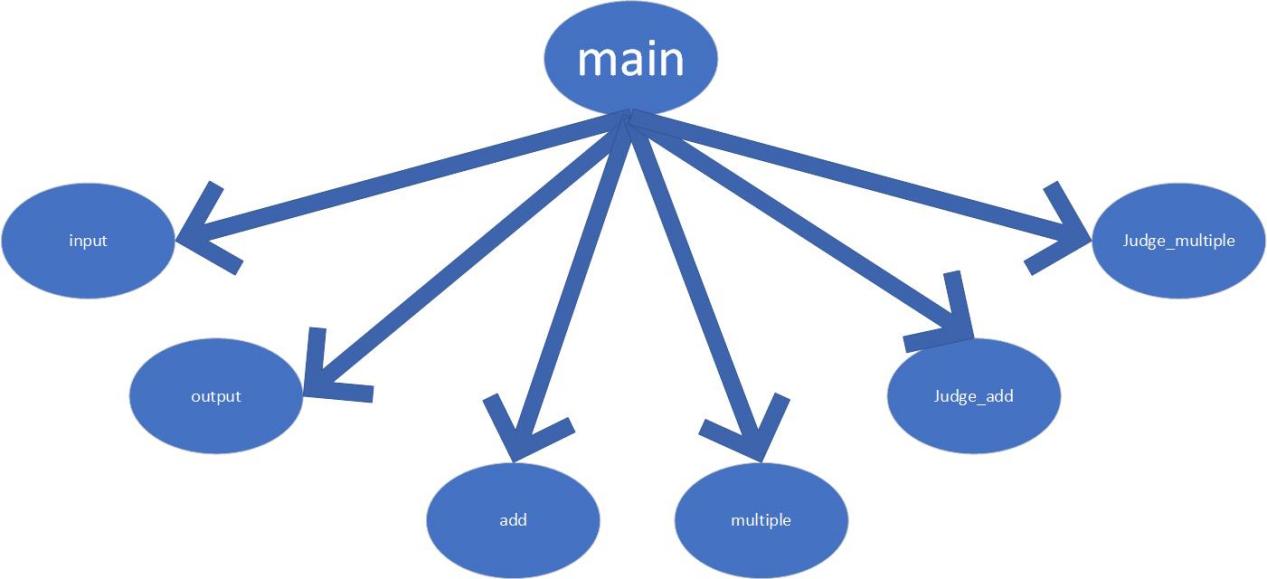
{

int flag = 0;

if (a->n != b->m)

flag = 1;

return flag; //判断前一个矩阵的列是否等于后一个的行

}

4、调试分析报告

遇到的问题：

相乘后从矩阵到三元组表出现错误。将相乘结果输入矩阵时没有-1，导致结果发生偏移。在将乘积输入矩阵时行号列号-1，输出时+1解决。

回顾讨论：

通过按照矩阵相加相乘定义实现。

假设相加两表长度分别为m和n

add 时间复杂度：O(m+n);空间复杂度：O(m+n);

multiple时间复杂度：O（mn）；空间复杂度O（mn）

经验和体会：对矩阵相加和相乘的定义需要明晰，数学基础很重要。将操作模块化之后有助于调试与修正。

5、用户使用说明

编译运行main.cpp,输入矩阵A的行数m、列数n、有效值t(m n t)，稍后按三元组表形式输入所有有效值的行i、列j、值v（i j v）,再按重复上述流程输入矩阵B的相关参数，最后程序会输出判断。若矩阵A和B无法相加/相乘，则输出“They can’t add together.”/”They can’t multiple together.”,否则输出“Add matrix”/“Multiple matrix”以及对应和矩阵/积矩阵。

6、测试结果

main.cpp 测试用例

输入用例1：

7 8 27

1 3 6

1 5 7

1 7 3

2 1 1

2 3 4

2 5 4

2 7 6

3 1 3

3 3 6

3 5 4

3 7 3

4 1 7

4 3 4

4 5 7

4 7 1

5 1 2

5 3 6

5 5 2

5 7 1

6 1 1

6 3 7

6 5 6

6 7 1

7 1 2

7 3 2

7 5 4

7 7 1

8 6 23

1 3 5

1 5 6

2 1 5

2 3 5

2 5 1

3 1 3

3 3 7

3 5 7

4 1 7

4 3 8

4 5 1

5 1 7

5 3 6

5 5 1

6 1 3

6 3 3

6 5 7

7 1 4

7 3 4

7 5 1

8 1 1

8 3 1

8 5 4

输入用例2：

5 7 16

1 2 2

1 3 5

1 7 5

2 1 3

2 2 5

2 6 3

2 7 4

3 1 2

3 5 3

3 6 5

3 7 1

4 4 1

4 5 3

4 6 3

5 3 1

5 4 4

5 7 18

1 2 5

1 3 1

1 5 5

1 7 2

2 2 2

2 4 5

2 6 2

3 1 3

3 3 3

3 5 2

3 7 5

4 2 5

4 4 4

4 6 5

5 1 5

5 3 4

5 5 3

5 7 3

输出结果1：

They can’t add together

Multiple matrix

row 1

column 1

value 79

row 1

column 3

value 96

row 1

column 5

value 52

row 2

column 1

value 64

row 2

column 3

value 81

row 2

column 5

value 44

row 3

column 1

value 58

row 3

column 3

value 93

row 3

column 5

value 67

row 4

column 1

value 65

row 4

column 3

value 109

row 4

column 5

value 78

row 5

column 1

value 36

row 5

column 3

value 68

row 5

column 5

value 57

row 6

column 1

value 67

row 6

column 3

value 94

row 6

column 5

value 62

row 7

column 1

value 38

row 7

column 3

value 52

row 7

column 5

value 31

输出结果2：

Add matrix

row 1

column 2

value 7

row 1

column 3

value 6

row 1

column 5

value 5

row 1

column 7

value 7

row 2

column 1

value 3

row 2

column 2

value 7

row 2

column 4

value 5

row 2

column 6

value 5

row 2

column 7

value 4

row 3

column 1

value 5

row 3

column 3

value 3

row 3

column 5

value 5

row 3

column 6

value 5

row 3

column 7

value 6

row 4

column 2

value 5

row 4

column 4

value 5

row 4

column 5

value 3

row 4

column 6

value 8

row 5

column 1

value 5

row 5

column 3

value 5

row 5

column 4

value 4

row 5

column 5

value 3

row 5

column 7

value 3

They can't multiple together