实验报告

一.实验目的

编写一个程序，计算稀疏矩阵的和。

二.实验内容

1.设计程序的数据结构与算法

2.实现通过三元组表示稀疏矩阵

3.实现通过三元组使稀疏矩阵相加

三、实验环境

编程语言：c++

编译环境：visual studio code，gcc14.2.0，-std=c++20 -O2 -fPIC -Wall -fno-asm lm march=native -Wl,-stack=268435456

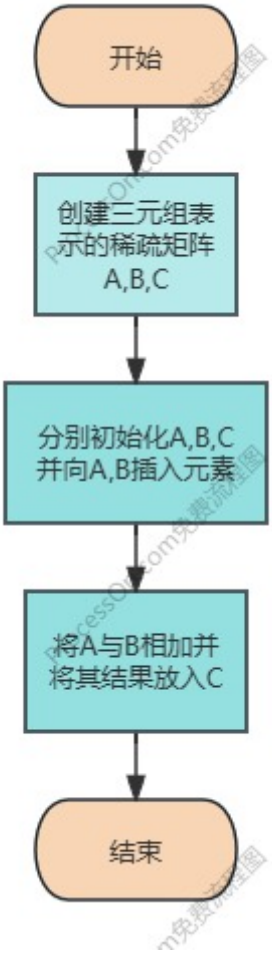
操作系统：Windows11

四、问题描述

假设稀疏矩阵A和B（具有相同大小的m*n）都采用三元组存储，编写程序计算C=A+B，要求C也采用三元组存储

五、主程序流程和函数说明

主函数流程



函数说明

`InitMatrix(matrix,rows,cols)`

- **描述:** 初始化一个稀疏矩阵
- **用途:** 在主函数中用于根据初始化表示行列为 `rows` 与 `cols` 的稀疏矩阵的三元组 `matrix`

`AssignValue(matrix,row,col,value)`

- **描述:** 将一个元素插入稀疏矩阵
- **用途:** 在主函数中，将值为 `value` 的元素插入矩阵 `matrix` 中的(`row` , `col`)位置

```
void AssignValue(TSMatrix &matrix, int row, int col, Elemtype value)
{
    if (matrix.tu >= MAXSIZE)
    {
        cout << "Matrix exceeds maximum allowed size!" << endl;
        return;
    }

    if (row <= 0 || row > matrix.mu || col <= 0 || col > matrix.nu)
    {
        cout << "The index is out of the matrix and the insert fails" << endl;
        return;
    }

    if (value == 0)
    {
```

```
        return;
    }

    // 遍历查找插入位置或更新点
    int pos = 1;
    while (pos <= matrix.tu)
    {
        if (matrix.data[pos].i == row && matrix.data[pos].j > col)
        {
            cout << "The element already exists in this position, and the
insertion fails" << endl;
            return;
        }
        if (matrix.data[pos].i > row || (matrix.data[pos].i == row &&
matrix.data[pos].j > col))
        {
            break;
        }
        pos++;
    }

    for (int i = matrix.tu; i >= pos; i--)
    {
        matrix.data[i + 1] = matrix.data[i];
    }
    matrix.data[pos].i = row;
    matrix.data[pos].j = col;
    matrix.data[pos].e = value;
    matrix.tu++;
}
```

`AddMatrices(matrixA,matrixB, matrixC)`

- **描述:** 将两个稀疏矩阵相加
- **用途:** 将稀疏矩阵A与B相加，并将结果记录到C中

```
void AddMatrices(const TSMatrix &A, const TSMatrix &B, TSMatrix &C)
{
    if (A.mu != B.mu || A.nu != B.nu)
    {
        cout << "Matrices sizes do not match!" << endl;
        return;
    }

    C.mu = A.mu;
    C.nu = A.nu;
    C.tu = 0;

    int a = 1, b = 1, c = 1;
    while (a <= A.tu && b <= B.tu)
    {
        if (A.data[a].i == B.data[b].i && A.data[a].j == B.data[b].j)
        {
            C.data[c] = A.data[a];
```

```
            C.data[c].e += B.data[b].e;
            if (C.data[c].e == 0)
            {
                c--;
            }
            a++;
            b++;
            c++;
        }
        else if (A.data[a].i < B.data[b].i || (A.data[a].i == B.data[b].i &&
A.data[a].j < B.data[b].j))
        {
            C.data[c] = A.data[a];
            a++;
            c++;
        }
        else if (B.data[b].i < A.data[a].i || (B.data[b].i == A.data[a].i &&
B.data[b].j < A.data[a].j))
        {
            C.data[c] = B.data[b];
            b++;
            c++;
        }
    }

    // 处理剩余元素
    while (a <= A.tu)
    {
        C.data[c] = A.data[a];
        a++;
        c++;
    }
    while (b <= B.tu)
    {
        C.data[c] = B.data[b];
        b++;
        c++;
    }

    C.tu = c - 1; // 更新非零元素总数
}
```

`PrintMatrix(matrix)`

- **描述:** 打印稀疏矩阵
- **用途:** 打印稀疏矩阵的大小，并以三元组的形式打印稀疏矩阵的元素

完整代码

```
#include "MatricesFunction.cpp"

int main()
{
    TSMatrix A,B,C;
```

```
    //TSMatrix A
    InitMatrix(A, 3, 3);

    AssignValue(A, 1, 1, 5);
    AssignValue(A, 1, 2, 3);
    AssignValue(A, 2, 3, 7);

    PrintMatrix(A);

    //TSMatrix B;
    InitMatrix(B, 3, 3);

    AssignValue(B, 3, 2, 5);
    AssignValue(B, 2, 3, 3);
    AssignValue(B, 3, 3, 7);

    PrintMatrix(B);

    //TSMatrix C;

    InitMatrix(C, 3, 3);

    AddMatrices(A, B, C);

    PrintMatrix(C);

    return 0;
}
```

六、实验调试、测试样例与结果分析

测试用例

1. 创建稀疏矩阵A,B,C
2. 为A,B赋值

A

3	**3**	**3**
1	1	5
1	2	3
2	3	7

B

3	**3**	**3**

3	**3**	**3**
3	2	5
2	3	3
3	3	7

3.展示其内容

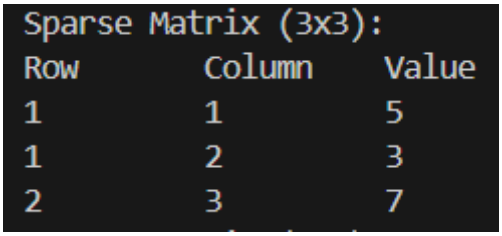
4.将A,B相加并将结果赋给C

5.展示C的内容

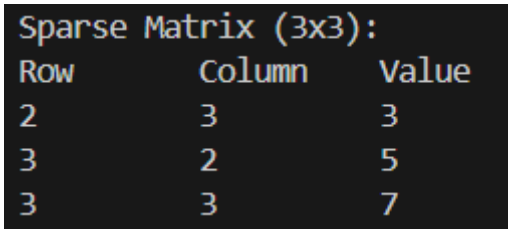
测试结果

创建稀疏矩阵并为A,B赋值后

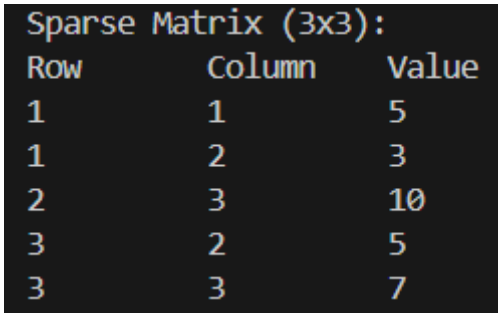
矩阵A



矩阵B



将A与B相加后得到矩阵C



七、小组成员任务分配

李坤霖：基础算法及结构体设计

姜高峰： 主要函数功能的实现

刘镇东：测试用例的编写及注释完善

八、实验改进意见与建议

1.采用面向对象编程加强规范性

2.用更好的形式展现矩阵

九、附录与说明

无