北京郵電大學

实验报告

实验名称: 三元组表表示稀疏矩阵及其运算

班级 : 2019211309 姓名 : 陈悦 学号 : 2019211413 分工 : 文档

班级 : 2019211309 姓名 : 马晓亮 学号 : 2019211400 分工 : 代码

2020 年 *10* 月 *27* 日

一　需求分析

本程序可以完成两个矩阵相乘或相减的计算任务。输入一个矩阵的格式为

$m\ n\ p$

$i_1\ j_1\ val_1$

$i_2\ j_2\ val_2$

$i_3\ j_3\ val_3$

.....

$i_p\ j_p\ val_p$

m 表示矩阵的行数，n 表示矩阵的列数，p 表示稀疏矩阵中结点的个数。

本程序要求用户输入两个矩阵 A、B，然后再选择计算方式。然后程序会在后端进行计算，将计算结果保存在矩阵 C 中。最后程序将 C 输出到终端，输出格式同样使用三元组进行输出，顺序为先列后行。

同时，我们对于用户的输入作出如下限制：

1. $n > 0$ 且 $m > 0$ 且 $p >= 0$
2. 对于一个矩阵，数值的下标应该保持在矩阵范围内（从 1 开始），不得超出矩阵。
3. 必须要输入两个矩阵后再进行运算。

对于不符合输入限制要求的操作，程序将在终端上输出 (Input Error)

同时，用户应该对输入的矩阵是否能进行运算进行检查。如果输入的矩阵不能进行运算，程序将返回 (Math Error). 可能的原因分别有：

1. Error Code 0: 相加的两个矩阵的行数与列数不相等。
2. Error Code 1: 两个矩阵不能相乘

二　概要设计

程序由三个模块组成，分别为 *Matrix*,*Solution* 和 *main*

Matrix 是一种数据类，定义了稀疏矩阵类以及它的运算方法，包括在矩阵中插入元素。

Solution 是可以与用户交互的类，定义了如何输入和输出，以及调用 *Matrix* 中的方法完成计算。

main 为程序的主函数，主要用于初始化 *Solution* 和调用 *Solution* 中与用户交互的方法。主程序的流程如下:

1. 初始化 *Solution*
2. 调用输入方法
3. 调用计算方法
4. 调用输出方法
5. 释放内存

三　详细设计

Algorithm 1 Solution 结构定义

```
1: 读入需要被操作的矩阵
2: function readMatrix(void)
3:     初始化矩阵 A
4:     读入 m,n,size
5:     for 1 → size do
6:         读入三元组，并插入到矩阵 A 中
7:     end for
8:     初始化矩阵 B
9:     读入 m,n,size
10:    for 1 → size do
11:        读入三元组，并插入到矩阵 B 中
12:    end for
13: end function
14: 读入运算类型并得出运算结果
15: function operate(void)
16:    从用户处读入运算类型，保存在 op 中
17:    if op = 0 then
18:        result = X + Y
19:    else
20:        result = X * Y
21:    end if
22: end function
23: 打印运算结果
24: function printMatrix(void)
25:    输出 Result 矩阵
26: end function
```

Algorithm 2 Matrix 结构定义

```
将一个三元组插入到矩阵中
function INSERT(Tuple node)
    检查 Matrix 是否超限
    for Matrix 所有的三元组 do
        if 第 i 个三元组与 node 在同一个位置 then
            将 node 的数值加到当前三元组数值上
        end if
    end for
    if Matrix 中没有该三元组 then
        在三元组尾部插入 node
    end if
end function
function PRINTMATRIX(Matrix X)
    输出 Result 矩阵
end function
```

Algorithm 3 Matrix 结构定义

```
1: 将当前矩阵与矩阵 X 相加，返回结果
2: function OPERATOR +(Matrix X)
3:     检查是否符合相加的条件
4:     初始化 result 矩阵，保存计算结果
5:     将 log1 指向左矩阵的头，log2 指向右矩阵的头
6:     while log1 没有达到尾部 ||log2 没有达到尾部 do
7:         if log1 达到了尾部 then
8:             将 log2 指向的元素加入 result 并递增 log2
9:         end if
10:        if log2 达到了尾部 then
11:            将 log1 指向的元素加入 result 并递增 log1
12:        end if
13:        if log1 指向的三元组与 log2 指向的三元组位置相同 then
14:            将两个三元组相加后加入到 result，同时递增两个指针
15:        end if
16:        if log1 指向的三元组在 log2 前面 then
17:            将 log1 指向的元素加入 result 并递增 log1
18:        else
19:            将 log2 指向的元素加入 result 并递增 log2
20:        end if
21:    end while
22: end function
```

Algorithm 4 Matrix 结构定义

```
1: 将当前矩阵右乘矩阵 X，返回结果
2: function OPERATOR *(Matrix X)
3:     判断两矩阵是否可以相乘
4:     将 cur 指向左矩阵的开头
5:     初始化矩阵 result 保存结果
6:     for i = 1 → n do
7:         递增 cur 直到 cur ≤ i
8:         for j = 1 → m do
9:             初始化 temp 三元组，保存乘法的结果
10:            for index = cur 其中 index 小于边界且 index 处三元组的行等于 i do
11:                for index_x = 1tosize do
12:                    if 三元组 index_x 可以与三元组 index 相乘 then
13:                        将三元组 index_x 与三元组 index 相乘，结果加入到 temp 中
14:                    end if
15:                end for
16:            end for
17:            if 判断 temp 结果非 0 then
18:                将 temp 插入到 result 中
19:            end if
20:        end for
21:    end for
22: end function
```

四　调试分析报告

在本次实验中我们继续沿用之前设计的 Vector 类。因为我们对矩阵相乘后的结果矩阵的大小未知，同时为了节省内存，避免开出 n*m 大小的矩阵。如果这样我们使用稀疏矩阵将会毫无意义。同时我们对计算过程做了优化，在插入 AB 矩阵的所有数据后，我们会对 AB 矩阵中的三元组进行排序。排序顺序表现为矩阵中个个元素的位置从前往后，即先比较行后比较列。

在经过这些优化后，我们得到的加法算法的复杂度为 $O(size_A + size_B)$。

乘法则较为复杂, 优化后，cur 指针每次递增次数的期望为 $\sqrt{size_A}$，N 与 M 表示结果矩阵的列数与行数乘法复杂度为 $O(NM * \sqrt{size_A} * size_B)$

五　用户使用说明

输入说明:

用户先根据提示输入 m,n,size，分别代表 A 矩阵的行数，列数与非空元素个数;

再输入 size 行，每行输入非空元素的坐标 i，j 与值 num。

根据提示重复上述操作，输入 B 稀疏矩阵。(每行的数字用空格分隔)

再输入操作码 (0/1)，0 代表 A、B 矩阵相加，1 代表 A、B 矩阵相乘。

输出说明:

若不能进行运算，则输出”B Matrix can’t add to A.” 或”B Matrix can’t multiply to A.”

若能进运算，则输出的第一行为运算后矩阵的行 m，列 n，与非空元素个数 size。

后面输出 size 行代表每个非空元素的坐标 i，j 与值 num。

用户可以使用 IDE 或者手动编译源代码 main.cpp，获得可执行文件。

另外，笔者使用的 gcc 版本为 8.1.0。

运行可执行文件后，用户可以选择文件输入或者交互式输入 (1/0)。

在选择文件输入模式下，输入将重定向到 in.txt, 输出将重定向到 output.txt 文件。

结束程序可以输入 EOF

六　测试结果

用于测试的样例如下:

Sample1 和 2 对应矩阵乘法加法失效的情况

Sample3 和 4 对应矩阵乘法加法正常运算的结果

Sample 1:

Input:

3 4 2

2 1 21

3 4 34

3 4 3

1 2 12

2 1 21

3 2 32

1

Output:

B Matrix can't multiply to A.

Sample 2:

Input:

2 3 3

1 1 11

1 3 13

2 2 22

3 2 3

1 1 66

2 3 88

3 3 33

0

Output:

B Matrix can't add to A.

Sample 3:

Input:

2 2 2

1 1 11

2 2 22

2 2 2

1 2 12

2 1 21

1

Output:

2 2 4

1 2 132

2 1 464

Sample 4:

Input: 3 3 5

1 1 11

1 3 13

2 2 22

3 2 32

3 3 33

3 3 3

1 1 66

2 3 88

3 3 33

0

Output:

1 1 77

1 3 13

2 2 22

2 3 88

3 2 32

3 3 66