# Lab0：稀疏矩阵乘法

## 一、实验介绍

矩阵(Matrix)是常用的数学工具。在计算机中，可以直观地将$M$行$N$列的矩阵存储为一维或者二维数组，其空间复杂度为$O(MN)$。

稀疏矩阵(Sparse Matrix)是一类特殊的矩阵，其特点是绝大部分元素的数值为0，且分布没有规律。如果仍用上述方式存储稀疏矩阵，数组中绝大部分元素是0，造成了不必要的空间浪费。

在本实验中，你需要通过某些更高效的数据结构存储稀疏矩阵，并正确实现稀疏矩阵的乘法运算。

## 二、实验要求

在`SparseMatrix.h`中，我们已经定义了稀疏矩阵类`class SparseMatrix`和以下方法：

| 方法 | 描述 |
| --- | --- |
| SparseMatrix(const std::string *input_file*) | 构造函数，读取矩阵文件并构建`SparseMatrix` |
| void to_file(const std::string *output_file*) | 将`SparseMatrix`存储为矩阵文件 |
| SparseMatrix operator*(const SparseMatrix &right*) | 稀疏矩阵乘法运算 |

你需要在`SparseMatrix.c`中实现这些方法。如有必要，你可以在`SparseMatrix.h`中为`class SparseMatrix`添加更多成员变量/成员函数。但是**不要**修改已有成员函数的接口。

矩阵文件是本实验中矩阵的存储格式。对于一个$M$行$N$列的矩阵$A$，其矩阵文件是一个$M+1$行的文本文件，其中：

1. 文件第1行：2个数字，用空格隔开，分别为矩阵的行数$M$和列数$N$。
2. 文件其余行：每行3个数字，用空格隔开，记为$(x,y,v)$，表示$A[x][y]$的值为$v$。
    - 输入文件未指定的矩阵元素值为0。
    - 这些非零元素一定按照从左到右，从上到下的顺序给出。即对于第i行$(x_i,y_i,v_i)$和第j行$(x_j,y_j,v_j)(i<j)$，必满足：

$$((x_i==x_j)\wedge(y_i<y_j))\vee(x_i<x_j)$$

    - 矩阵下标从0开始计算。

例如矩阵

$$\begin{bmatrix}1&0&3\\0&5&0\end{bmatrix}$$

被存储为：

```
2 3
0 0 1
0 2 3
1 1 5
```

## 三、输入输出

本实验的输入输出通过文件给出。

对于测试用例`test1`，参与乘法运算的左右两个矩阵分别为`input/test1.left`和`input/test1.right`。

测试代码会调用`to_file()`将运算结果矩阵输出为`output/test1.yourans`并与正确答案`output/test1.ans`比较，你不需要向stdout额外输出任何字符。

## 四、数据范围

矩阵行数$M$和列数$N$满足：

$$1 \leq M, N \leq 100000$$

每个矩阵（包括矩阵乘法的运算结果）中的非零元素个数$T$满足：

$$1 \leq T \leq 100000$$

每个矩阵（包括矩阵乘法的运算结果）中元素$e$满足：

$$0 \leq e \leq 10^9, e \in N$$

## 五、实现提示

1. 实验输入文件用三元组$(x, y, v)$描述稀疏矩阵中每个非零元素，你也可以构建这样的数据结构来描述稀疏矩阵。
2. 观察任一测试文件（如`test3.right`）的第一列（即矩阵中所有非零元素的行号），你会发现这是一个非严格的单调递增数组，其中有许多重复元素。我们能否利用这一性质优化数据结构，节约存储空间？
3. 矩阵乘法的一种朴素实现方式是：遍历所有$(i, j)$，取左矩阵第$i$行，右矩阵第$j$列做向量内积，得到结果矩阵第$i$行第$j$列的值。对你的数据结构而言，这样的运算顺序足够高效吗？可以使用一种更高效的运算顺序计算稀疏矩阵的乘法吗？

## 六、探究实验

```
// 矩阵乘法的朴素实现
// long A[N][M], B[M][Q], C[N][Q];
// C = A * B
for (int i = 0; i < N; ++i)
    for (int j = 0; j < Q; ++j)
        C[i][j] = 0;
for (int i = 0; i < N; ++i)
    for (int j = 0; j < Q; ++j)
        for (int k = 0; k < M; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

根据输入数据的特点选择合适的算法是极为重要的。上面给出了一个朴素的矩阵乘法实现。相对于我们此前实现的稀疏矩阵乘法，朴素的矩阵乘法的实现更简单。那么在实际使用中，应该选择使用哪种矩阵乘法的实现呢?
请给出你的猜想，并通过设计实验来证明你的猜想。

你需要至少考虑的影响因素包括：

- 矩阵的维度
- 矩阵的稀疏程度（即非零元素占总元素的比例）

所考虑的评价指标至少包括：

- 矩阵运算性能
- 矩阵存储所占用内存空间的影响

请完成一份书面报告，要求：

- **简述**一下你的稀疏矩阵乘法算法中的数据结构设计和实现思路。
- **简单**分析对比一下两种稀疏矩阵乘法算法的时间/空间复杂度。
- 给出对两种矩阵乘法的使用建议，并通过图表的方式展现实验数据，反映以上参数变化对两种矩阵乘法算法的影响。

我们不鼓励在实验报告上无意义内卷，因此实验报告**不应**超过1500字，且字数多少不影响我们对实验报告的评判。

## 七、实验评分

我们提供了Makefile文件，在主目录下执行`make`指令生成可执行程序`sparsematrix`。该程序接受0或1个参数。例如：

```
./sparsematrix test1
```

使用`test1`测试用例进行测试。

```
./sparsematrix
```

使用提供的全部3个测试用例进行测试。

如果你的输出与答案不符合，测试程序会打印第一处不符合的数据。否则，测试程序会输出通过测试的信息。

我们提供了3组测试数据，它们是评分时使用的测试数据的真子集，所以**请勿针对测试数据编程**。

**提交实验请将`SparseMatrix.h`和`SparseMatrix.cpp`打包上传Canvas，命名使用“学号+姓名+lab0”，如“520123456789+张三+lab0.zip”。**

**探究实验报告同样上传到Canvas平台（会有单独作业用于提交），要求PDF格式，命名无要求。不需要提交探究实验代码。**

Lab0截止日期为北京时间**2023年3月8日21:59**（最终截止时间以 canvas 系统为准）。

## 八、注意事项

1. 切勿抄袭！若发现明显雷同现象，该实验评分将以0分计。
2. 注意实验截止日期，迟交或不交会视情况扣分。
3. 请注意按照实验要求完成，避免因为自动脚本评分造成的判分失误。
4. 有问题可通过邮件或微信群联系助教赵逸凡。