作业五

T1

> 设有三对角矩阵 $A_{n\times n}$，将其按行优先顺序压缩存储于一维数组 $b[3*n-2]$ 中，使得 $a_{ij}=b[k]$，请用 k 表示 i，j 的下标变换公式。

对于「三对角矩阵」的第 i 行第 j 列的元素 $A[i][j]$，我们尝试找到「一维压缩矩阵」的对应元素 $b[k]$：

- 当 $i\geq 1$ 时。前面一共有 i 行元素，对应有 $3i-1$ 个元素。第 i 行共有 $j-(i-1)+1=j-i+2$ 个元素，因此 $A[i][j]$ 对应三对角矩阵中的第 $(3i-1)+(j-i+2)=2i+j+1$ 个元素，同样也是一维压缩矩阵中的第 $2i+j+1$ 个元素。由于下标是从 0 开始的，因此此时 $k=2i+j$。
- 当 $i=0$ 时，$k=2i+j$ 也成立。

综上所述：$k=2i+j$

T2

> 若在矩阵中存在一个元素 $a_{ij}(0\leq i\leq m-1,0\leq j\leq n-1)$ 满足：a_{ij} 是第 i 行元素中最小值，且又是第 j 列元素中最大值，则称此元素值为该矩阵的一个马鞍点。假设以二维数组存储矩阵 $A_{m\times n}$，试编写求出矩阵中所有马鞍点的算法。

遍历两遍二维数组，第一遍得到每行的最小值 $r[i]$ 和每列的最大值 $c[j]$，第二遍将每个元素 $mat[i][j]$ 和行最小值、列最大值 $O(1)$ 比较即可。

时间复杂度：$O(mn)$

```cpp
vector<pair<int, int>> getSaddlesOfMatrix(
    const vector<vector<int>>& mat = {{0,  2, 0},
                                      {-1, 2, -2}}) {
    if (mat.empty() || mat[0].empty()) {
        return {};
    }

    vector<pair<int, int>> res;
    int n = mat.size(), m = mat[0].size();
    vector<int> r(m, INT_MAX), c(n, INT_MIN);

    for (int i = 0; i < n; i++) {
```

```
        for (int j = 0; j < m; j++) {
            r[i] = min(r[i], mat[i][j]);
            c[j] = max(c[j], mat[i][j]);
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (mat[i][j] <= r[i] && mat[i][j] >= c[j]) {
                res.push_back({i, j});
            }
        }
    }

    return res;
}
```

T3

> 编写算法计算一个稀疏矩阵的对角线元素之和，要求稀疏矩阵用三元组顺序表表示。

遍历三元组，遇到其中行列号相等的元素，就将存储的值加上去即可

```
template<class T>
T SparseMatrix<T>::sumOfDiag() {
    T res = 0;
    for (auto t: data) {
        if (t.r == t.c) {
            res += t.value;
        }
    }
    return res;
}
```

实验五

实验题 实现一个能进行稀疏矩阵基本运算的运算器程序。

基本要求：

（1）以三元组顺序表结构表示稀疏矩阵，实现矩阵转置和两个矩阵相加、相减的运算。

（2）稀疏矩阵的输入形式采用三元组表示，程序可以对三元组的输入顺序加以限制，如按行优先。

（3）矩阵转置和矩阵相加或相减所得结果矩阵另外生成。

（4）运算结果矩阵以通常的阵列形式输出。

由于最终结果均为阵列形式，因此稀疏矩阵的输入顺序并不影响计算效率。

转置：

```cpp
template<class T>
vector<vector<T>> SparseMatrix<T>::transpose() {
    swap(row, col);
    vector<vector<T>> res(row + 1, vector<T>(col + 1, 0));
    for (auto& tri: data) {
        res[tri.c][tri.r] = tri.value;
        swap(tri.r, tri.c);
    }
    return res;
}
```

相加：

```cpp
template<class T>
vector<vector<T>> SparseMatrix<T>::plus(SparseMatrix<T>& sm, bool is_plus) {
    if (row != sm.row || col != sm.col) {
        cerr << "different shape! can't calculate" << "\n";
        exit(1);
    }

    vector<vector<T>> res(row + 1, vector<T>(col + 1, 0));
    for (auto& tri: data) {
        int r = tri.r, c = tri.c;
        T value = tri.value;
        res[r][c] += value;
    }
    for (auto& tri: sm.data) {
        int r = tri.r, c = tri.c;
        T value = tri.value;
```

```
            res[r][c] += is_plus ? value : -value;
        }

        return res;
    }
```

相减：

```
template<class T>
vector<vector<T>> SparseMatrix<T>::minus(SparseMatrix<T>& sm) {
    return plus(sm, false);
}
```