**矩阵的快速转置算法实现（超级困难）**

**用空间换取时间，是一种常用的手段**

void FastTransposeMatrix(SMatrix\* M, SMatrix\* T) {

T->mu = M->nu;

T->nu = M->mu;

T->tu = M->tu;

int\* num = (int\*)malloc(sizeof(int) \* M->nu); //num[col]表示第col列非零元素的个数

assert(num != NULL);

int\* cpot = (int\*)malloc(sizeof(int) \* M->nu); //cpot[col]表示第col列第一个非零元素在转置后的矩阵b中的位置

assert(cpot != NULL);

if (T->tu != 0) {

for (int col = 0; col < M->nu; col++) {

num[col] = 0; //先将num数组里每一个值都赋空

}

for (int t = 0; t < M->tu; t++) {

num[M->data[t].j]++; //遍历矩阵里的非零元素，对应的列非零元的个数加一

}

cpot[0] = 0; //第0列的第一个非零元素在转置后的矩阵b中的位置毫无疑问是0

for (int col = 1; col < M->nu; col++) {

cpot[col] = cpot[col - 1] + num[col - 1]; //通过一系列推倒得出的公式

}

int q = 0;

for (int p = 0; p < M->tu; p++) { //进行放置

int col = M->data[p].j; //进行循环，找到每个M中的三元组所对应的列的值

q = cpot[col]; //由相应的列的值可以得到这一列第一个非零元素的在转置矩阵b中的位置 赋值给q

T->data[q].i = M->data[p].j; //由于转置前的矩阵里三元组的排列方式是行越小越在前，那么第一个找到的其实就是所在列的第一个非零元素

T->data[q].j = M->data[p].i;

T->data[q].e= M->data[p].e;

cpot[col]++; //将三元组插入到新的矩阵后，其所在列对应的第一个非零元素在转置后矩阵的位置要+1；

}

}

free(num);

free(cpot);

}