//计算 Q = M + N(要求稀疏矩阵 M 和 N 有相同的行数和列数)

//算法思路：

/\*稀疏矩阵的相加：用循环来判断存储M和N稀疏矩阵的两个三元组表中的行列下标是否相等和其大小关系。若行列号相等，则将两个三元组表的val值相加存入新的数组Q里，行列号不变存入Q；若M的列小于N的列，则将M的row，col，val直接存入Q中；若N的列小于M的列，则将N的row，col，val直接存入Q中；若M的行小于N的行，则将M的row，col，val直接存入Q中；若M的行大于N的行，则将N的row，col，val直接存入Q中。\*/

//参考代码

int tableAdd(table M, table N, table \*Q)

{

//矩阵信息

Q->cnum = M.cnum;

Q->rnum = M.rnum;

Q->len = 0;

while (i < M.len && j < N.len)

{

// 如果 i j 指向元素是同一行的元素

if (M.data[i].row == N.data[j].row)

{

// 如果 i 和 j 指向的元素指向的是同一个元素

if (M.data[i].col == N.data[j].col)

{

Q->data[k].row = M.data[i].row;

Q->data[k].col = M.data[i].col;

Q->data[k].val = M.data[i].val + N.data[j].val;

Q->len++;

i++;

j++;

k++;

}

// 如果 i 指向元素的列下标大于 j 指向元素的列下标

// 把下标小（j 指向的元素）的放入到 Q 矩阵中

else if (M.data[i].col > N.data[j].col)

{

Q->data[k].row = N.data[j].row;

Q->data[k].col = N.data[j].col;

Q->data[k].val = N.data[j].val;

Q->len++;

j++;

k++;

}

// 如果 i 指向元素的列下标小于 j 指向元素的列下标

// 把下标小（i 指向的元素）的放入到 Q 矩阵中

else if (M.data[i].col < N.data[j].col)

{

Q->data[k].row = M.data[i].row;

Q->data[k].col = M.data[i].col;

Q->data[k].val = M.data[i].val;

Q->len++;

i++;

k++;

}

}

// 如果 i 指向的元素行下标大于 j 指向元素的行下标

// 把行下标小（j 指向的元素）的放入到 Q 矩阵中

else if (M.data[i].row > N.data[j].row)

{

Q->data[k].row = N.data[j].row;

Q->data[k].col = N.data[j].col;

Q->data[k].val = N.data[j].val;

Q->len++;

k++;

j++;

}

// 如果 i 指向元素行下标小于 j 指向元素的行下标

// 把行下标小（i 指向的元素）的放入到 Q 矩阵中

else if (M.data[i].row < N.data[j].row)

{

Q->data[k].row = M.data[i].row;

Q->data[k].col = M.data[i].col;

Q->data[k].val = M.data[i].val;

Q->len++;

i++;

k++;

}

}

// 如果还有剩余元素，按顺序把元素添加到结果矩阵中

while (i < M.len)

{

Q->data[k].row = M.data[i].row;

Q->data[k].col = M.data[i].col;

Q->data[k].val = M.data[i].val;

Q->len++;

i++;

k++;

}

while (j < N.len)

{

Q->data[k].row = N.data[j].row;

Q->data[k].col = N.data[j].col;

Q->data[k].val = N.data[j].val;

Q->len++;

j++;

k++;

}

}