数据结构实验报告之三元组顺序表存储的稀疏矩阵练习

原创

nufe_wwt 于 2020-04-18 17:00:15 发布 2728 收藏 36

分类专栏： 数据结构大讨论 #稀疏矩阵快速转置 文章标签： 数据结构 c语言 算法 编程语言

本文链接：https://blog.csdn.net/W_REP/article/details/105600593

数据结构实验报告之三元组顺序表存储的稀疏矩阵练习

一、实验内容

1.实验目的

为了更加深入了解三元组以及稀疏矩阵，学会快速转置稀疏矩阵的算法设计并以此为基础思考其他相关问题。本次实验主要以解决某特定6阶稀疏矩阵三对角线元素之和以及转置该6阶矩阵为目的而进行算法设计以及代码实现。

2.实验内容

(1)构建一个三元组顺序表存储的稀疏6阶方阵
(2)求解该方阵中三条对角线上的元素之和
(3)实现该方阵的快速转置
(4)实现代码并输出相关数据

一．要实现（1）只需要给与指定的行、列以及元素值即可。本次实验原稀疏矩阵设置为S，转置矩阵设置为T，下面展示的是S以及S对应的三元组：

S					
0	0	0	0	0	0
1	0	0	0	0	0
0	0	0	2	0	0
0	0	0	0	0	0
0	0	3	0	0	0
0	0	0	0	0	4

S.mu=6	S.nu=6	S.tu=4	
S.data	i	j	e
0			
1	2	1	1
2	3	4	2
3	5	3	3
4	6	6	4

即转置后的T方阵以及对应的三元组应当如下所示：

T					
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	3	0
0	0	2	0	0	0
0	0	0	0	0	0
0	0	0	0	0	4

T.mu=6	T.nu=6	T.tu=4	
T.data	i	j	e
0			

T.mu=6	T.nu=6	T.tu=4	
1	1	2	1
2	3	5	3
3	4	3	2
4	6	6	4

二.为实现（2）中的三对角线总和，本此实验设计了一个**sumDiagnoal3**函数以求和。

三.快速转置算法为课上所学**FastTransposeTSMatrix**函数，并给与了更加详细的描述。

四.实验输出见下文（三、**实验结果**）。

二、实现过程

本次实验环境：macOS Mojave, Xcode, gcc, C99

根据实验要求，本次实验依次设计了两个函数以完成三对角线求和方阵转置，并给与较为清晰的输出。下面是实验过程：

1.编译预处理以及使用类别名

编译预处理以及使用类别名，并没有使用到过大的矩阵但还是将MAXSIZE设置的足够大，创建了三元组(Triple)结构体以及稀疏矩阵(TSMatrix)结构体如下所示:

```
#include <stdio.h>
#define MAXSIZE 12500
#define OK 1
#define ERROR 0
typedef int Status, ElemType;
typedef struct {
   int i, j;//该非零元素的行下标和列下标
   ElemType e;
} Triple;
typedef struct {
   Triple data[MAXSIZE+1];//非零元三元组表，data[0]未用
   int mu, nu, tu;//矩阵的行数，列数和非零元个数
} TSMatrix;//稀疏矩阵
```

2.sumDiagnoal3()函数

计算三条对角线元素之和的函数(sumDiagnoal3)，如下所示：
初实现求和算法时并没有仔细考虑，但是突然产生一个念想，要是这次实验要我们完成的是五条对角线呢？于是带着这样的想法，我决定吧算法单独作为一个小函数拿出来。

```
int sumDiagonal3(TSMatrix S) {//求三条对角线上元素之和
   int i, sum=0;
   for(i=1; i<=S.tu; i++) {
      if((S.data[i].i-S.data[i].j)<=1 && (S.data[i].j-S.data[i].i)<=1)
         sum += S.data[i].e;
   }
   return sum;
}
```

3.FastTransposeTSMatrix()函数

快速转置算法(FastTransposeTSMatrix)，如下所示：
由于该函数实现机制过于巧妙，故在编写代码时我添加了很多的注释以帮助自己读懂。不妨下面我再解释几行较为难懂的代码，一遍日后复习时需要使用：

```
Status FastTransposeTSMatrix(TSMatrix M, TSMatrix *T) {
   //采用三元组顺序表存储表示，求稀疏矩阵M的转置矩阵T。
   T->mu=M.nu; T->nu=M.mu;T->tu=M.tu;
   int col,t;//col指示列,t指示元素个数
   int num[M.mu], cpot[M.mu];//分别表示M矩阵中第col列非零元素个数 和
   if(T->tu) {                //M中第col列的第一个非零元在T->data中的恰当位置
      for(col=1; col<=M.nu; ++col) num[col]=0;//均赋初值为0
      for(t=1; t<=M.tu; ++t) ++num[M.data[t].j];//求M每一列中含非零元素个数
      cpot[1]=1;
      //求第col列中第一个非零元在b.data的序号
      for(col=2; col<=M.nu; ++col) cpot[col]=cpot[col-1]+num[col-1];
      for(int p=1,q; p<=M.tu; ++p) {//p是M.data的下标,q是T->data的下标,两两对应
         col=M.data[p].j;
         q=cpot[col];
         T->data[q].i = M.data[p].j; T->data[q].j = M.data[p].i;//换位置
         T->data[q].e = M.data[p].e; ++cpot[col];//换元素
      }//for
   }//if
   return OK;
}//FastTransposeTSMatrix
```

解释：line1：由于是基于C设计的算法，故使用指针来传入T（因为需要传出）；line5：num与cpot数组为教材上讲到的两个辅助向量（page100），num用来指示原方阵每一列非零元素的个数，cpot用来指示原方阵每一列的第一个非零元素在转置矩阵的data数组中恰当的位置（即指示下标）；line8：求出原矩阵每一列非零元素个数并给num数组完成赋值（这里的赋值算法很巧妙，值得细究）；line9：总所周知cpot[1]=1；line11：利用公式cpot[i]=cpot[i-1]+numi-1给每一个cpot赋值；line12-17：p指示原方阵的data数组的下标，q指示转换后的方阵的data数组的下标，data三元组即为存放非零元素的数组。

4.主函数

4.主函数，如下分成四块：

```
(main1)
int main() {
   TSMatrix S,T;
   S.mu=6; S.nu=6; S.tu=4;
   S.data[1].i=2; S.data[1].j=1; S.data[1].e=1;
   S.data[2].i=3; S.data[2].j=4; S.data[2].e=2;
   S.data[3].i=5; S.data[3].j=3; S.data[3].e=3;
   S.data[4].i=6; S.data[4].j=6; S.data[4].e=4;
```

```
(main2)
   printf(" 稀疏矩阵S共有%d个元素,分别为:\n", S.tu);
   int k;
   for(k=1; k<=S.tu; k++)
      printf(" 第%d行、第%d列:%d\n", S.data[k].i, S.data[k].j, S.data[k].e);
```

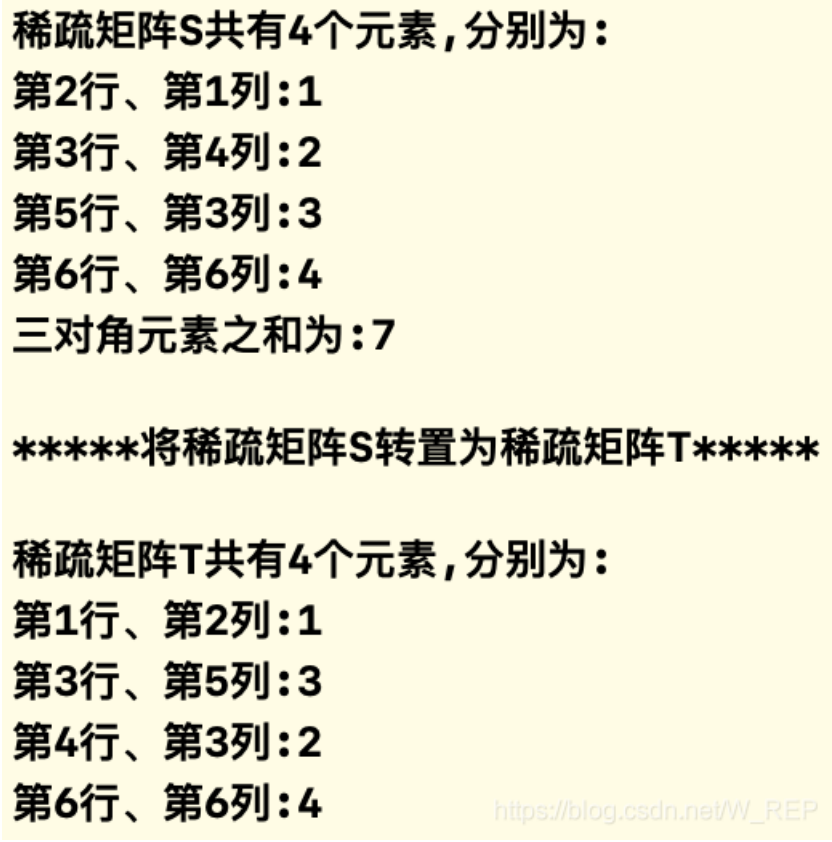
```
(main3)
   printf(" 三对角元素之和为:%d\n",sumDiagonal3(S));
```

```
(main4)
   FastTransposeTSMatrix(S, &T);//将S转置为T
   printf("\n *****将稀疏矩阵S转置为稀疏矩阵T*****\n\n");
   printf(" 稀疏矩阵T共有%d个元素,分别为:\n", S.tu);
   for(k=1; k<=T.tu; k++)
      printf(" 第%d行、第%d列:%d\n", T.data[k].i, T.data[k].j, T.data[k].e);
   printf("\n");
}
```

解释：main1部分定义了原方阵S和转置后的方阵T，并将稀疏矩阵S的非0元素的行列以及值赋值；main2部分进行了一次输出，给出了S的非0元个数并把其所在的行列以及值展示出来；main3部分输出S三对角线元素之和；main4部分转置S为T并输出方阵T的相关信息。

三、实验结果

输出结果如图3-1所示



四、编程时遇到的问题

鲁棒性问题：在进行实验的第三大步：实验结果输出时我想到此次试验没有考虑到鲁棒性。而要实现鲁棒性就要考虑在初始化S三元组时的随机性了，所以需要使用到rand随机函数，但是由于本次实验耗时太久（主要核心算法用时太久了），所以实现鲁棒性的操作等空闲时间来做吧！下次一定！

五、总结

三元组在处理非零元素极少的稀疏矩阵时有奇效!确实是这样，但是我偷偷试了试时间复杂度为n*m的算法，直观感受不到运行速度的差距，不过我这只是个6X6的方阵而已啦！等到以后处理的数据量庞大起来，就不可能考虑那么费时的代码了。

六、完整代码

附上完整代码

```c
#include <stdio.h>
#define MAXSIZE 12500
#define OK 1
#define ERROR 0
typedef int Status, ElemType;
typedef struct {
   int i, j;//该非零元素的行下标和列下标
   ElemType e;
} Triple;
typedef struct {
   Triple data[MAXSIZE+1];//非零元三元组表，data[0]未用
   int mu, nu, tu;//矩阵的行数，列数和非零元个数
} TSMatrix;//稀疏矩阵

Status FastTransposeTSMatrix(TSMatrix M, TSMatrix *T) {
   //采用三元组顺序表存储表示，求稀疏矩阵M的转置矩阵T。
   T->mu=M.nu; T->nu=M.mu;T->tu=M.tu;
   int col,t;//col指示列,t指示元素个数
   int num[M.mu], cpot[M.mu];//分别表示M矩阵中第col列非零元素个数 和
   if(T->tu) {                //M中第col列的第一个非零元在T->data中的恰当位置
      for(col=1; col<=M.nu; ++col) num[col]=0;//均赋初值为0
      for(t=1; t<=M.tu; ++t) ++num[M.data[t].j];//求M每一列中含非零元素个数
      cpot[1]=1;
      //求第col列中第一个非零元在b.data的序号
      for(col=2; col<=M.nu; ++col) cpot[col]=cpot[col-1]+num[col-1];
      for(int p=1,q; p<=M.tu; ++p) {//p是M.data的下标,q是T->data的下标,两两对应
         col=M.data[p].j;
         q=cpot[col];
         T->data[q].i = M.data[p].j; T->data[q].j = M.data[p].i;//换位置
         T->data[q].e = M.data[p].e; ++cpot[col];//换元素
      }//for
   }//if
   return OK;
}//FastTransposeTSMatrix

int sumDiagonal3(TSMatrix S) {//求三条对角线上元素之和
   int i, sum=0;
   for(i=1; i<=S.tu; i++) {
      if((S.data[i].i-S.data[i].j)<=1 && (S.data[i].j-S.data[i].i)<=1)
         sum += S.data[i].e;
   }
   return sum;
}

int main() {
   TSMatrix S,T;
   S.mu=6; S.nu=6; S.tu=4;
   S.data[1].i=2; S.data[1].j=1; S.data[1].e=1;
   S.data[2].i=3; S.data[2].j=4; S.data[2].e=2;
   S.data[3].i=5; S.data[3].j=3; S.data[3].e=3;
   S.data[4].i=6; S.data[4].j=6; S.data[4].e=4;
   printf(" 稀疏矩阵S共有%d个元素,分别为:\n", S.tu);
   int k;
   for(k=1; k<=S.tu; k++)
      printf(" 第%d行、第%d列:%d\n", S.data[k].i, S.data[k].j, S.data[k].e);

   printf(" 三对角元素之和为:%d\n",sumDiagonal3(S));

   FastTransposeTSMatrix(S, &T);//将S转置为T
   printf("\n *****将稀疏矩阵S转置为稀疏矩阵T*****\n\n");
```

```
    printf("\n\n*****将稀疏矩阵S转置为稀疏矩阵T*****\n\n");
    printf(" 稀疏矩阵T共有%d个元素,分别为:\n", S.tu);
    for(k=1; k<=T.tu; k++)
        printf(" 第%d行、第%d列:%d\n", T.data[k].i, T.data[k].j, T.data[k].e);
    printf("\n");
}
```