实验五矩阵压缩存储及应用

实验周次：第12周 学时：2学时 地点：学院机房

学号：319070712 姓名：武新纪 班级：人工智能191 序号：21

提示：请务必填写 以上个人信息。

**【实验目的】**

1.掌握对称矩阵的压缩存储方法及相关算法设计。

2.掌握稀疏矩阵三元组存储结构即基本算法设计。

**【实验内容】**

实验题1.已知A和B为两个n\*n阶的对称方阵，在输入时，对称矩阵只输入主对角线元素和下三角元素。假设对称方阵A的主对角元素和下三角元素存入一维数组C中；对称方阵B的主对角元素和下三角元素存入一维数组D中。设计一个程序exp5-1.cpp实现以下功能：

假设对称方阵A为：

假设对称方阵B为：

# define N 4

1. 输入并压缩存储对称方阵A和B主对角线元素和下三角元素到一维数组中。可以从键盘录入也可以直接初始化赋值。如int C={9,1,2,3,7,6,5,8,4,0};int D={11,14,12,31,71,13,56,83,42,14}
2. 基于压缩存储，给出求解对称方阵A和B的求和运算实现算法代码。（和可以存储在二维数组int C[4][4]中）
3. 基于压缩存储，给出求解对称方阵A和B的乘积运算实现算法代码。（和可以存储在二维数组int C[4][4]中）

**要求给出算法实现函数代码，并通过main()函数调用并调试，给出调试结果截图。**

参考代码：

//c为矩阵压缩存储一维数组,i,j为A矩阵元素ai,j的行，列下标。返回A矩阵元素ai,j的值

int value(int c[],int i,int j) a中A[i][j]之值

{

if (i>=j)

return c[(i\*(i-1))/2+j];

else

return c[(j\*(j-1))/2+i];

}

// a和b为对称矩阵A和B的压缩存储一维数组。C为存放对称矩阵A和B求和结果的矩阵二维数组

void madd(int a[],int b[],int c[][N]) //求压缩存储a和b的和

{

int i,j;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=value(a,i,j)+value(b,i,j);

}

// a和b为对称矩阵A和B的压缩存储一维数组。C为存放对称矩阵A和B乘积结果的矩阵二维数组，A,B,C均为N\*N的方阵。

void mult(int a[],int b[],int c[][N]) //求压缩存储a和b的乘积

{

int i,j,k,s;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

{

s=0;

for (k=0;k<N;k++)

s=s+value(a,i,k)\*value(b,k,j);

c[i][j]=s;

}

}

-------------------------如提交电子版实验报告，请在下方撰写电子版实验报告-------------------------

**程序:**

**//exp5-1.cpp**

#include <stdio.h>

#define N 4

#define M 10

void fun(int a[][N], int b[M])

{

    int i, j;

    for (i = 0; i < N; i++)

        for (j = 0; j < N; j++)

        {

            if (i >= j)

                b[(i \* (i + 1)) / 2 + j] = a[i][j];

            else

                b[(j \* (j + 1)) / 2 + i] = a[i][j];

        }

}

int value(int a[], int i, int j) //压缩为一维数组

{

    if (i >= j)

        return a[(i \* (i + 1)) / 2 + j];

    else

        return a[(j \* (j + 1)) / 2 + i];

}

void madd(int a[], int b[], int c[][N]) //压缩存储a和b的和

{

    int i, j;

    for (i = 0; i < N; i++)

        for (j = 0; j < N; j++)

            c[i][j] = value(a, i, j) + value(b, i, j);

}

void mult(int a[], int b[], int c[][N]) //压缩存储a和b的乘积

{

    int i, j, k, s;

    for (i = 0; i < N; i++)

        for (j = 0; j < N; j++)

        {

            s = 0;

            for (k = 0; k < N; k++)

                s = s + value(a, i, k) \* value(b, k, j);

            c[i][j] = s;

        }

}

void disp1(int a[]) //输出原始矩阵

{

    int i, j;

    for (i = 0; i < N; i++)

    {

        for (j = 0; j < N; j++)

            printf("%4d", value(a, i, j));

        printf("\n");

    }

}

void disp2(int c[][N]) //输出求和，求积矩阵

{

    int i, j;

    for (i = 0; i < N; i++)

    {

        for (j = 0; j < N; j++)

            printf("%4d", c[i][j]);

        printf("\n");

    }

}

int main()

{

    int a[M], b[M];

    int A[N][N] = {{9, 1, 3, 5}, {1, 2, 7, 8}, {3, 7, 6, 4}, {5, 8, 4, 0}};

int B[N][N] = {{11, 14, 31, 56}, {14, 12, 71, 83},

 {31, 71, 13, 42}, {56, 83, 42, 14}};

    int c1[N][N], c2[N][N];

    fun(A, a);

    fun(B, b);

    madd(a, b, c1);

    mult(a, b, c2);

    printf("a矩阵为:\n");

    disp1(a);

    printf("b矩阵为:\n");

    disp1(b);

    printf("a+b:\n");

    disp2(c1);

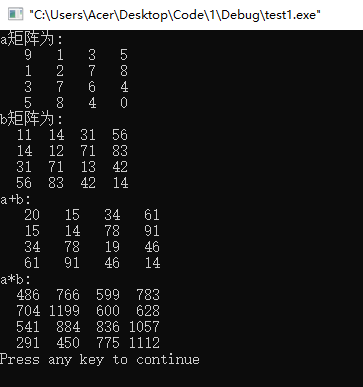
    printf("a\*b:\n");

    disp2(c2);

    return 0;

}

**调试结果:**



实验2.已知A和B为两个n\*n阶的稀疏矩阵，，给出三元组顺序表的压缩存储，在输入或者赋值时，只输入或者赋值非0元素。假设稀疏矩阵A非0元素存入三元组顺序表a中；稀疏矩阵B非0元素存入三元组顺序表b中；设计一个程序exp5-2.cpp实现以下功能：

假设稀疏矩阵A为：

假设稀疏矩阵B为：

# define N 4

1. 给出稀疏矩阵A和B的三元组顺序表a和b的存储实现算法代码。
2. 输出a转置矩阵的三元组。
3. 求稀疏矩阵A和B的和。（和可以存储在二维数组int Array[4][4]中）
4. 求稀疏矩阵A和B的乘积。（和可以存储在二维数组int Array[4][4]中）

矩阵的求和和乘积可以自行查找资料了解，实现代码参考实验题1给出的提示代码。

要求：给出实现算法代码并用main()函数调用，调试并给出输出结果截图。

-------------------------如提交电子版实验报告，请在下方撰写电子版实验报告-------------------------

**程序：**

//文件名:exp5-2.cpp

#include <stdio.h>

#define N 4

typedef int ElemType;

#define MaxSize 100 //矩阵中非零元素最多个数

typedef struct

{

    int r;      //行号

    int c;      //列号

    ElemType d; //元素值

} TupNode;      //三元组定义

typedef struct

{

    int rows; //行数值

    int cols; //列数值

    int nums; //非零元素个数

    TupNode data[MaxSize];

} TSMatrix;

void CreatMat(TSMatrix &t, ElemType A[N][N])

{

    int i, j;

    t.rows = N;

    t.cols = N;

    t.nums = 0;

    for (i = 0; i < N; i++)

    {

        for (j = 0; j < N; j++)

            if (A[i][j] != 0)

            {

                t.data[t.nums].r = i;

                t.data[t.nums].c = j;

                t.data[t.nums].d = A[i][j];

                t.nums++;

            }

    }

}

void DispMat(TSMatrix t)

{

    int i;

    if (t.nums <= 0)

        return;

    printf("\t%d\t%d\t%d\n", t.rows, t.cols, t.nums);

    printf("\t------------------\n");

    for (i = 0; i < t.nums; i++)

        printf("\t%d\t%d\t%d\n", t.data[i].r, t.data[i].c, t.data[i].d);

}

void TranMat(TSMatrix t, TSMatrix &tb){

    int p, q = 0, v;

    tb.rows = t.cols;

    tb.cols = t.rows;

    tb.nums = t.nums;

    if (t.nums != 0)

    {

        for (v = 0; v < t.cols; v++)

            for (p = 0; p < t.nums; p++)

                if (t.data[p].c == v)

                {

                    tb.data[q].r = t.data[p].c;

                    tb.data[q].c = t.data[p].r;

                    tb.data[q].d = t.data[p].d;

                    q++;

                }

    }

}

bool MatAdd(TSMatrix a, TSMatrix b, TSMatrix &c)

{

    int i = 0, j = 0, k = 0;

    ElemType v;

    if (a.rows != b.rows || a.cols != b.cols)

        return false;

    c.rows = a.rows;

    c.cols = a.cols;

    while (i < a.nums && j < b.nums)

    {

        if (a.data[i].r == b.data[j].r)

        {

            if (a.data[i].c < b.data[j].c)

            {

                c.data[k].r = a.data[i].r;

                c.data[k].c = a.data[i].c;

                c.data[k].d = a.data[i].d;

                k++;

                i++;

            }

            else if (a.data[i].c > b.data[j].c)

            {

                c.data[k].r = b.data[j].r;

                c.data[k].c = b.data[j].c;

                c.data[k].d = b.data[j].d;

                k++;

                j++;

            }

            else

            {

                v = a.data[i].d + b.data[j].d;

                if (v != 0)

                {

                    c.data[k].r = a.data[i].r;

                    c.data[k].c = a.data[i].c;

                    c.data[k].d = v;

                    k++;

                }

                i++;

                j++;

            }

        }

        else if (a.data[i].r < b.data[j].r)

        {

            c.data[k].r = a.data[i].r;

            c.data[k].c = a.data[i].c;

            c.data[k].d = a.data[i].d;

            k++;

            i++;

        }

        else

        {

            c.data[k].r = b.data[j].r;

            c.data[k].c = b.data[j].c;

            c.data[k].d = b.data[j].d;

            k++;

            j++;

        }

        c.nums = k;

    }

    return true;

}

int getvalue(TSMatrix t, int i, int j)

{

    int k = 0;

    while (k < t.nums && (t.data[k].r != i || t.data[k].c != j))

        k++;

    if (k < t.nums)

        return (t.data[k].d);

    else

        return (0);

}

bool MatMul(TSMatrix a, TSMatrix b, TSMatrix &c)

{

    int i, j, k, p = 0;

    ElemType s;

    if (a.cols != b.rows)

        return false;

    for (i = 0; i < a.rows; i++)

        for (j = 0; j < b.cols; j++)

        {

            s = 0;

            for (k = 0; k < a.cols; k++)

                s = s + getvalue(a, i, k) \* getvalue(b, k, j);

            if (s != 0)

            {

                c.data[p].r = i;

                c.data[p].c = j;

                c.data[p].d = s;

                p++;

            }

        }

    c.rows = a.rows;

    c.cols = b.cols;

    c.nums = p;

    return true;

}

int main()

{

    ElemType a1[N][N] = {{1, 0, 4, 0}, {0, 1, 0, 0}, {0, 0, 3, 0}, {0, 0, 1, 1}};

    ElemType b1[N][N] = {{5, 0, 0, 0}, {0, 4, 0, 0}, {0, 0, 2, 0}, {0, 0, 0, 8}};

    TSMatrix a, b, c;

    CreatMat(a, a1);

    CreatMat(b, b1);

    printf("A的三元组:\n");

    DispMat(a);

    printf("\nB的三元组:\n");

    DispMat(b);

    printf("\nA的转置三元组:\n");

    TranMat(a, c);

    DispMat(c);

    printf("\nA+B的三元组:\n");

    MatAdd(a, b, c);

    DispMat(c);

    printf("\nA×B的三元组:\n");

    MatMul(a, b, c);

    DispMat(c);

    return 0;

}

**调试结果:**

