**附录：实验报告**

# 实验题目 离散数学实验1

学号 2019011777 姓名 刘康来 班级 计算机19-3 时间 2020、10、8

# 实验题目解答

（对解题的整体思路、过程进行提炼和描述，包括算法描述、程序结构、主要变量说明、设计技巧、调试情况、运行结果、心得体会等）

（1）1 2 0 0

0 0 1 0

1 0 0 1

0 0 1 0

1. 计算长度为4的通路数，可以计算邻接矩阵的4次方，所有元素相加即可。

- 定义三个函数：矩阵的乘法，n次方，计算所有元素和

- 变量：矩阵长度n，要求长度（几次方）k，

矩阵a，b，c（a，b相乘得矩阵c）

二维指针p

错误：

1.记得将所有矩阵赋初值，用memcpy，加头文件string.h

2.开始我没有定义矩阵c，在程序中两矩阵a，b相乘时，直接改变了矩阵b的值，导致结果出错。。。

3.有些函数传递的量太多，容易忘记（或许可用全局变量）

（3）可达矩阵

Method1：总共4个节点，若两节点可达，则距离最大为3（无回路），故可算出A、A2、A3，在三矩阵中元素为0的即为不可达。(所写程序为method2）

Method2：要知两节点是否可达，只用A即可：

如：要查v1到v4是否可达，先看邻接矩阵a14是否不等0，若是，则可达；否，则看v1到v2（或v3）是否可达，进而看v2（或v3）到v4是否可达；此过程可用递归实现。

- 先使可达矩阵中对角线元素全为1，其他为0（ps：对二维数组需赋初值，为啥一维全默认0？）

- 查找可达矩阵元素bij为0，则查找节点i到j是否可达（用函数Findway）

Findway：先看aij是否非0，有则可达，返回1；进而找i到哪一节点x可达，则递归x到j；若x到j不return 1，则可再找x，到底为止，一旦找到i可达x可达x1可达····到j，则不断返回1。

S数组的作用：为避免查i到j时，出现回路，重复查找，还可能造成死循环。故从i到x，x到j时，不可出现x又到i，再查i到j。可对出现节点赋1，未出现节点赋0实现。

> 注意每次查找完后，对S全赋0，再开始新一轮查找。

# 附：源程序

（2）

#include<stdio.h>

#include<string.h>

int (\*p)[4];

void Matrixmultiply(int (\*a)[4],int (\*p)[4], int n,int (\*b)[4])

{

int sum = 0;

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

for(int j = 0;j < n;j++){

for(int m = 0;m < n;m++){

sum += a[i][m] \* b[m][j];

}

\*(\*(p + i) + j) = sum;

sum = 0;

}

}

}

void Nmultiply(int k, int (\*a)[4],int (\*b)[4], int n,int (\*c)[4])

{

p = c;

for(int i = 1;i < k;i++){

Matrixmultiply(a,p,n,b);

memcpy(b,c,64);

}

}

void CalculateElement(int (\*p)[4],int n)

{

int sum = 0;

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

for(int j = 0;j < n;j++){

sum += p[i][j];

}

}

printf("%d\n",sum);

}

int main(void)

{

int a[4][4]={

1,2,0,0,

0,0,1,0,

1,0,0,1,

0,0,1,0

}, b[4][4], n = 4,k;

int c[4][4];

memcpy(b,a,64);

memcpy(c,a,64);

k = 4;

Nmultiply(k,a,b,n,c);

CalculateElement(p,n);

/\*for(int i = 0;i < n;i++){

for(int j = 0;j < n;j++){

printf("%d ",b[i][j]);

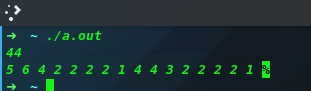
}

}

\*/

return 0;

}



（3）

#include<stdio.h>

int Findway(int (\*a)[4],int n,int i,int j,int \*s)

{

if(a[i][j] != 0){

return 1;

}

else{

s[i] = 1;

for(int x = 0;x < n;x++){

if(a[i][x] != 0 && s[x] == 0 && Findway(a,n,x,j,s) == 1) return 1;

}

}

return 0;

}

int main(void)

{

int a[4][4]={

1,2,0,0,

0,01,0,

1,0,0,1,

0,0,1,0

}, b[4][4], n = 4;

//s records the nodes that has been found.

int s[4];

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

for(int j = 0;j < n;j++){

if(i == j) b[i][j] = 1;

else b[i][j] = 0;

}

}

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

for(int j = 0;j < n;j++){

if(b[i][j] == 0 && Findway(a,n,i,j,s) == 1) b[i][j] = 1;

// finish b[i][j],then s should be back.

for(int c = 0;c < n;c++){

s[c] = 0;

}

}

}

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

for(int j = 0;j < n;j++){

printf("%d ",b[i][j]);

}

printf("\b\n");

}

return 0;

}

