19组离散作业

合成关系

题目描述

输入两个关系R和S，输出它们的合成关系。（10组测试用例）

程序代码

```
#include<stdio.h>
int main()
{
    int i,j,k;
    int a,b,c;
    scanf("%d %d %d",&a,&b,&c);

    int R[50][50] = {0},S[50][50] = {0};
    for(i = 0;i < a;i++)
        for(j = 0;j < b;j++)
            scanf("%d",&R[i][j]);
    for(i = 0;i < b;i++)
        for(j = 0;j < c;j++)
            scanf("%d",&S[i][j]);

    int d[50][50];
    for(i = 0;i < a;i++)
    {
        for(j = 0;j < c;j++)
```

```
        {
            d[i][j] = 0;
            for(k = 0;k<b;k++)
                d[i][j] = d[i][j] + R[i][k] * S[k][j];
        }
    }

    printf("\n");
    for(i = 0;i < a;i++)
    {
        for(j = 0;j < c;j++)
            if(d[i][j] != 0) printf("1 ");
            else printf("0 ");
        printf("\n");
    }

    return 0;
}

```

测试数据

```
# 共十组数据，每组之间用空格分隔
# 输入格式：第一行三个整数 x, y, z，分别表示接下来要输入一个 x * y 的关系矩阵和一个 y * z 的关系矩阵
# 后面的 x 行，每行 y 个整数，且为 0 或者 1 ，表示关系
# 后面的 y 行，每行 z 个整数，且为 0 或者 1 ，表示关系
# 输出：两个关系矩阵的合成关系

6 3 9
0 1 1
0 1 0
0 1 0
0 0 1
0 0 0
0 0 1
1 0 1 0 0 1 0 0 1
0 1 1 1 0 0 0 1 0
0 1 0 0 0 1 1 0 0

9 5 3
1 0 1 1 0
1 0 0 1 1
1 1 1 1 0
1 1 1 0 1
0 1 0 0 1
0 1 1 1 0
1 1 0 0 0
1 0 0 0 0
0 1 0 0 0
0 0 1
1 0 0
1 1 1
0 0 0
1 0 0

7 4 8
```

```
1 0 1 1
1 0 0 1
1 1 1 0
1 1 1 1
1 1 0 1
1 1 0 0
0 1 1 1
0 1 0 1 1 1 0 1
1 1 1 1 1 0 1 1
0 0 1 1 0 1 0 1
1 0 1 0 0 1 1 0

3 5 4
1 0 0 0 0
0 0 1 0 1
0 1 1 0 0
0 1 1 0
1 1 1 0
1 1 1 1
0 1 0 1
0 0 1 0

4 6 7
0 0 1 0 1 1
0 1 1 0 1 1
1 0 0 0 1 1
0 0 0 0 1 1
0 1 1 0 0 1 1
0 0 1 0 1 0 0
1 1 1 0 0 0 1
0 0 1 1 0 0 1
0 1 0 1 1 0 1
0 0 0 0 1 0 0

9 9 3
1 1 1 1 1 0 1 0 0
0 0 0 0 0 1 0 1 0
0 1 0 0 1 0 0 0 0
1 1 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 1
1 1 1 1 0 0 1 1 1
0 1 1 1 1 0 1 0 1
0 1 0 1 0 1 1 1 1
0 1 0 1 0 0 1 1 0
0 0 0
0 0 0
1 0 0
1 0 1
0 1 0
1 0 1
1 1 0
0 0 0
0 1 1

7 9 6
0 0 1 1 0 1 1 1 1
1 0 0 0 0 1 0 0 1
0 0 0 1 0 0 0 1 0
```

```
1 0 1 1 1 0 0 1 1
1 1 1 0 1 1 1 1 0
1 0 1 0 1 1 1 0 0
0 0 1 0 1 0 0 1 0
1 1 0 0 0 1
0 1 0 1 1 1
0 1 0 0 0 1
1 0 0 1 0 1
1 1 0 1 0 0
1 0 1 1 0 1
0 1 1 1 0 1
1 0 1 1 0 0
1 1 1 1 1 0

9 3 8
0 0 0
0 1 1
0 1 1
0 0 0
0 1 0
1 1 1
0 1 1
1 0 1
1 0 0
1 1 1 0 0 0 0 0
1 0 0 1 0 1 1 1
1 0 1 1 0 1 0 0

9 3 4
1 1 0
1 1 0
0 1 1
0 0 0
1 1 1
1 1 0
0 1 0
0 1 0
0 1 1
1 0 0 0
0 1 0 0
1 1 0 0

5 3 7
0 1 1
0 1 1
1 0 1
0 1 0
1 0 1
1 0 0 1 1 0 1
1 1 0 0 1 1 0
1 1 0 0 0 1 1
```

逆关系

题目描述

输入关系R，输出关系的逆关系。

程序代码

```
#include<stdio.h>

int matrix[100][100];
void introduction();
int main()
{
    introduction();
    int dom,ran;
    scanf("%d %d",&dom,&ran);
    for(int i=0;i<dom;i++)
    {
        for(int j=0;j<ran;j++)
        {
            scanf("%d",&matrix[i][j]);
        }
    }

    printf("关系R的逆关系为:\n");
    for(int i=0;i<dom;i++)
    {
        for(int j=0;j<ran;j++) if(matrix[i][j])
            printf("<%d,%d> ",j+1,i+1);
    }


}

void introduction()
{
    printf("①输入格式：\n");
    printf("1) 输入关系R的前域元素个数和培域元素个数，并且R关系的前域和培域都是从1开始的连续正整数（1，2，3，4……）。\n");
    printf("2) 输入关系矩阵\n");
    printf("②输出格式：\n");
    printf("以若干组 <a,b>的形式输出逆关系\n");
    printf("样例输入：\n");
    printf("3 3\n1 1 1\n0 0 0\n0 0 0\n");
    printf("样例输出：\n");
    printf("<1,1> <2,1> <3,1>\n");
}
```

测试数据

```
# 共十组数据，每组之间用空格分隔
# 输入格式：第一行三个整数 x，y，表示接下来要输入一个 x * y 的关系矩阵
# 后面的 x 行，每行 y 个整数，且为 0 或者 1 ，表示关系
# 输出：关系矩阵的逆关系

8 7
1 0 0 0 0 0 0
```

```
1 0 1 1 0 0 1
0 1 0 1 1 0 1
1 0 1 1 1 1 0
1 0 0 1 0 0 0
0 1 1 1 0 1 0
1 1 0 1 0 1 0
0 1 0 1 1 1 0

6 9
1 0 0 1 1 0 0 0 0
1 1 1 1 0 0 1 1 1
1 1 1 0 1 1 1 1 0
1 1 0 0 1 0 0 1 1
1 0 0 0 0 0 0 1 1
0 0 0 0 0 1 0 1 1

5 6
1 1 1 0 1 0
1 0 0 1 1 1
0 0 0 1 0 0
1 1 0 0 0 1
0 0 0 0 0 0

5 8
0 0 0 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 1 0 1 0
1 1 1 1 1 0 0 0
1 0 1 0 0 1 1 1

6 6
0 1 1 1 0 0
0 0 1 1 1 1
0 1 1 0 1 0
0 0 1 1 1 0
1 0 0 0 1 1
0 1 0 0 0 1

9 4
1 0 0 0
1 1 0 0
0 0 0 0
1 1 1 1
1 1 1 1
0 1 1 1
1 1 1 0
1 1 1 1
1 0 0 1

3 8
1 0 1 0 1 1 0 1
1 1 0 1 0 1 1 1
1 0 0 1 0 0 1 1

7 3
1 1 1
1 1 0
1 1 0
```

```
0 1 1
0 0 1
0 1 1
0 0 1

8 5
1 0 1 0 1
1 1 0 0 0
1 0 1 0 1
0 1 1 0 1
0 1 0 1 0
0 1 0 0 0
1 1 1 0 0
0 1 1 1 0

3 3
0 0 0
1 0 0
1 0 1
```

三元关系

题目描述

输出所有的三元关系，满足:

$$\{(a,b,c)|a,b,and\ c\ are\ integers\ with\ 0<a<b<c<5\} \tag{1}$$

程序代码

```c
#include<stdio.h>
int main()
{
    int i = 1, j = 2, k = 3;
    for(i = 1;i < 5;i++)
    {
        for(j = 1;j < i;j++)
        {
            for(k = 1;k < j;k++)
            {
                printf("(%d,%d,%d)\n",k,j,i);
            }
        }
    }
    return 0;
}

```

四元关系

题目描述

输出所有的四元关系，满足:

$$\{(a,b,c,d)|a,b,c,and\ d\ are\ positive\ integers\ with\ abcd=6\} \tag{2}$$

程序代码

```
#include<cstdio>
#include<algorithm>
using namespace std;
int main()
{
    int nums1[] = {1,1,2,3};
    int nums2[] = {1,1,1,6};
    printf("4-tuples in the relation---\n{(a,b,c,d)|a,b,c and d are positive integers with abcd=6}:\n");
    for (int i=0;i<12;i++)
    {
        next_permutation(nums1,nums1+4);
        printf("(%d,%d,%d,%d)\n",nums1[0],nums1[1],nums1[2],nums1[3]);
    }
    for (int j=0;j<4;j++)
    {
        next_permutation(nums2,nums2+4);
        printf("(%d,%d,%d,%d)\n",nums2[0],nums2[1],nums2[2],nums2[3]);
    }
    return 0;
}

```

五元关系

题目描述

输出下表所有的五元关系

TABLE 8 Flights.

Airline	*Flight_number*	*Gate*	*Destination*	*Departure_time*
Nadir	122	34	Detroit	08:10
Acme	221	22	Denver	08:17
Acme	122	33	Anchorage	08:22
Acme	323	34	Honolulu	08:30
Nadir	199	13	Detroit	08:47
Acme	222	22	Denver	09:10
Nadir	322	34	Detroit	09:44

程序代码

```
relation = [
    ('Nadir', 122, 34, 'Detroit', '08:10'),
    ('Acme', 221, 22, 'Denver', '08:17'),
    ('Acme', 122, 33, 'Anchorage', '08:22'),
    ('Acme', 323, 34, 'Honolulu', '08:30'),
    ('Nadir', 199, 13, 'Detroit', '08:47'),
    ('Acme', 222, 22, 'Denver', '09:10'),
    ('Nadir', 322, 34, 'Detroit', '09:44')
    ]

for i in relation:
    print(i)
```

数据生成器

所有所需要的测试数据为随机生成，生成器代码如下：

```
import random

def generate01(n, m):
    for i in range(n):
        for j in range(m):
            print(random.randint(0, 1), end=' ')
        print()

def generateRS(low = 3, high = 9):
    x = random.randint(low, high)
    y = random.randint(low, high)
    z = random.randint(low, high)
    print(x, y, z)
    generate01(x, y)
    generate01(y, z)

def generateR(low = 3, high = 9):
    x = random.randint(low, high)
    y = random.randint(low, high)
    print(x, y)
    generate01(x, y)

def generate(cnt, func):
    for i in range(cnt):
        func()
        print()

if __name__ == '__main__':
    generate(10, generateRS)
    generate(10, generateR)
    pass

```