

19组离散作业

19组离散作业

合成关系

题目描述

程序代码

测试数据

逆关系

题目描述

程序代码

测试数据

三元关系

题目描述

程序代码

四元关系

题目描述

程序代码

五元关系

题目描述

程序代码

数据生成器

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

```
1  #include<stdio.h>
2  int main()
3  {
4      int i,j,k;
5      int a,b,c;
6      scanf("%d %d %d",&a,&b,&c);
7
8      int R[50][50] = {0},S[50][50] = {0};
9      for(i = 0;i < a;i++)
10         for(j = 0;j < b;j++)
11             scanf("%d",&R[i][j]);
12     for(i = 0;i < b;i++)
13         for(j = 0;j < c;j++)
14             scanf("%d",&S[i][j]);
15
16     int d[50][50];
17     for(i = 0;i < a;i++)
18     {
19         for(j = 0;j < c;j++)
```

```

20     {
21         d[i][j] = 0;
22         for(k = 0; k < b; k++)
23             d[i][j] = d[i][j] + R[i][k] * S[k][j];
24     }
25 }
26
27 printf("\n");
28 for(i = 0; i < a; i++)
29 {
30     for(j = 0; j < c; j++)
31         if(d[i][j] != 0) printf("1 ");
32         else printf("0 ");
33     printf("\n");
34 }
35
36 return 0;
37 }
38

```

测试数据

```

1  # 共十组数据，每组之间用空格分隔
2  # 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
3  # 后面的 x 行，每行 y 个整数，且为 0 或者 1，表示关系
4  # 后面的 y 行，每行 z 个整数，且为 0 或者 1，表示关系
5  # 输出：两个关系矩阵的合成关系
6
7  6 3 9
8  0 1 1
9  0 1 0
10 0 1 0
11 0 0 1
12 0 0 0
13 0 0 1
14 1 0 1 0 0 1 0 0 1
15 0 1 1 1 0 0 0 1 0
16 0 1 0 0 0 1 1 0 0
17
18 9 5 3
19 1 0 1 1 0
20 1 0 0 1 1
21 1 1 1 1 0
22 1 1 1 0 1
23 0 1 0 0 1
24 0 1 1 1 0
25 1 1 0 0 0
26 1 0 0 0 0
27 0 1 0 0 0
28 0 0 1
29 1 0 0
30 1 1 1
31 0 0 0
32 1 0 0
33
34 7 4 8

```

35	1 0 1 1
36	1 0 0 1
37	1 1 1 0
38	1 1 1 1
39	1 1 0 1
40	1 1 0 0
41	0 1 1 1
42	0 1 0 1 1 1 0 1
43	1 1 1 1 1 0 1 1
44	0 0 1 1 0 1 0 1
45	1 0 1 0 0 1 1 0
46	
47	3 5 4
48	1 0 0 0 0
49	0 0 1 0 1
50	0 1 1 0 0
51	0 1 1 0
52	1 1 1 0
53	1 1 1 1
54	0 1 0 1
55	0 0 1 0
56	
57	4 6 7
58	0 0 1 0 1 1
59	0 1 1 0 1 1
60	1 0 0 0 1 1
61	0 0 0 0 1 1
62	0 1 1 0 0 1 1
63	0 0 1 0 1 0 0
64	1 1 1 0 0 0 1
65	0 0 1 1 0 0 1
66	0 1 0 1 1 0 1
67	0 0 0 0 1 0 0
68	
69	9 9 3
70	1 1 1 1 1 0 1 0 0
71	0 0 0 0 0 1 0 1 0
72	0 1 0 0 1 0 0 0 0
73	1 1 0 0 0 1 1 1 1
74	1 0 0 0 0 0 0 0 1
75	1 1 1 1 0 0 1 1 1
76	0 1 1 1 1 0 1 0 1
77	0 1 0 1 0 1 1 1 1
78	0 1 0 1 0 0 1 1 0
79	0 0 0
80	0 0 0
81	1 0 0
82	1 0 1
83	0 1 0
84	1 0 1
85	1 1 0
86	0 0 0
87	0 1 1
88	
89	7 9 6
90	0 0 1 1 0 1 1 1 1
91	1 0 0 0 0 1 0 0 1
92	0 0 0 1 0 0 0 1 0

93	1 0 1 1 1 0 0 1 1
94	1 1 1 0 1 1 1 1 0
95	1 0 1 0 1 1 1 0 0
96	0 0 1 0 1 0 0 1 0
97	1 1 0 0 0 1
98	0 1 0 1 1 1
99	0 1 0 0 0 1
100	1 0 0 1 0 1
101	1 1 0 1 0 0
102	1 0 1 1 0 1
103	0 1 1 1 0 1
104	1 0 1 1 0 0
105	1 1 1 1 1 0
106	
107	9 3 8
108	0 0 0
109	0 1 1
110	0 1 1
111	0 0 0
112	0 1 0
113	1 1 1
114	0 1 1
115	1 0 1
116	1 0 0
117	1 1 1 0 0 0 0 0
118	1 0 0 1 0 1 1 1
119	1 0 1 1 0 1 0 0
120	
121	9 3 4
122	1 1 0
123	1 1 0
124	0 1 1
125	0 0 0
126	1 1 1
127	1 1 0
128	0 1 0
129	0 1 0
130	0 1 1
131	1 0 0 0
132	0 1 0 0
133	1 1 0 0
134	
135	5 3 7
136	0 1 1
137	0 1 1
138	1 0 1
139	0 1 0
140	1 0 1
141	1 0 0 1 1 0 1
142	1 1 0 0 1 1 0
143	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

```
1  #include<stdio.h>
2
3  int matrix[100][100];
4  void introduction();
5  int main()
6  {
7      introduction();
8      int dom,ran;
9      scanf("%d %d",&dom,&ran);
10     for(int i=0;i<dom;i++)
11     {
12         for(int j=0;j<ran;j++)
13         {
14             scanf("%d",&matrix[i][j]);
15         }
16     }
17
18     printf("关系R的逆关系为:\n");
19     for(int i=0;i<dom;i++)
20     {
21         for(int j=0;j<ran;j++) if(matrix[i][j])
22             printf("<%d,%d> ",j+1,i+1);
23     }
24
25
26 }
27
28 void introduction()
29 {
30     printf("@输入格式: \n");
31     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");
32     printf("2) 输入关系矩阵\n");
33     printf("@输出格式: \n");
34     printf("以若干组 <a,b>的形式输出逆关系\n");
35     printf("样例输入: \n");
36     printf("3 3\n1 1 1\n0 0 0\n0 0 0\n");
37     printf("样例输出: \n");
38     printf("<1,1> <2,1> <3,1>\n");
39 }
```

测试数据

```
1  # 共十组数据，每组之间用空格分隔
2  # 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
3  # 后面的 x 行，每行 y 个整数，且为 0 或者 1，表示关系
4  # 输出：关系矩阵的逆关系
5
6  8 7
7  1 0 0 0 0 0 0
```

8	1 0 1 1 0 0 1
9	0 1 0 1 1 0 1
10	1 0 1 1 1 1 0
11	1 0 0 1 0 0 0
12	0 1 1 1 0 1 0
13	1 1 0 1 0 1 0
14	0 1 0 1 1 1 0
15	
16	6 9
17	1 0 0 1 1 0 0 0 0
18	1 1 1 1 0 0 1 1 1
19	1 1 1 0 1 1 1 1 0
20	1 1 0 0 1 0 0 1 1
21	1 0 0 0 0 0 0 1 1
22	0 0 0 0 0 1 0 1 1
23	
24	5 6
25	1 1 1 0 1 0
26	1 0 0 1 1 1
27	0 0 0 1 0 0
28	1 1 0 0 0 1
29	0 0 0 0 0 0
30	
31	5 8
32	0 0 0 1 1 1 1 0
33	0 1 0 1 1 0 1 0
34	0 1 1 0 1 0 1 0
35	1 1 1 1 1 0 0 0
36	1 0 1 0 0 1 1 1
37	
38	6 6
39	0 1 1 1 0 0
40	0 0 1 1 1 1
41	0 1 1 0 1 0
42	0 0 1 1 1 0
43	1 0 0 0 1 1
44	0 1 0 0 0 1
45	
46	9 4
47	1 0 0 0
48	1 1 0 0
49	0 0 0 0
50	1 1 1 1
51	1 1 1 1
52	0 1 1 1
53	1 1 1 0
54	1 1 1 1
55	1 0 0 1
56	
57	3 8
58	1 0 1 0 1 1 0 1
59	1 1 0 1 0 1 1 1
60	1 0 0 1 0 0 1 1
61	
62	7 3
63	1 1 1
64	1 1 0
65	1 1 0

```
66 0 1 1
67 0 0 1
68 0 1 1
69 0 0 1
70
71 8 5
72 1 0 1 0 1
73 1 1 0 0 0
74 1 0 1 0 1
75 0 1 1 0 1
76 0 1 0 1 0
77 0 1 0 0 0
78 1 1 1 0 0
79 0 1 1 1 0
80
81 3 3
82 0 0 0
83 1 0 0
84 1 0 1
```

三元关系

题目描述

输出所有的三元关系，满足：

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

程序代码

```
1  #include<stdio.h>
2  int main()
3  {
4      int i = 1, j = 2, k = 3;
5      for(i = 1; i < 5; i++)
6      {
7          for(j = 1; j < i; j++)
8          {
9              for(k = 1; k < j; k++)
10             {
11                 printf("(%d,%d,%d)\n", k, j, i);
12             }
13         }
14     }
15     return 0;
16 }
17
```

四元关系

题目描述

输出所有的四元关系，满足：

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

程序代码

```

1  #include<stdio>
2  #include<algorithm>
3  using namespace std;
4  int main()
5  {
6      int nums1[] = {1,1,2,3};
7      int nums2[] = {1,1,1,6};
8      printf("4-tuples in the relation---\n{(a,b,c,d)|a,b,c and d are
positive integers with abcd=6}:\n");
9      for (int i=0;i<12;i++)
10     {
11         next_permutation(nums1,nums1+4);
12         printf("(%d,%d,%d,%d)\n",nums1[0],nums1[1],nums1[2],nums1[3]);
13     }
14     for (int j=0;j<4;j++)
15     {
16         next_permutation(nums2,nums2+4);
17         printf("(%d,%d,%d,%d)\n",nums2[0],nums2[1],nums2[2],nums2[3]);
18     }
19     return 0;
20 }
21

```

五元关系

题目描述

输出下表所有的五元关系

TABLE 8 Flights.				
<i>Airline</i>	<i>Flight_number</i>	<i>Gate</i>	<i>Destination</i>	<i>Departure_time</i>
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


```

1 relation = [
2     ('Nadir', 122, 34, 'Detroit', '08:10'),
3     ('Acme', 221, 22, 'Denver', '08:17'),
4     ('Acme', 122, 33, 'Anchorage', '08:22'),
5     ('Acme', 323, 34, 'Honolulu', '08:30'),
6     ('Nadir', 199, 13, 'Detroit', '08:47'),
7     ('Acme', 222, 22, 'Denver', '09:10'),
8     ('Nadir', 322, 34, 'Detroit', '09:44')
9 ]
10
11 for i in relation:
12     print(i)

```

数据生成器

所有需要的测试数据为随机生成，生成器代码如下：

```

1 import random
2
3 def generate01(n, m):
4     for i in range(n):
5         for j in range(m):
6             print(random.randint(0, 1), end=' ')
7         print()
8
9 def generateRS(low = 3, high = 9):
10     x = random.randint(low, high)
11     y = random.randint(low, high)
12     z = random.randint(low, high)
13     print(x, y, z)
14     generate01(x, y)
15     generate01(y, z)
16
17 def generateR(low = 3, high = 9):
18     x = random.randint(low, high)
19     y = random.randint(low, high)
20     print(x, y)
21     generate01(x, y)
22
23 def generate(cnt, func):
24     for i in range(cnt):
25         func()
26         print()
27
28 if __name__ == '__main__':
29     generate(10, generateRS)
30     generate(10, generateR)
31     pass
32

```