Zoj3861Valid Pattern LockDFS回溯

**题意：**

给定1~n这些数，按3\*3排列。求这n个数合法的模式(手势锁)数，以及按字典顺序输出这些合法模式。

合法模式要求：

1) 由首次经过的点组成。（1->3->2 X，因为到3前已经经过2）

2) 对于每两个连续点A和B,如果A和B的连线通过一些其他的点,那么这些点必须在序列中，并且经过的点在A和B前已经经过。如1->3，要经过2，则2必须在1和3前经过，所以，2->1->3是合法的。

3) 模式中的每一条边必须从一个点到达另一个未接触过的点，它也可能经过一些早就经过了的点。

输入分析：

序列1->2->3合法

序列1->3->2非法

序列2->1->3合法

序列2->3->1合法

序列3->1->2非法

序列3->2->1合法

再如：

|  |  |  |
| --- | --- | --- |
| 1 |  |  |
|  | 5 |  |
| 7 |  | 9 |

结果：

2

7 5 1 9

7 5 9 1

要注意的是，本题可能输出的模式较多，应该采用printf()节约时间。

**参考代码：**

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <vector>

#include <string>

using namespace std;

int F[10][10];//F[a][b]=0，a,b直接可达，F[a][b]=c，表示a和b间有c

vector<string> vec; //vec容器用来存储所有的合法模式，

//num数组用来存激活数，visit存构造图文时已经用过的数，cvisit存可以用的数

int num[10], visit[10], cvisit[10];

int n, ans = 0;//ans累计合法图文的数目

void init()

{

memset(F, 0, sizeof(F));

F[1][3] = F[3][1] = 2;

F[1][9] = F[9][1] = 5;

F[1][7] = F[7][1] = 4;

F[2][8] = F[8][2] = 5;

F[3][7] = F[7][3] = 5;

F[3][9] = F[9][3] = 6;

F[4][6] = F[6][4] = 5;

F[7][9] = F[9][7] = 8;

}

//搜至num[u]时的模式s，目前模式已有count个激活数

void DFS(int u, string s, int count)

{

if (count == n)

{

vec.push\_back(s);

ans++;

return;

}

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

{

if (visit[num[i]]) //这个数用过一次了，就不用把它保存到字符串中了

continue;

if (F[num[i]][num[u]]) //从num[i]到num[u]中间隔着一个数w

{

if (!cvisit[F[num[i]][num[u]]]) //这w不是激活数的话那肯定不能用

continue;

//若w是激活数但它还没有被用过那就不能直接从num[u]连到num[i]

if (!visit[F[num[i]][num[u]]])

continue;

}

string ss = s + char(num[i] + '0');

visit[num[i]] = 1;

DFS(i, ss, count + 1);

//要理解为什么要回溯。

//若继续成功找到合法模式，函数返回到本dfs(u,s,count)时，

//下次循环还要在u,新i后,重新利用上次那个数，继续寻找合法的模式。

//若上面DFS(i,ss,count+1)不成功返回到本函数，现在的num[i]后续还要再用

visit[num[i]] = 0;//回溯，该num[i]后续还要再用

//用了ss，回溯就不用改s

}

}

int main()

{

int T;

cin >> T;

init();

while (T--)

{

cin >> n;

vec.clear();

memset(num, 0, sizeof(num));

memset(visit, 0, sizeof(visit));

memset(cvisit, 0, sizeof(cvisit));

ans = 0;

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

{

cin >> num[i];

cvisit[num[i]] = 1;//num[i]是激活数

}

sort(num, num + n);//排序之后再去搜索的时候它就是按词典序存的

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

{

string s = "";

s = s + char(num[i] + '0');//每次-都以num[i]作为开头元素

visit[num[i]] = 1;//标记该点已用

DFS(i, s, 1);

visit[num[i]] = 0;//回溯重新来过找下一个

}

cout << ans << endl;

for (int i = 0; i<int(vec.size()); i++)

{

string s = vec[i];

printf("%c", s[0]);

for (int j = 1; j<n; j++)

printf(" %c", s[j]);

printf("\n");

}

}

return 0;

}