深度优先搜索
第二课

搜索例题四 四色问题 NKOJ2165

四色定理：“任何一张地图只用四种颜色就能使具有共同边界的国家着上不同的颜色。”——弗南西斯·格思里

有形如下列图形的地图，图中每一块区域代表一个国家，现请你用红（1）、兰（2）、黄（3）、绿（4）四种颜色给这些国家填上颜色，要求每一国家用一种颜色，且任意两个相邻国家的颜色不能相同，请给出一种符合条件的填色方案。给出国家的数量（<=50）和它们的邻接关系。找出所有方案，按国家编号由小到大输出。

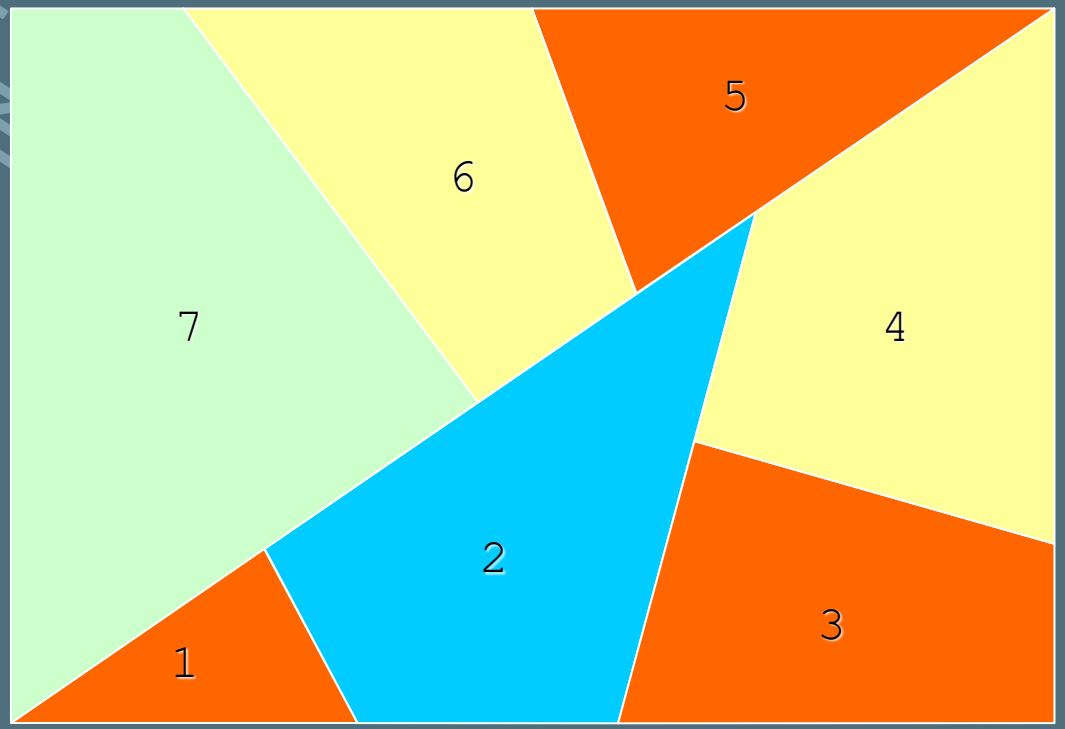
输入

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	1
2	1	0	1	1	1	1	1
3	0	1	0	1	0	0	0
4	0	1	1	0	1	0	0
5	0	1	0	1	0	1	0
6	0	1	0	0	1	0	1
7	1	1	0	0	0	1	0

输出

1213134

.....



从第一个国家开始，依次对每个国家依次尝试四种颜色，判断当前选择的颜色是否和相邻的国家相同。

若都不相同，则将此国家填上当前的颜色。

若存在相同，则取下一种颜色继续尝试。

若四种颜色都不能填上，则退回上一个国家并选择下一种颜色继续尝试。

当最后一个国家也被填色后就找到一组可行解。

```
 int Color[51];    //Color[j] 记录第j号国家的颜色，用1..4表示'a'..'d'四种颜色
 bool Map[51][51];   //Map存国家的邻接关系，Map[x][y]=1表示相邻
void search(int x)    // 搜索第x个国家，即讨论第x个国家涂什么颜色
{
     bool flag;
     if(x > n)
     {                                  //总共n个国家，若已全部着色，则输出结果
          for(int i= 1;i<=n;i++)cout<<Color[i];
          cout<<endl;
          return;
     }
   // 若还未全部填色，则继续涂色
          for(int i = 1;i<=4;i++) // 依次用四种颜色对当前第x号国家填色,当前试探第i种颜色
          {
               flag=true;        //flag用于标记当前要图的颜色是否有冲突
               for(int k=1;k<x;k++)//枚举前面已经涂色的国家，也就是第1到第x-1号国家
                    if( map[x][k] && Color[k]==i ){ flag=false; break;  }//有冲突
               if (flag==true)
               {                                          // 没有冲突
                    Color[x] = i;                         // 记录当前国家颜色
                    search(x + 1);                        // 搜索下一个国家
               }
          }
}
```

搜索例题六 图书分配 NKOJ1074

老师有n(1<=n<=10)本书要分给参加竞赛的n个学生。如：A，B，C，D，E共5本书要分给参加培训的简、温、张、赵、韩5位学生，每人只能选1本。教师事先让每个人将自己喜爱的书填写在如下的表中，然后根据他们填写的表来分配书本。请你设计一个程序帮助教师求出可能的分配方案，使每个学生都满意。

输入格式：第一行一个数n（学生的个数，书的数量）

以下共n行，每行n个0或1（由空格隔开），第i行数据表示第i个同学对所有书的喜爱情况。0表示不喜欢该书，1表示喜欢该书。

输出格式：依次输出每个学生分到的书号(输出所有可行的方案)和总共有多少种可行方案。

样例：

输入

```
5
0 0 1 1 0
1 1 0 0 0
0 1 1 0 0
0 0 0 1 0
0 1 0 0 1
```

输出

```
3 1 2 4 5
1
```

	A	B	C	D	E
简			√	√	
温	√	√			
张		√	√		
赵				√	
韩		√			√

```
const int maxn=11;
bool like[maxn][maxn],book[maxn];              //book[i]第i本书, 可借为true,不可借为false
int  ans[maxn];                                //ans[i]记录第i个人借的书本的编号

void DFS(int i)                        //搜索第i个人可以借的书
{
     if(i==n+1)输出结果;                //i=n+1表示n个人已讨论完,找到了一种方案
     else
    {
         for(int j=1; j<=n; j++)              //枚举第i号人可以借的书
         if((book[j]) && (like[i][j]))        //如果第j本书可借, 并且第i个人喜欢这本书,借出
         {
              ans[i]=j;                       //记录下第i个人借的书的编号
              book[j]=false;                  //将这本书设置为已借出
              DFS(i+1);                       //搜索下一个人可借的书
              book[j]=true;                   //回溯时应恢复现场, 将已借的书j设置为未借出
         }
    }
}
```

搜索例题七：工作安排 NKOJ2166

n个人从事n项工作，每人只能从事一项，求最佳安排使效益最高。设有A，B，C，D，E五人从事J1，J2，J3，J4，J5五项工作，每人只能从事一项，他们的效益如下:当 A从事J5，B从事J3， C从事J4 ， D从事J1 ，E从事J2时收益最大值：50

输入：n和矩阵
输出：最大效益
输入：

```
5
13   11   10   4    7
13   10   10   8    5
5    9    7    7    4
15   12   10   11   5
10   11   8    8    4
```

输出：

```
50
```

	J1	J2	J3	J4	J5
A	13	11	10	4	7
B	13	10	10	8	5
C	5	9	7	7	4
D	15	12	10	11	5
F	10	11	8	8	4

```
const int maxn=11
int  Work[maxn][maxn],n,Ans=0;   //Work[ ][ ]存工人的工作收益表
bool mark[maxn];       //mark[i]记录第i份工作是否已有人做，false表示没有


void DFS(int k,int tot)      // k表示当前讨论第k个人，tot表示前k-1个人的总收益
{
    if(k==n+1)
     {                  //n个人已全部分配工作
      if(tot>Ans)                //判断当前这种方案的收益是否更优
           Ans=tot;              //若当前方案收益更高，更新结果Ans
      return;
     }
      for(int i=1;i<=n;i++)                //工作未分配完，讨论当前k号人可做的工作
         if(mark[i]==false)                //若i号工作没人做则分配给k
         {
              mark[i]=true;                          //将i号工作设置为已做
              DFS(k+1,tot+Work[k][i]);     //搜索第k+1个人
              mark[i]=false; //当前总收益为前k-1个人的收益t加上k的收益Work[k][i]
         }
}
```

深搜小结：

1.确定搜索顺序：按人1到n，按国家1到n，按空位1到n，……

2.确定搜索类型：资源无限型还是资源有限型

资源有限型：占用资源时做标记，完成讨论后，及时释放资源，修改标记。