**实验24 地图填色实验**

【问题描述】

1976年，美国科学家Appel和Haken利用计算机证明了：对一张地图，可以用不超过4种颜色对其填色，使得相邻的区域填上不同的颜色。现在我们就来模拟这一填色过程，输入一张行政区地图（见图10-2），用4种颜色对其填色，要求相邻的行政区域内没有相同的颜色，给出所有的填色方案，并统计方案的个数。

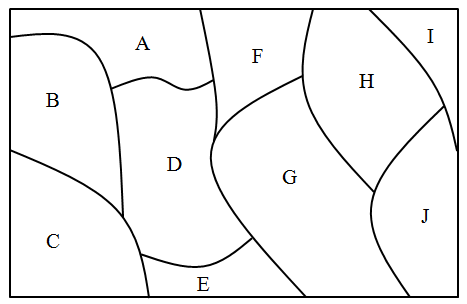


图10-2 一张行政区地图

【数据描述】

首先考虑如何存储行政区地图，由于邻接矩阵能更好地描述各行政区之间的关系，我们采用邻接矩阵G来存储地图。

  1 i，j两个行政区相邻

G[i,j] =

0 i，j两个行政区不相邻

邻接矩阵G可以用二维数组表示；另设一维数组color[i]记录第 i（i=0...n-1）个行政区所填的颜色，分别取值为：{1(红),2(黄),3(蓝),4(绿)}；数据描述如下：

int G[n][n];

int color[n+1];

【算法描述】

该问题的实现是从第一个行政区到最后一个行政区依次填色的过程，如果采用递归的算法，可以简单描述如下：

void trycolor(&color[],i){

// 设前i-1个行政区已经填好颜色，现对第i个行政区用递归法填色；

   for (c=1;c<=4;c++) {

对第i个行政区填上c颜色；

检查其合法性；

if (合法)

    if (i<n)   //递归

tricolor(&color[],i+1);

     else   //递归出口

打印一种填色方案;

} //end\_for

} //end\_tricolor

如果采用非递归的算法，可用试探和回溯来实现：对每一个行政区（用邻接矩阵表示后，可称为结点）先试填一个颜色，在确定其合法性后，再填下一个结点；如果4种颜色都填完后仍不合法，就回溯到上一个结点重新填色。算法描述如下：

Color[0..n-1]=0;

选择第一个行政区为当前要处理的结点;

do{

将当前结点顺序向后试填一个颜色;

if (所填颜色合法)

当前区域确定了所填的颜色;

if (当前结点是最后一个区域) {

打印一种填色方案；

if (所填颜色是4)

回溯到上一个结点继续填色;

else

下一个结点设为当前结点;

else

if (所填颜色是4)

回溯到上一个结点;

}while(所有的结果都试探完);

现在的问题是：如何确定合法的填色？设前i-1个行政区已经填好颜色，现对第i个行政区填色c，通过邻接矩阵的定义，我们可以分析出这样的结论：

for (k=0;k<=i-1;k++)

if (color[k]\*G[i][k]==c) {

i与第k个结点相邻并同色，c是不合法的;

}

所以我们可以用color[k] \* G[i][k] != color[i]来确定当前color[i]的颜色是合法的。

如何回溯？如果对当前结点，从颜色1到颜色4都不合法，则要退到上一个结点重新填色，如果上一个结点也填了4种颜色，就继续后退；这就是回溯的思想。

【C源程序】

#include "stdio.h"

#define n 10

int G[n][n]={ {1,1,0,1,0,1,0,0,0,0},

{1,1,1,1,0,0,0,0,0,0},

{0,1,1,1,1,0,0,0,0,0},

{1,1,1,1,1,1,1,0,0,0},

{0,0,1,1,1,0,1,0,0,0},

{1,0,0,1,0,1,1,1,0,0},

{0,0,0,1,1,1,1,1,0,1},

{0,0,0,0,0,1,1,1,1,1},

{0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,1,1,1,1}

};

void backtrack(int color[],int \*i){ //回溯过程

if (\*i>0)

while (color[\*i]==4)

(\*i)--;

}

int check(int color[], int \*i){ //检查是否合法

int k,b;

b=1;

for(k=0;k<=\*i-1;k++)

if(color[k]\*G[\*i][k]==color[\*i])

b=0;

if(b)

return 1;

else

return 0;

}

void print (int color[]){ //打印一种填色方案

int k;

for(k=0;k<n;k++)

printf("%d",color[k]);

printf("\n");

}

void main(){

int i,total=0;

int color[n];

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

color[i]=0;

i=0;

do{

color[i]++;

if(check(color,&i))

if (i==n-1){

total++;

printf("%d: ",total);

print(color);

if(color[i]==4)

backtrack(color,&i);

}

else{

i=i+1;

color[i]=0;

}

else

if (color[i]==4)

backtrack(color,&i);

}while (i);

printf("\n");

printf("total = %d",total);

scanf("\n");

}

【测试结果】

以下是图10-2的部分填色方案和计算出的所有填色方案总和：

…………

519: 1432441324

520: 1432441342

521: 1432443124

522: 1432443132

523: 1432443134

524: 1432443142

525: 1432443214

526: 1432443231

527: 1432443234

528: 1432443241

total = 528

【说明】

本实验给出的方法称为回溯，这种方法可以试探性地按一定的规律例举出所有的可能性，最终找到目标结点。一般性的回溯问题可用以下算法简单描述其思想：

初始化;

do {

可行（合法）则进;     //按一定顺序试探前进

不行（不合法）则换;   //判断其可行性，对不可行的换成下一个试探

换不成则退;           //对当前结点所有试探都不可行，回溯到上一个结点

}while(所有试探都完成);

【实验题】

1. 对行政区地图改用邻接表存储，程序应该怎样改动？
2. 将4种颜色用数字表示{1(红)，2(黄)，3(蓝)，4(绿)} 改为用字符表示{‘R’(红)}，’Y’(黄)， ‘B’(蓝)，‘G’(绿)}，改写程序完成字符的填色方案。
3. 编程递归实现地图填色问题。