第 9 章　平面图及图的着色问题

平面图是一类重要的图，应用非常广泛，例如集成电路的设计就需要用到平面图的相关理论。与平面图有密切关系的一个图论应用问题是图的着色，包括顶点着色、边着色和面着色，研究图的着色有着重要的实际意义。本章将集中讨论平面图和图的着色这两个问题。

9.1　基 本 概 念

9.1.1　平面图与非平面图

假设 A～F 表示 6 个村庄，要在这些村庄之间修筑如图 9.1(a)所示的路，且使得任何两条路都不交叉，能找到满足条件的方案吗？答案是不可能的，无论怎么画(如图 9.1(b)所示)，总是有边相交。而图 9.1(c)所示的无向图表面上存在相交的边，但可以把它改画成图 9.1(d)，这样就不存在相交的边了。

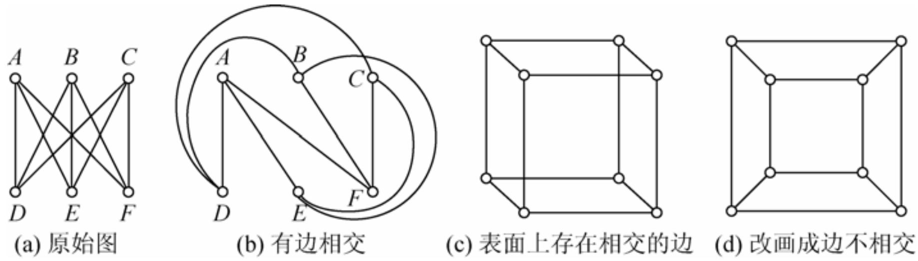


图 9.1　平面图与非平面图

平面图(Planar Graph)：设一个无向图为 $G(V, E)$，如果能把它画在平面上，且除 V 中的顶点外，任意两条边均不相交，则称该图为平面图。对平面图 G，画出其无边相交的图，称为 G 的**平面嵌入**(Planar Embedding)。无平面嵌入的图称为**非平面图**(Nonplanar Graph)。

图 9.2(a)所示的四阶完全图 K_4，存在两条相交的边，但可以将这两条相交的边改画成不相交，因此 K_4 是平面图。

又如 9.2(c)是从 K_5 中去掉了一条边，对其中相交的边可以改画成不相交，如图 9.2(d)所示，因此该图也是平面图。

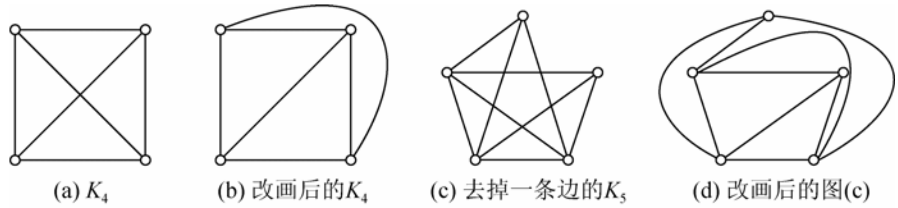


图 9.2　平面图

完全二部图 $K_{1,n}$(n≥1)和 $K_{2,n}$(n≥2)都是平面图，如图 9.3(a)、9.3(b)和 9.3(c)所示。其中，图 9.3(a)中 $K_{1,n}$ 的标准画法已经是平面嵌入了，图 9.3(b)左图所示的 $K_{2,n}$ 不是平面嵌入，但可以改画成右图所示的平面嵌入画法，图 9.3(c)所示的 $K_{3,n}$ 同样可以改画成平面嵌入画法。

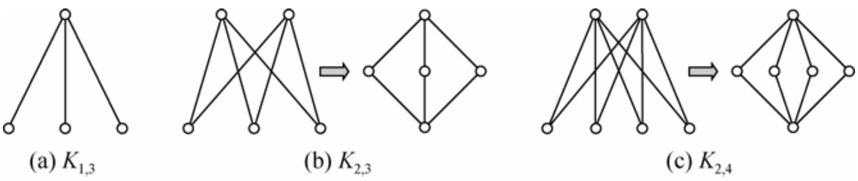


图 9.3　$K_{1,n}$ 和 $K_{2,n}$ 都是平面图

现在可以提前指出的是，在研究平面图理论中居重要地位的两个图 K_5 和 $K_{3,3}$，如图 9.4 所示，都是非平面图。

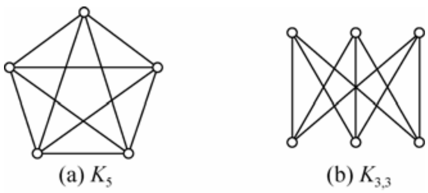


图 9.4　K_5 和 $K_{3,3}$ 都是非平面图

9.1.2　区域与边界

一个平面图将平面划分成若干个部分，每个部分称为一个区域。下面给出区域的严格定义。

区域(Region)：设 G 为一平面图(且已经是平面嵌入)，若由 G 的一条或多条边所界定的范围内不含图 G 的顶点和边，则该范围内的平面部分称为 G 的一个区域(也称为**面**，Face)，记为 R。一个平面图所划分的区域中，总有一个区域是无界的，该区域称为**外部区域**(Exterior Region)，通常记为 R_0。其他区域称为**内部区域**(Interior Region)。

区域个数：将平面图所划分的平面区域个数简称为平面图的区域个数，并用 r 表示。

例如，图 9.5(a)所示的平面图，将平面分成 6 个区域，如图 9.5(b)所示，因此，该平面图的区域个数 r = 6。其中 R_0 为外部区域，R_1～R_5 为内部区域。

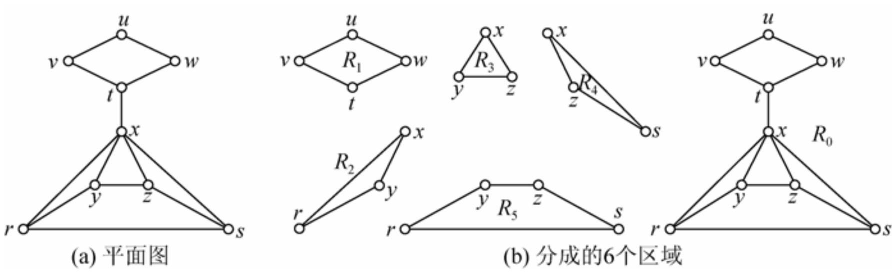


图 9.5　区域与边界

边界(Boundary)：在一个平面图中，顶点和边都与某个区域 R 关联的子图称为 R 的边界。边界实际上是平面图的一个回路，但不一定是圈(即简单回路)。

区域的度数：区域 R 的边界中，边的个数称为区域 R 的度数，记为 $\deg(R)$。

图 9.5(b)所示的区域 R_1 的边界为圈：(u, v, t, w, u)，R_1 的度数为 4；外部区域 R_0 的边界为回路：$(u, v, t, x, r, s, x, t, w, u)$，它不是一个圈，因为顶点 t 和 x 重复了，R_0 的度数为 9。

9.1.3　极大平面图与极小非平面图

极大平面图：设 G 为平面图，若在 G 的任意不相邻的顶点 u、v 之间加边(u, v)，所得图为非平面图，则称 G 为极大平面图。

例如，图 9.2(c)所示的平面图就是极大平面图，该图只有一对顶点间没有边，如果加上这条边，则在图 9.2(d)中，无论这条边怎么画，总是会和某些边相交。加上一条边后，该图变成了 5 阶完全图 K_5，因此 K_5 是一个非平面图。

极小非平面图：如果在非平面图 G 中任意删除一条边，所得图为平面图，则称 G 为极小非平面图。

例如，在 5 阶完全图 K_5 中，任意删除一条边后，都是平面图，因此 K_5 就是一个极小非平面图。

9.1.4　平面图的对偶图

设 G 是平面图，且已经是平面嵌入了，按照如下方式构造图 G^*，称 G^*为图 G 的**对偶图**(Dual Graph)。

(1) 在 G 的区域 R_i 中放置 G^*的顶点 v_i^*。

(2) 设 $e \in E(G)$，若 e 是 G 的区域 R_i 和 R_j 的公共边界，则在 G^*中，顶点 v_i^*和 v_j^*之间有一条边，记为 e^*，e^*与 e 相交，且 e^*不与图 G 中其他边相交。

(3) 若 e 为 G 中的桥，且为区域 R_i 的边界，则在 G^*的顶点 v_i^*上，存在一条从自身环(v_i^*, v_i^*)。

例如，对图 9.6(a)所示的平面图，构造对偶图的过程和结果分别如图 9.6(b)和 9.6(c)所示。在图 9.6 中，空心圆圈为平面图 G 中的顶点，实心圆圈为对偶图 G^*中的顶点。

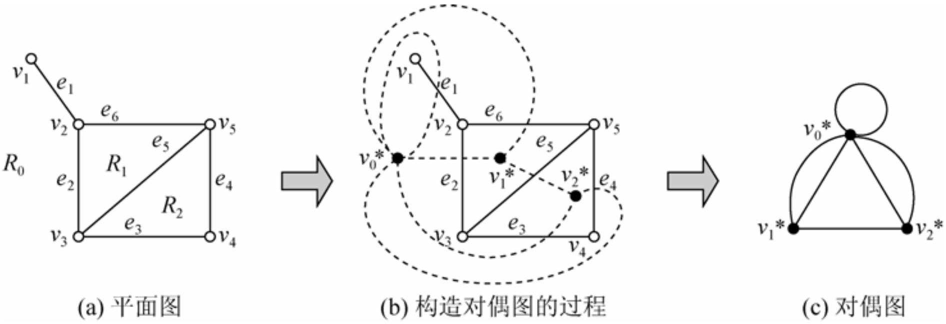


图 9.6　平面图与对偶图

由对偶图的定义不难看出，平面图 G 的对偶图 G^*有以下性质。

(1) G^*是平面图，而且在构造时各条边就互不相交，即构造时就是其平面嵌入。

(2) G^*是连通图。

(3) 若边 e 为 G 中的环，则 G^*与 e 对应的边 e^*为桥；若 e 为桥，则 G^*中与 e 对应的

边 e^*为环。

(4) 在多数情况下，G^*中含有较多的平行边。

9.1.5　关于平面图的一些定理

关于平面图，有如下一些定理(证明略)。

定理 9.1　若图 G 是平面图, 则 G 的任何子图都是平面图。

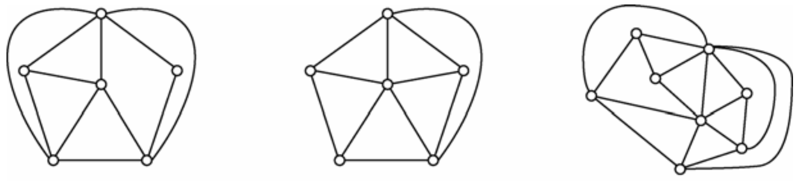
定理 9.2　若图 G 是平面图, 则在 G 中加平行边和自身环后得到的图还是平面图。

定理 9.3　平面图 G 中所有区域的度数之和等于边数 m 的两倍，即：

$$\sum_i \deg(R_i) = 2 \times m \text{。} \tag{9-1}$$

定理 9.4　设 G 为 n (n≥3)阶简单连通的平面图，G 为极大平面图，当且仅当 G 的每个区域的度数均为 3。

根据定理 9.4 可以判定图 9.7(a)和 9.7(b)为一般平面图，图 9.7(c)为极大平面图。



(a) 平面图1　(b) 平面图2　(c) 极大平面图

图 9.7　平面图与极大平面图

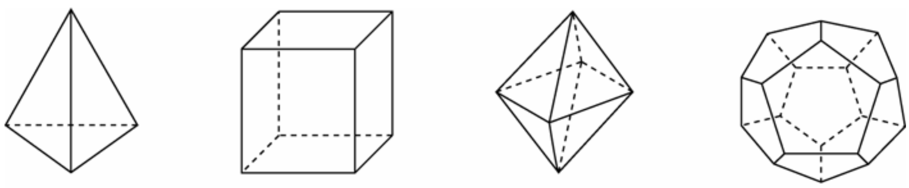
平面图 G 与它的对偶图 G^*的顶点数，边数和区域数有如下定理给出的关系。

定理 9.5　设 G^*是连通平面图 G 的对偶图，n^*、m^*、r^*和 n、m、r 分别为 G^*和 G 的顶点数、边数和面数，则有：① $n^* = r$，$m^* = m$；② $r^* = n$；③ 设 G^*的顶点 v_i^*位于 G 的区域 R_i 中，则 $deg(v_i^*) = deg(R_i)$，即对偶图中顶点 v_i^*的度数等于平面图中区域 R_i 的度数。

9.2　欧拉公式及其应用

9.2.1　欧拉公式

欧拉在研究凸多面体时发现：凸多面体顶点数减去棱数加上面数等于 2。图 9.8 分别画出了正四面体、正六面体、正八面体和正十二面体。以正十二面体为例，其顶点(即棱角)数为 20，棱数为 30，面数为 12，即：20 − 30 + 12 = 2。



(a) 正四面体　(b) 正六面体　(c) 正八面体　(d) 正十二面体

图 9.8　正多面体

后来欧拉又发现：连通平面图的阶数、边数、区域个数之间也存在同样的关系。这就是欧拉公式。

定理 9.6(欧拉公式)　如果 G 是一个阶为 n、边数为 m 且含有 r 个区域的连通平面图，则有恒等式：

$$n-m+r=2。\tag{9-2}$$

例如，对于图 9.5(a)所示的连通平面图，其阶为 9，边数为 13，含有 6 个区域，则：$9-13+6=2$。

定理 9.7(欧拉公式的推广)　对于具有 $k(k\geqslant 2)$个连通分支的平面图 G，有：$n-m+r=k+1$。其中，n、m、r 分别为 G 的阶数、边数和区域数。

9.2.2　欧拉公式的应用

以下通过一道 ACM/ICPC 试题的分析，详细介绍欧拉公式的应用。

例 9.1　美好的欧拉回路(That Nice Euler Circuit)

题目来源：

Asia 2004, Shanghai (Mainland China), ZOJ2394, POJ2284

题目描述：

Joey 发明了一种名为欧拉(为了纪念伟大的数学家欧拉)的画图机器。在 Joey 上小学时，他知道了欧拉是从一个著名的问题开始研究图。这个问题是在一张纸上一笔画出一个图形(笔尖不离开纸面)，并且笔尖要回到起点。欧拉证明了当且仅当画出的平面图形具备以下两个属性，才能按照要求画出这种图形：① 图形是连通的；② 每个顶点度数为偶数。

Joey 的欧拉机器也是这样工作的。机器中包含了一支与纸面接触的铅笔，机器的控制中心发出一系列指令指示铅笔如何画图。纸面可以看成是无限的二维平面，也就是说不必担心笔尖会超出边界。

开始画图时，机器发出一条指令，格式为(X_0, Y_0)，这意味着铅笔将移动到起点(X_0, Y_0)。接下来的每条指令的格式均为(X', Y')，表示铅笔将从当前位置移动到新位置(X', Y')，从而在纸面上画出一条线段。新位置与前面每条指令中的位置都不相同。最后，欧拉机器总是发出一条指令，将铅笔移动到起点(X_0, Y_0)。另外，欧拉机器画出来的线绝不会重叠，但有可能会相交。

当所有指令发布并执行后，在纸上已经画出一个完美的图形，由于笔尖没有离开纸面，画出来的图形可以看成是一个欧拉回路。

试计算这个欧拉回路将平面分成了多少个区域。

输入描述：

输入文件中至多包含 25 个测试数据。每个测试数据的第 1 行为一个整数 N，$N\geqslant 4$，表示测试数据中指令的数目；接下来有 N 对整数，每对整数占一行，用空格隔开，表示每条指令中的位置；第 1 对整数为起点位置。假定每个测试数据中指令的数目不超过 300，所有位置的坐标范围在(−300, 300)。$N=0$ 表示输入结束。

输出描述：

对每个测试数据，输出一行，格式为：

"Case x: There are w pieces."。

其中 x 为测试数据的序号，从 1 开始计起。

样例输入：

```
5
0 0 0 1 1 1 1 0 0 0
7
1 1 1 5 2 1 2 5 5 1 3 5 1 1
0
```

样例输出：

```
Case 1: There are 2 pieces.
Case 2: There are 5 pieces.
```

注意：样例输入所描述的两个例子如图 9.9 所示。

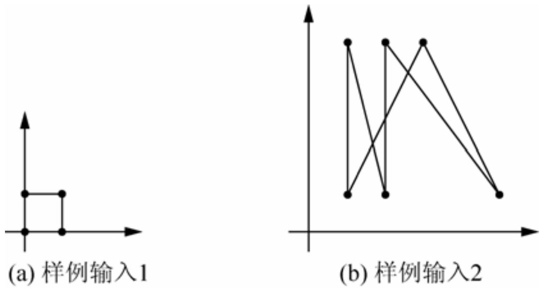


图 9.9　美好的欧拉回路

分析：

本题要根据平面图的欧拉定理“$n - m + r = 2$”来求解区域个数 r。

顶点个数的计算：两两线段求交点，每个交点都是图中的顶点。

边数的计算：在求交点时判断每个交点落在几条边上，如果一个交点落在一条边上，这条边就分裂成两条边，边数加 1。

求得顶点个数和边数后，根据欧拉定理即可求得区域个数。

代码如下：

```
#define EP 1e-10
#define MAXN 90000
struct Point       //点
{
    double x, y;
    Point( double a=0, double b=0 ){ x=a; y=b; }
};
struct LineSegment  //线段
{
    Point s, e;
    LineSegment( Point a, Point b ) { s=a; e=b; }
};
struct Line                            //直线
{
    double a, b, c;
};
bool operator<( Point p1, Point p2 )
{
    return p1.x<p2.x || p1.x==p2.x && p1.y<p2.y;
}
```

```
    bool operator==( Point p1, Point p2 )
    {
        return abs( p1.x-p2.x )<EP && abs( p1.y-p2.y )<EP;
    }
    bool Online( LineSegment l, Point p )  //判断点是否在线段上
    {
        return abs( ( l.e.x-l.s.x )*( p.y-l.s.y )-( p.x-l.s.x )*( l.e.y-l.s.y ) )
<EP&&( p.x-l.s.x )*( p.x-l.e.x )<EP&&( p.y-l.s.y )*( p.y-l.e.y )<EP;
    }
    Line MakeLine( Point p1, Point p2 )    //将线段延长为直线
    {
        Line l;
        l.a=( p2.y>p1.y )?p2.y-p1.y:p1.y-p2.y;
        l.b=( p2.y>p1.y )?p1.x-p2.x:p2.x-p1.x;
        l.c=( p2.y>p1.y )?p1.y*p2.x-p1.x*p2.y:p1.x*p2.y-p1.y*p2.x;
        return l;                          //返回直线
    }
    bool LineIntersect( Line l1, Line l2, Point &p )//判断直线是否相交,并求交点p
    {
        double d=l1.a * l2.b-l2.a*l1.b;
        if( abs( d ) < EP ) return false;
        //求交点
        p.x=( l2.c*l1.b-l1.c*l2.b )/d;
        p.y=( l2.a*l1.c-l1.a*l2.c )/d;
        return true;
    }
    //判断线段是否相交
    bool LineSegmentIntersect( LineSegment l1, LineSegment l2, Point &p )
    {
        Line a, b;
        a=MakeLine( l1.s, l1.e ), b=MakeLine( l2.s, l2.e ); //将线段延长为直线
        if( LineIntersect( a, b, p ) ) //如果直线相交
            //判断直线交点是否在线段上,是则线段相交
            return Online( l1, p ) && Online( l2, p );
        else return false;
    }
    Point p[MAXN], Intersection[MAXN];
    int N, m, n;
    int main( )
    {
        int i, j, Case=1;
        while( scanf( "%d", &N ) && N != 0 )
        {
            m=0, n=0;
            for( i=0; i<N; i++ )scanf( "%lf%lf", &p[i].x, &p[i].y );//输入数据
            for( i=0; i<N; i++ )
            {
                for( j=0; j<N; j++ )
                {
                    LineSegment l1(p[i],p[(i+1)%N]), l2(p[j], p[(j+1)%N]);
                    Point p;
```

```
                    if( LineSegmentIntersect( l1, l2, p ) )
                        Intersection[n++]=p;       //记录交点
                }
            }
            sort( Intersection, Intersection+n );  //排序
            //unique 移除重复点,求得 n
            n=unique( Intersection, Intersection+n )-Intersection;
            for( i=0; i<n; i++ )
            {
                for( j=0; j<N; j++ )
                {
                    LineSegment t( p[j], p[( j+1 ) % N] );
                    //若有交点落在边上,则该边分裂成两条边
                    if(Online(t,Intersection[i])&&!(t.s==Intersection[i]))m++;
    }
            }
            //输出欧拉定理的结果
            printf( "Case %d: There are %d pieces.\n", Case++, 2+m-n );
        }
        return 0;
    }
```

练　　习

9.1　圆(Circles)

题目来源：

Amdrew Stankvich’s Contest #5, IOJ2589

题目描述：

考虑平面上 N 个不同的圆，它们将平面分成若干个部分。试计算这些圆将平面分成了几个部分。

输入描述：

输入文件中包含了多个测试数据。输入文件的第 1 行为一个整数 T，$1 \leqslant T \leqslant 20$，表示测试数据的数目。接下来是 T 个测试数据，测试数据间用空行隔开。每个测试数据的第 1 行为一个整数 N，$1 \leqslant N \leqslant 50$，表示圆的个数，接下来有 N 行，每行为 3 个整数：x_0，y_0，r，分别表示圆心坐标和圆的半径。所有坐标位置的绝对值不超过 10^3，半径为整数，且不超过 10^3。任何两个圆都不会重合。

输出描述：

对每个测试数据，输出一行，为一个整数 K，表示这 N 个圆将平面分成了 N 个区域。

注意：由于浮点数精度的原因，在计算时不考虑面积小于 10^{-10} 的区域。

样例输入：	样例输出：
2	3
	3
2	
0 0 3	
0 0 2	

```
2
0 0 1
2 0 1
```

9.3　平面图的判定

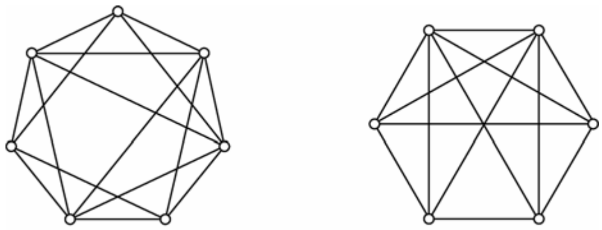
本节介绍判定一个图是否为平面图的一些结论和定理。

定理 9.8　如果 G 是一个阶 $n \geqslant 3$，边数为 m 的平面图，则 $m \leqslant 3n - 6$。证明略。

该定理给出了一个图是平面图的必要条件。从另一个角度看，这也是一个图是非平面图的充分条件。因此，可以得到定理 9.8 的逆否命题：

推论 1　如果 G 是一个阶 $n \geqslant 3$，边数为 $m > 3n - 6$ 的图，则图 G 是非平面图。

根据此推论，可以判定图 9.10(a)和 9.10(b)都是非平面图。其中，在图 9.10(a)中，顶点数 $n = 7$，边数 $m = 16 > 3 \times 7 - 6$；在图 9.10(b)中，顶点数 $n = 6$，边数 $m = 13 > 3 \times 6 - 6$。



(a) 非平面图1　　(b) 非平面图2

图 9.10　非平面图

从定理 9.8 还可以得到以下推论。

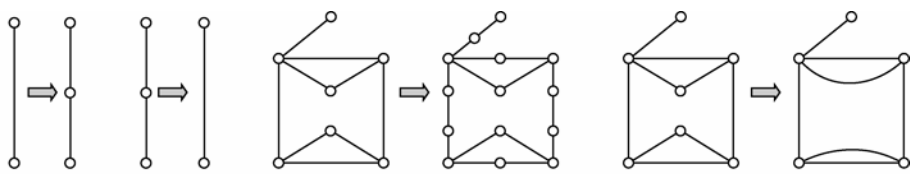
推论 2　每个平面图含有一个度小于或等于 5 的顶点。

推论 3　5 阶完全图 K_5 是非平面图。(实际上，5 阶以上的完全图都是非平面图。)

注意定理 9.8 中的条件并不是充分条件，例如对 $K_{3,3}$ 来说，由于有 6 个顶点、9 条边，因此 $3 \times 6 - 6 \geqslant 9$，即满足 $3n - 6 \geqslant m$，但可以证明 $K_{3,3}$ 是非平面图(其证明详见其他图论教材，从图 9.1(b)也可以直观地看出来，对 $K_{3,3}$ 来说，无论怎么画都存在相交的边)。

K_5 和 $i_{3,3}$ 不是平面图这个结论可以用来判定任何一个图是否是平面图，这就是下面将要介绍的 Kuratowski 定理和 Wagner 定理。

在图 9.11 中，可以看到，在给定图 G 的边上，插入一个新的度数为 2 的顶点，使一条边分成两条边(如图 9.11(a)和图 9.11(c)所示)；或者对于关联于一个度数为 2 的顶点的两条边，去掉这个顶点，使两条边化成一条边(如图 9.11(b)和图 9.11(d)所示)，这些都不会影响图的平面性。



(a) 插入顶点　(b) 删除顶点　　(c) 插入顶点　　(d) 删除顶点

图 9.11　2 度顶点内同构的图

2 度顶点内同构：给定两个图 G_1 和 G_2，如果它们是同构的，或者可以通过反复插入和(或)去掉度数为 2 的顶点后，使得 G_1 和 G_2 同构，则称 G_1 和 G_2 是在 2 度顶点内同构的。

定理 9.9(Kuratowski 定理)　一个图是平面图，当且仅当它不包含与 $K_{3,3}$ 或 K_5 在 2 度结点内同构的子图。

$K_{3,3}$ 和 K_5 常被称作**库拉托夫斯基**(Kuratowski)图。

另外，还可以通过收缩边来判定一个图是否是平面图。

收缩：设 e 是图 G 的一条边，从 G 中删去 e 并将 e 的两个顶点合并，删去由此得到的环边和平行边，这个过程称为边 e 的收缩。若图 G_1 可通过一系列边的收得到与 G_2 同构的图，则称 G_1 可以收缩到 G_2。注意，收缩边 e 并不要求 e 的两个顶点的度数为 2。

定理 9.10(Wagner 定理)　图 G 是平面图，当且仅当 G 中既没有可收缩到 K_5 的子图，也没有可收缩到 $K_{3,3}$ 的子图。

彼得森图(Peterson，如图 9.12 所示)可收缩到 K_5，如在 9.12(a)中，将粗线边收缩，则彼得森图变成了 K_5。因此，根据 Wagner 定理，可以判定彼得森图为非平面图。

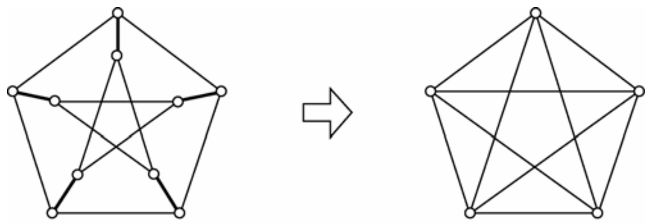


图 9.12　彼得森图可以收缩成 K_5

9.4　图的着色问题

9.4.1　地图染色与四色猜想

与平面图有密切关系的一个图论应用问题是图的着色。图的**着色问题**起源于地图染色问题和四色猜想。例如，对图 9.13 所示的美国地图，要给每个州染色，使得任何两个相邻的州颜色均不同。与 S_1 相邻的州有 S_2～S_7，要使得这 7 个州中任何两个相邻的州颜色均不同，至少需要使用 3 种颜色。图 9.13 给出了一个用 3 种颜色染色的方案：S_1 染为红色(r)，S_3、S_5、S_7 染为绿色(g)，S_2、S_4、S_6 染为蓝色(b)。现在的问题是，要给所有州染色，使得任何两个相邻的州颜色均不同，至少需要使用多少种颜色？

这个问题最早是由英国数学家弗南西斯·格思里(Francis Guthrie)于 1852 年提出来的。他发现有些地图能用 3 种颜色完成染色。他也发现，对所有地图，4 种颜色就足以完成染色，但他无法证明。此后，“每张地图都能用 4 种或者更少的颜色来染色”这个猜想开始以**四色猜想**(Four Color Conjecture)而闻名，很多数学家都尝试着去证明它。四色猜想不止一次地被有的数学家宣称证明了，但又被其他数学家证明是错误的。

1976 年，美国数学家阿佩尔(Kenneth Appel)与哈肯(Wolfgang Haken)在美国伊利诺斯大学的两台不同的电子计算机上，用了 1 200 个小时，作了 100 亿个判断，终于完成了四色定理的证明，并在当年的美国数学学会的夏季会议上，向全世界宣布他们已经证明了四色

猜想。值得一提的是，并不是所有的数学家都满意他们的证明。

图 9.13　地图染色

9.4.2　图的着色

根据着色对象不同，图的**着色问题**(Coloring Problem)分为顶点着色、边着色、和平面图的面着色。前面介绍的地图着色实际上是平面图的面着色。

1. 顶点着色

图的**顶点着色**(Vertex Coloring)：给图 G(图 G 中不存在自身环)的每个顶点指定一种颜色，使得任何两个相邻的顶点颜色均不同。

如果能用 k 种颜色对图 G 进行顶点着色，就称对图 G 进行了 ***k*** **着色**(k-Coloring)，也称 G 是 ***k*-可着色的**(k-Colorable)。若 G 是 k-可着色的，但不是(k-1)-可着色的，则称 G 是 ***k*** **色**(k-Colormatic)的图，并称这样的 k 为图 G 的**色数**(Colormatic Number)，记为 $\chi(G)$。所谓图 G 的**色数**，就是在对图 G 进行顶点着色时所用的最少颜色数。

关于顶点着色有如下一些结论和定理。

定理 9.11　$\chi(G)=1$，当且仅当 G 为零图(即边集 $E(G)$为空的图)。

定理 9.12　$\chi(K_n)=n$。

定理 9.13　奇圈的色数为 3。

定理 9.14　图 G 的色数是 2 当且仅当 G 是一个非空的二部图。

定理 9.15　对任意的图 G(图 G 中不存在自身环)，均有：$\chi(G)\leqslant\Delta(G)+1$。

定理 9.16(Brooks 定理)　设连通图 G 不是完全图 K_n，也不是奇圈，则 $\chi(G)\leqslant\Delta(G)$。

试计算图 9.14 中各图的色数。

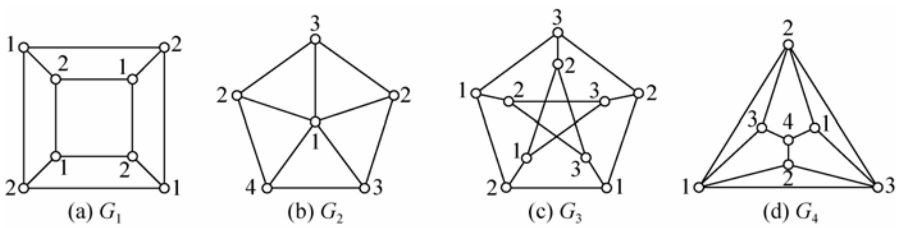


图 9.14　图的顶点着色

由定理 1.3 可知，G_1 实际上是二部图，而由定理 9.14 可知，$\chi(G_1)=2$。$\chi(G_2)=4$。图 G_3 是彼得森图，由布鲁斯定理可知，$\chi(G_3)\leqslant\Delta(G_3)=3$，又因为 G_3 中有奇圈，所以 $\chi(G_3)\geqslant3$，因此 $\chi(G_3)=3$。由布鲁斯定理可知，$\chi(G_4)\leqslant\Delta(G_4)=4$，又因为 G_3 中有奇圈，所以 $\chi(G_4)\geqslant3$，因而 $\chi(G_4)$ 为 3 或 4，但试着用 3 种颜色去着色，发现是不可能的，所以 $\chi(G_4)$ 只能为 4。

2. 边着色

图的**边着色**(Edge Coloring)：给图 G 的每条边指定一种颜色，使得任何两条相邻的边颜色均不同。

如果能用 k 种颜色对图 G 进行边着色，就称对图 G 进行了 ***k* 边着色**(k-Edge Coloring)，也称 G 是 ***k*-边可着色的**(k-Edge Colorable)。若 G 是 k-边可着色的，但不是(k-1)-边可着色的，则称 G 是 ***k* 边色**(k-Edge Colormatic)的图，并称这样的 k 为图 G 的**边色数**(Edge Colormatic Number)，记为 $\chi_1(G)$。所谓图 G 的边色数，就是在对图 G 进行边着色时所用的最少颜色数。

关于边着色有如下一些结论和定理。

定理 9.17(Vizing 定理) 对于任何一个非空简单图 G，都有：

$$\chi_1(G)=\Delta(G)，\text{或者 } \chi_1(G)=\Delta(G)+1。$$

维津(Vizing)定理说明，对简单图 G 来说，它的边色数 $\chi_1(G)$ 为 $\Delta(G)$ 或 $\Delta(G)+1$，但哪些图的边色数为 $\Delta(G)$，哪些图的边色数为 $\Delta(G)+1$，至今还是一个没有解决的问题，只是对于一些特殊的图有一些结论，如定理 9.18、9.19。

定理 9.18 设图 G 为长度大于或等于 2 的偶圈，则 $\chi_1(G)=\Delta(G)=2$；设图 G 为长度大于或等于 3 的奇圈，则 $\chi_1(G)=\Delta(G)+1=3$。

定理 9.19 对二部图 G，有：$\chi_1(G)=\Delta(G)$。

试计算图 9.15 中各图的边色数。

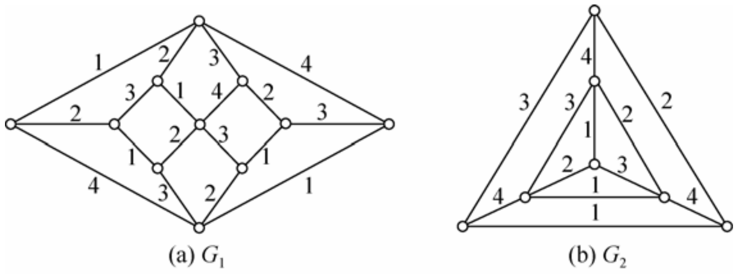


图 9.15 图的边着色

因为 G_1 中无奇数长度的回路，所以它是二部图，由定理 9.19 可知，$\chi_1(G_1)=\Delta(G_1)=4$。由维津定理可知，$\chi_1(G_2)$ 为 $\Delta(G_2)=4$，或 $\Delta(G_2)+1=5$，又存在如图 9.15(b)所示的 4 种颜色的边着色，所以 $\chi_1(G_2)=4$。

3. 地图着色与平面图的面着色

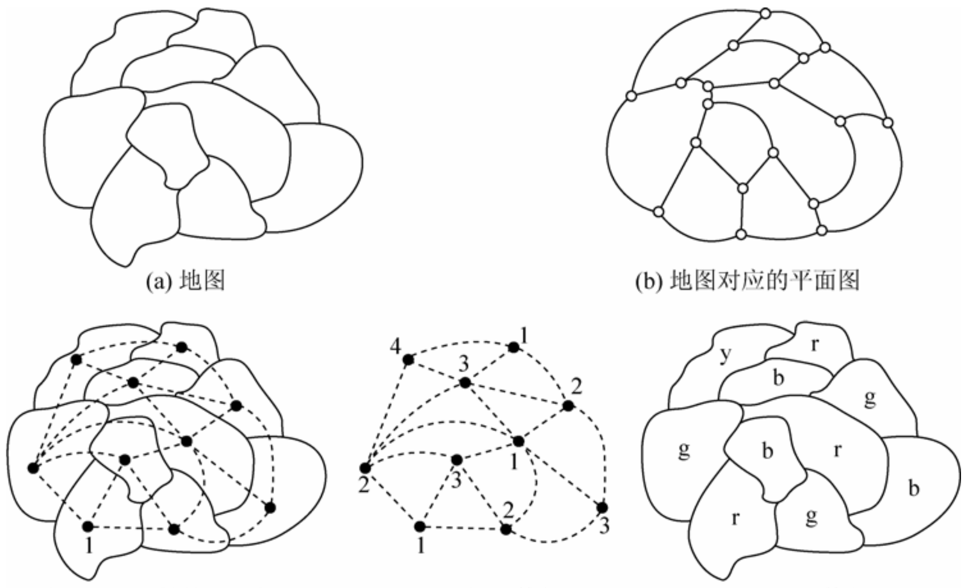
地图实际上是一种平面图：平面图的区域(即面)代表国家，边表示国家之间的边界，顶点则是边界的交汇处。例如，对图 9.16(a)所示的地图，在国家边界的每个交汇处放置一个顶点，对国家之间的每条边界用直线或曲线将对应的顶点连起来，就得到图 9.16(b)所示的平面图。

对平面图 G 来说，它将平面分成 r 个区域(即面)，现在对每个区域染色，使得有公共边的区域颜色均不同，这种染色称为平面图的**面着色**(Face Coloring)。如果能用 k 种颜色给平面图 G 进行面着色，则称 G 是 ***k*-面可着色的**(k-Face Colorable)，在进行面着色时，所用最少颜色数称为平面图的**面色数**(face Colormatic Number)，记为 $\chi^*(G)$。

平面图的面着色可以转换成其对偶图的顶点着色，依据是下面的定理 9.20。

定理 9.20　平面图 G 是 k-面可着色的，当且仅当它的对偶图 G^*是 k 色的图。

例如，对图 9.16(b)所示的平面图转换成其对偶图的过程和结果分别如图 9.16(c)和图 9.16(d)所示。在图 9.16(d)中，因为存在奇圈，所以 $\chi(G)\geqslant 3$，并且因为 $\Delta(G)=3$，根据定理 9.5，可知 $\chi(G) = 4$。图 9.16(d)给出了用 4 种颜色着色的方案，图 9.16(e)在对应的地图上用红色(r)、绿色(g)、蓝色(b)和黄色(y)4 种颜色进行着色。



(a) 地图　(b) 地图对应的平面图　(c) 将平面图图转换成对偶图　(d) 对偶图着色　(e) 地图着色

图 9.16　地图着色与对偶图顶点着色

4. 平面图面着色与四色猜想

前面已经提到过四色猜想，此处对四色猜想进一步描述。

四色猜想：连通简单平面图的色数不超过 4。

这个猜想于 1976 年由 Appel 和 Haken 宣称证明了，当然，不是所有的数学家都满意他们的证明。事实上，大部分数学家都持很高的怀疑态度，并对这个证明很不满意。这引起了关于“什么是数学证明”的许多讨论。

尽管现在四色猜想还没有被完全证明，但已经能确定的是：连通简单平面图的色数不超过 5，即以下的五色定理。

定理 9.21(五色定理)　连通简单平面图 G 的色数不超过 5。

9.4.3　图着色的应用

图着色有着丰富的应用，第 7 章的例 7.3(化学药品存放)和例 7.4(交通信号灯设计)实际

上就是求图的顶点着色。本节再举一个例子。

例 9.2　考试安排问题——图的顶点着色

某高校有 n 门选修课需要进行期末考试，同一个学生可能选修了多门课程，但他不能在同一时间段参加两门课程的考试。问该校的期末考试至少需要安排几个时间段。

例如，假设需要安排 7 门课程的期末考试，课程编号为 1～7，已知以下课程之间有公共的学生选修：(1, 2)、(1, 3)、(1, 4)、(1, 7)、(2, 3)、(2, 4)、(2, 5)、(2, 7)、(3, 4)、(3, 6)、(3, 7)、(4, 5)、(4, 6)、(5, 6)、(5, 7)、(6, 7)。

很显然，可以以课程为顶点来构图，如果两门课程之间有公共学生选修，则在这两门课程之间连边。构造好的图如图 9.17 所示，问题转换成求图的顶点着色，每种颜色的顶点安排在同一个时间段考试。该图的色数为 4，因此需要安排 4 个时间段：Ⅰ－课程 1，Ⅱ－课程 2、6，Ⅲ－课程 3、5，Ⅳ－课程 4、7。

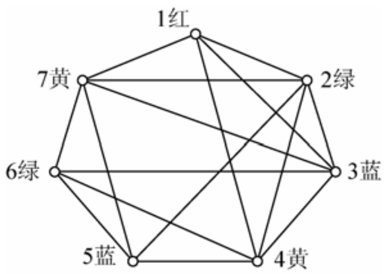


图 9.17　考试安排问题

9.4.4　图着色求解算法及例题解析

需要说明的是，尽管有很多定理来判定图的色数、边色数，但求图的色数、边色数以及具体的着色方案并没有有效的算法。本节介绍一种求 $\chi(G)$的近似有效算法——**顺序着色算法**。

设图 G 的顶点数为 n，要求对图 G 进行顶点着色，步骤如下。

(1) 用 i 表示顶点序号，$i = 1$。

(2) 用 c 表示给顶点 i 着色为第 c 种颜色，$c = 1$；

(3) 对第 i 个顶点着色：考虑它的每个邻接顶点，如果都没有使用第 c 种颜色，则给顶点 i 着色为第 c 种颜色，并转第(5)步；只要有一个顶点使用了第 c 种颜色，则转第(4)步；

(4) $c = c + 1$，并转第(3)步；

(5) 若还有顶点未着色，则 $i = i + 1$，并转向第(2)步，否则算法结束。

顺序着色算法实际上是采取了一种贪心策略：在给任何一个顶点着色时，采用其邻接顶点中没有使用的、编号最小的颜色。

以对图 9.18(a)所示的图 G 进行顶点着色为例解释顺序着色算法的执行过程。在图 9.18(b)中首先给顶点 x_1 着色为第 1 种颜色；然后在图 9.18(c)中对顶点 x_2 进行着色，因为它的邻接顶点中已经使用了第 1 种颜色，所以给 x_2 着色为第 2 种颜色；在图 9.18(d)中对顶点 x_3 进行着色，因为它的邻接顶点中没有使用第 1 种颜色，所以给顶点 x_3 着色为第 1 种颜色；……；在图 9.18(h)中，对最后一个顶点 x_7 进行着色，它的邻接顶点中，使用了第 1、3 种颜色，所以给顶点 x_7 着色为第 2 种颜色，至此着色完毕，求得 $\chi(G)=3$，并求得一个着色方案。

顺序着色算法与顶点的着色顺序有密切的关系，这就是为什么叫顺序着色算法的原因。例如，考虑图 9.19(a)所示的二部图，若按 $x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4$ 顺序执行该算法，则只需用两种颜色进行着色，如图 9.19(b)所示。但如果按 $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$ 顺序执行该算法，则需用 4 种颜色进行着色，如图 9.19(c)所示。因此，顺序着色算法并不一定有效。

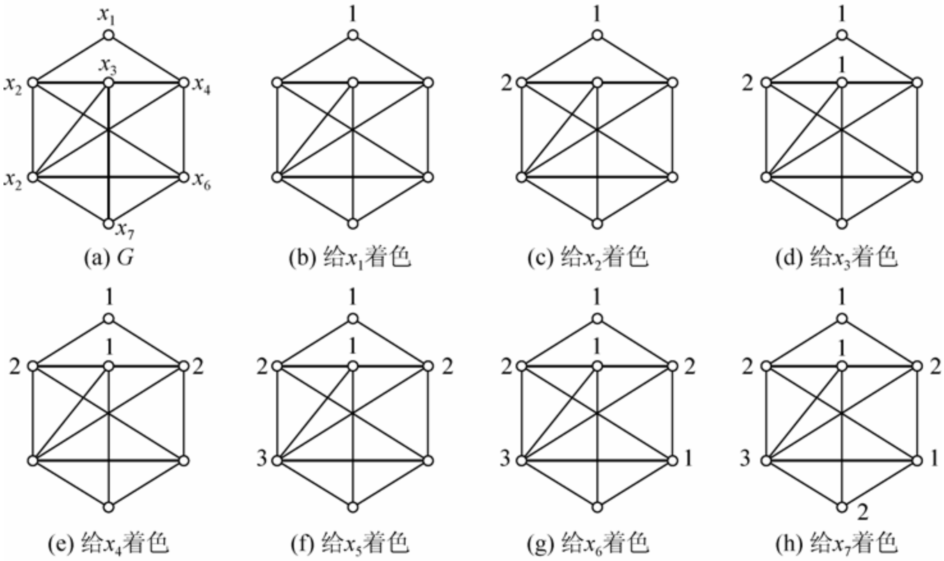


图 9.18 顺序着色算法实现过程

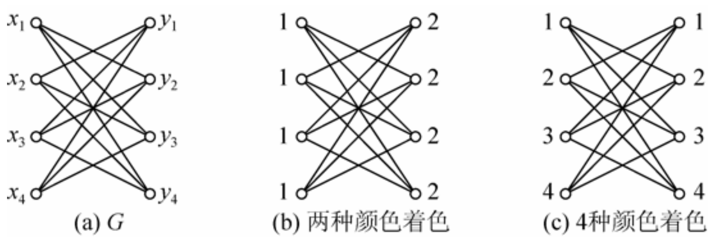


图 9.19 顺序着色算法并不一定有效

接下来通过一道 ACM/ICPC 例题，讲解有关顶点着色问题的求解和顺序着色算法的程序实现。

例 9.3 频道分配(Channel Allocation)

题目来源：

South Africa 2001, ZOJ1084, POJ1129

题目描述：

当一个广播站向一个很广的地区广播时需要使用中继器，用来转发信号，使得接收器都能接收到足够强的信号。然而，每个中继器所使用的频道必须很好地选择，以保证相邻的中继器不会互相干扰。要满足这个条件，相邻中继器必须使用不同的频道。

由于广播频率带宽是一种很宝贵的资源，对于一个给定的中继器网络，所使用频道数量应该尽可能少。编写程序，读入中继器网络的信息，计算需要使用频道的最少数目。

输入描述：

输入文件中包含多个测试数据，每个测试数据描述了一个中继器网络。每个中继器网络的格式如下。

(1) 第 1 行为一个整数 N，表示中继器的数目，$1 \leqslant N \leqslant 26$，中继器用前 N 个大写字母表示，例如，假设有 10 个中继器，则这 10 个中继器的名字为 $A, B, C, \cdots, I$ 和 J。

(2) 接下来有 N 行，描述了这 N 个中继器的相邻关系，第 1 行描述和中继器 A 相邻的中继器，第 2 行描述和中继器 B 相邻的中继器；等等。每行的格式为：

A:BCDH

表示和中继器 A 相邻的中继器有 B、C、D 和 H(按字母升序排列)。如果一个中继器没有相邻中继器，则其格式为：

A:

注意：相邻关系是对称的，A 与 B 相邻，则 B 也与 A 相邻；另外，中继器网络是一个平面图，即中继器网络所构成的图中不存在相交的边。

输入文件最后一行为 $N = 0$，表示输入结束。

输出描述：

对每个中继器网络，输出一行，为该中继器网络所需频道的最小数目。

样例输入：

```
2
A:
B:
6
A:BEF
B:ACF
C:BDF
D:CEF
E:ADF
F:ABCDE
0
```

样例输出：

```
1 channel needed.
4 channels needed.
```

分析：

很明显，本题要求的是图 G 的**色数** $\chi(G)$。样例输入中第 2 个测试数据所描述的中继器网络如图 9.20 所示。本题采用前面介绍的顺序着色算法求解，例如在图 9.20(c)中给顶点 C 着色时，它的邻接顶点中，顶点 D 和 F 目前没有着色，顶点 B 着色为第 1 种颜色，所以给顶点 C 着色为第 0 种颜色。最终的着色方案如图 9.20(d)所示，求得的 $\chi(G)$为 4。

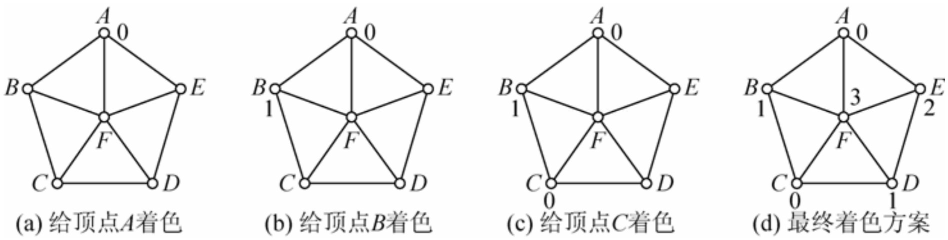


图 9.20　频道分配

代码如下：

```
char s[26];          //读入描述中继器相邻关系的每行字符串
int Edge[26][26];    //邻接矩阵
int n;               //中继器数目
int ans, c[26];      //求得的顶点着色数，各顶点的颜色
int b[26];           //每种颜色使用的标志,b[1]=1 表示第 i 种颜色已经使用了
void greedy( )
```

```
{
    int i, k;
    for( i=0; i<n; i++ )       //给第 i 个顶点着色
    {
        memset( b, 0, sizeof(b) );
        for( k=0; k<n; k++ )   //检查顶点 i 的每个邻接顶点
        {
            if( Edge[i][k] && c[k]!=-1 ) //邻接顶点 k 已经着色了，颜色为 c[k]
            {
                b[c[k]]=1;
            }
        }
        for(k=0;k<=i;k++) //k 为顶点 i 的所有邻接顶点中没有使用的、编号最小的颜色
        {
            if(!b[k])  break;
        }
        c[i]=k; //给顶点 i 着色为第 k 种颜色
    }
    for( i=0; i<n; i++ )
    {
        if( ans<c[i] )  ans=c[i];
    }
    ans++;
}
int main()
{
    int i, k;
    while( 1 )
    {
        scanf("%d", &n);
        if( n==0 )  break;
        memset( Edge, 0, sizeof(Edge) );
        for( i=0; i<n; i++ )
            c[i]=-1;
        ans=0;
        for( i=0; i<n; i++ )
        {
            scanf( "%s", &s );
            int m=strlen(s)-2;  //与第 i 个中继器相邻的中继器数目
            for( k=0; k<m; k++ )
            {
                Edge[i][s[k+2]-'A']=1;
                Edge[s[k+2]-'A'][i]=1;
            }
        }
        greedy( );
        if( ans!=1 )  printf( "%d channels needed.\n", ans );
        else  printf( "%d channel needed.\n", ans );
    }
    return 0;
}
```

练　　习

9.2　图着色(Graph Coloring)

题目来源：

Southwestern European Regional Contest 1995, POJ1419

题目描述：

编写程序，实现：对于一个给定的图，求其最优着色图。着色时用白色或黑色给图中的顶点着色，并且任何两个相邻的顶点不能同时为黑色。最优着色图是指黑色顶点数最多的着色图。

在图 9.21 中，给出一个最优着色的例子，其中包含了 3 个黑色顶点。

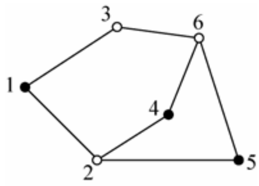


图 9.21　一幅有 3 个黑色顶点的最优着色图

输入描述：

输入文件包含多个测试数据。输入文件第 1 行为整数 m，表示测试数据个数。每个测试数据描述了一个图，其格式为：第 1 行为整数 n 和 k，$n \leqslant 100$，分别表示顶点数和边数；接下来有 k 行，每行为两个整数，表示一条边的两个顶点序号，顶点序号为 1～n。

输出描述：

对每个测试数据，输出两行。第 1 行为最优着色图中黑色顶点个数，第 2 行给出了一个最优着色方案，输出黑色顶点序号，用空格隔开。

样例输入：

```
1
6 8
1 2
1 3
2 4
2 5
3 4
3 6
4 6
5 6
```

样例输出：

```
3
1 4 5
```

9.3　着色问题(Coloration Problem)

题目来源：

Regional Contest Shang Hai, 2000, UVA2029

题目描述：

Richard 是一个数学爱好者，目前正在上海度假。他对他的研究是如此的狂热，以至于他总是把所有的事情都和数学问题联系起来。昨天黄昏，他在广场散步时，广场上铺设的

地板砖让他产生了灵感，想起了一道组合题目。

地板砖是一个等边三角形，边长为 1 米。如果把 4 块砖放在一起，可得到一个边长为 2 的大三角形。Richard 把 4 块砖编号为 1、2、3 和 4(如图 9.22(a)所示)，然后用红、绿、蓝 3 种颜色给每块砖涂上颜色。由于相同的颜色连在一起会比较难看，所以他觉得任意两块有公共边的三角砖都不能涂上相同的颜色。他很快就算出了所有的 24 种着色方案。由于砖是有编号的，所以旋转起来或者对称后看起来一样的方案也被看成是不同的。

如果用更多的三角砖组成更大的大三角形(例如用 9 块砖组成边长为 3 米的，如图 9.22(b)所示；用 16 块砖组成边长为 4 米的，……)，用更多的颜色去着色，方案数是多少？例如用 5 种颜色去给组成边长为 3 米的大三角形中的 9 块砖着色，方案数将会是一个很大的数字。没有计算机，Richard 不能得出最终的结果，尽管他知道如何去计算方案数。所以他要求你编写程序计算准确的方案数。

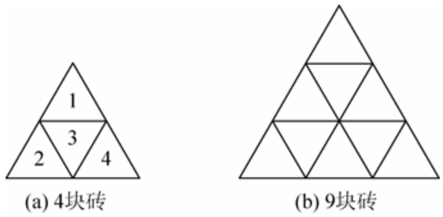


图 9.22　地板砖着色问题

输入描述：

输入文件中包含多个测试数据。每个测试数据占一行，为两个整数 L 和 C，$0 \leqslant L \leqslant 6$，$1 \leqslant C \leqslant 4$，$L$ 表示大三角形的边长，C 表示使用的颜色数。$L = 0$ 表示输入结束。

输出描述：

对每个测试数据，计算并输出着色的方案数，要求任何两个相邻小三角砖颜色不一样。如果给定的颜色数不足以按要求着色，输出 0。

样例输入：	样例输出：
2 3	24
6 4	3470494144278528
0 0	