



回溯法





思考题

1. 亚瑟王打算请150名骑士参加宴会，但是有些骑士相互间会有口角，而亚瑟王知道谁与谁不和。亚瑟王希望能让他的客人围着一张圆桌坐下，而所有不和的骑士都不会挨着坐。请回答下列问题：
- (1) 哪一个经典问题能够作为亚瑟王问题的模型？
 - (2) 设计回溯算法求解亚瑟王问题。





- 对于大部分问题来说，其解空间的规模为输入规模的指数函数甚至更高。
- 显然，在如此巨大的解空间**实施穷举**将是费时费力、**低效率**的。
- 因此，寻找更加有效的搜索手段就是算法不断推进的源动力。





- 我们要介绍的回溯法属于一种——智能穷举搜索。
- 智能穷举的核心思想，顾名思义，有两个：
穷举（在最坏情况下要对整个解空间进行指数级搜索）+ **智能**（利用各种途径减少搜索量）。





解空间树的动态搜索（1）

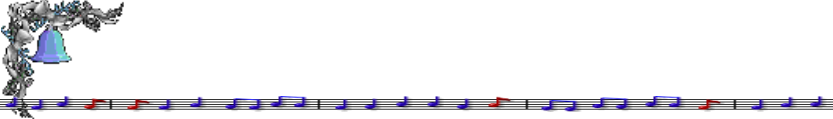
回溯法从根结点出发

按照深度优先策略遍历解空间树

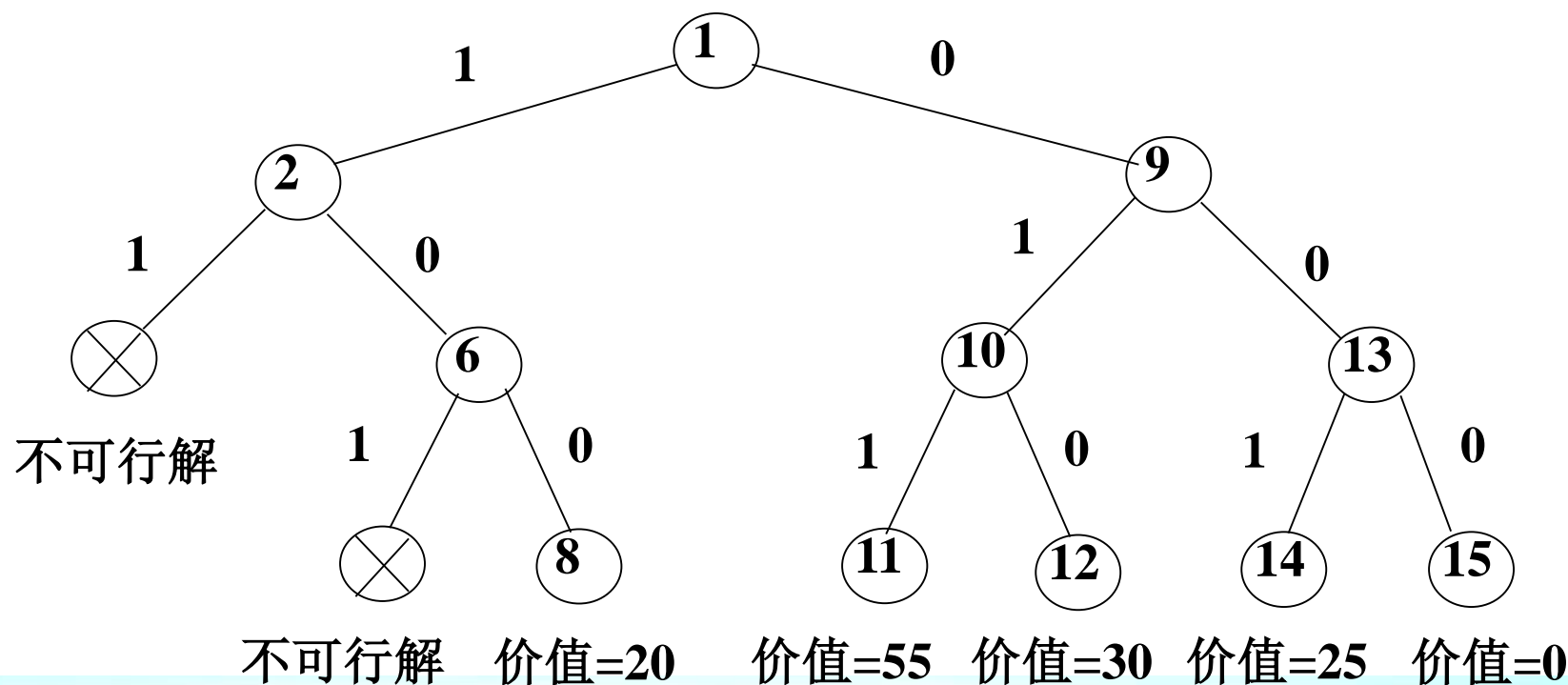
判断部分解是否满足约束条件，是否超出评估函数的界，如果不包含，则跳过对以该结点为根的子树的搜索，即所谓剪枝（Pruning）；

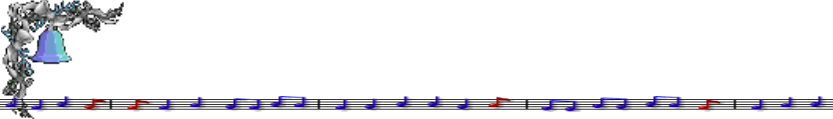
否则，进入以该结点为根的子树，继续按照深度优先策略搜索。





例如，对于 $n=3$ 的0/1背包问题，三个物品的重量为 $\{20, 15, 10\}$ ，价值为 $\{20, 30, 25\}$ ，背包容量为25，从图2所示的解空间树的根结点开始搜索，搜索过程如下：






再如，对于 $n=4$ 的TSP问题，其代价矩阵如图5所示，

$$C = \begin{pmatrix} \infty & 3 & 6 & 7 \\ 12 & \infty & 2 & 8 \\ 8 & 6 & \infty & 2 \\ 3 & 7 & 6 & \infty \end{pmatrix}$$

图5 TSP问题的代价矩阵





$$C = \begin{bmatrix} \infty & \mathbf{3} & \mathbf{6} & \mathbf{7} \\ \mathbf{12} & \infty & \mathbf{2} & \mathbf{8} \\ \mathbf{8} & \mathbf{6} & \infty & \mathbf{2} \\ \mathbf{3} & \mathbf{7} & \mathbf{6} & \infty \end{bmatrix}$$

TSP问题的代价矩阵

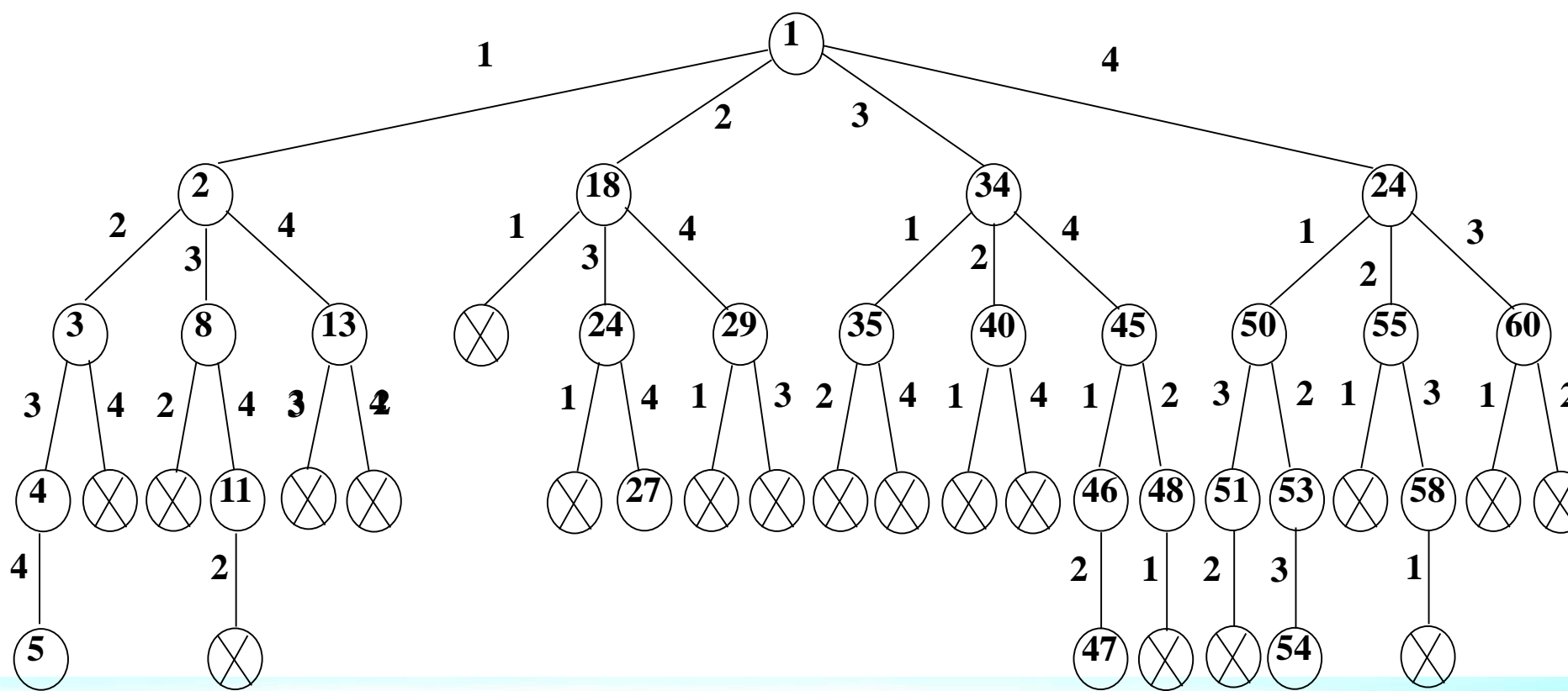


图6 TSP问题的搜索空间



➤在搜索过程中，通常采用两种策略避免无效搜索：

(1) 用约束条件剪去得不到可行解的子树；

(2) 用评估函数剪去得不到最优解的子树。

这两类函数统称为剪枝函数 (Pruning Function)

❖ 需要注意的是，问题的解空间树是虚拟的，并不需要在算法运行时构造一棵真正的树结构，只需要存储从根结点到当前结点的路径。





组合问题中的回溯法

八皇后问题

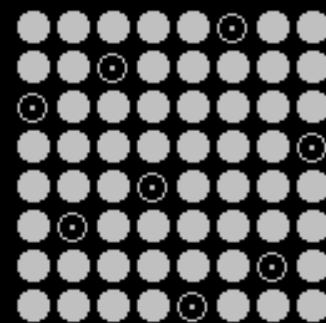


八皇后问题

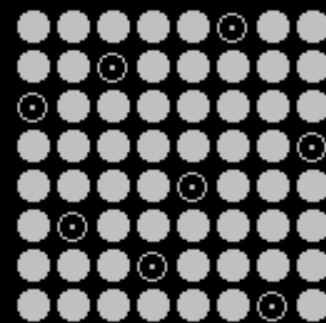
八皇后问题是十九世纪著名的数学家高斯于1850年提出的。

在 8×8 的棋盘上摆放八个皇后，使其不能互相攻击，即：

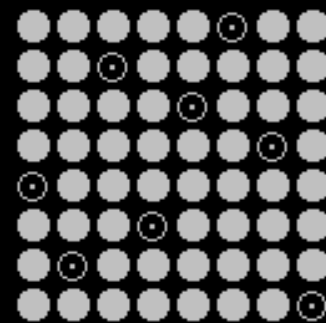
任意两个皇后都不能处于同一行、同一列或同一斜线上。



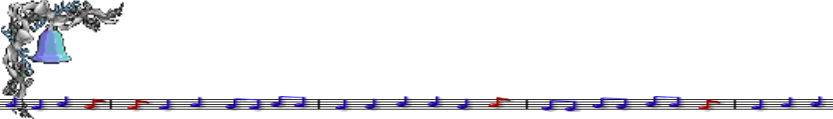
对应的数据为： 63184275



对应的数据为： 63185247



对应的数据为： 63571428



棋盘的每一行上可以而且必须摆放一个皇后， n 皇后问题的可能解用一个 n 元向量 $X=(x_1, x_2, \dots, x_n)$ 表示，其中， $1 \leq i \leq n$ 并且 $1 \leq x_i \leq n$ ，即第 i 个皇后放在第 i 行第 x_i 列上。

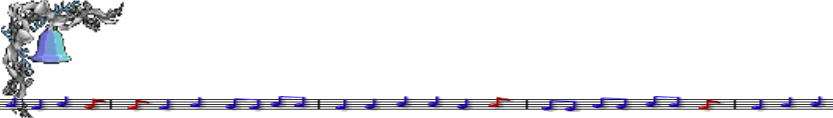
由于两个皇后不能位于同一列上，解向量 X 约束条件：

$$x_i \neq x_j \quad (\text{式1})$$

由于两个皇后不能位于同一斜线上，所以，解向量 X 必须满足约束条件：

$$|i - x_i| \neq |j - x_j| \quad (\text{式2})$$





为了简化问题，下面讨论四皇后问题。
四皇后问题的解空间树是一个完全4叉树，

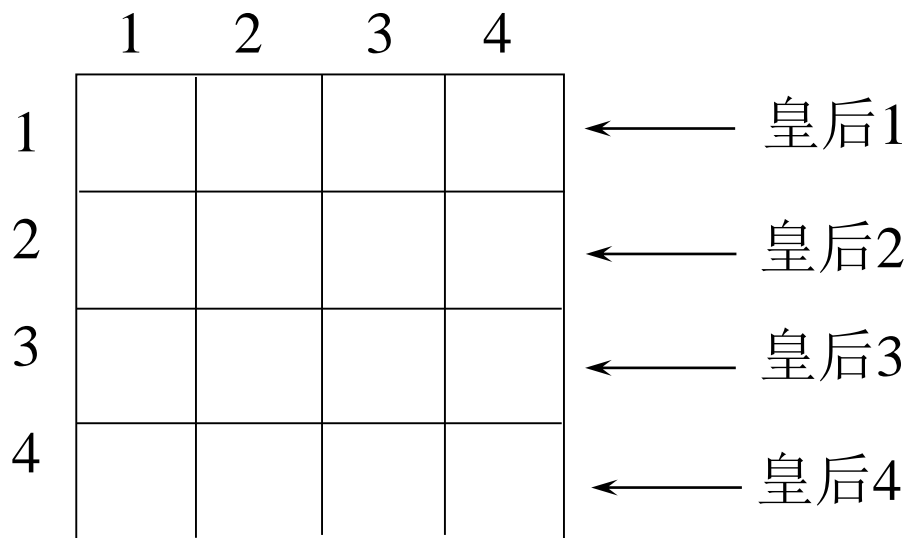
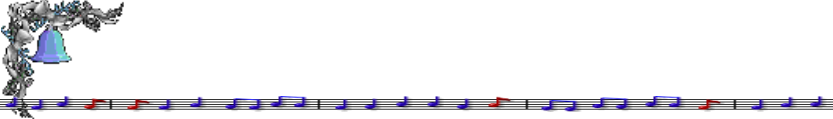


图11 四皇后问题





回溯法求解4皇后问题的搜索过程

Q			

(a)

Q			
×	×	Q	

(b)

Q			
×	×	Q	
×	×	×	×

(c)

Q			
			Q

(d)

Q			
			Q
×	Q		

(e)

Q			
			Q
×	Q		
×	×	×	×

(f)

	Q		

(g)

	Q		
×	×	×	Q

(h)

	Q		
			Q
Q			

(i)

	Q		
			Q
Q			
×	×	Q	

(j)





算法——n皇后问题

C++描述

```
void Queue(int n)
{
    for (i=1; i<=n; i++) //初始化
        x[i]=0;
    k=1;
    while (k>=1)
    {
        x[k]=x[k]+1; //在下一列放置第k个皇后
        while (x[k]<=n && !Place(k))
            x[k]=x[k]+1; //搜索下一列
        if (x[k]<=n && k==n) { //得到一个解，输出
            for (i=1; i<=n; i++)
                cout<<x[i];
            return;
        }
    }
}
```





```
else if (x[k]<=n && k<n)
    k=k+1;    //放置下一个皇后
else {
    x[k]=0;    //重置x[k], 回溯
    k=k-1;
}
}
```

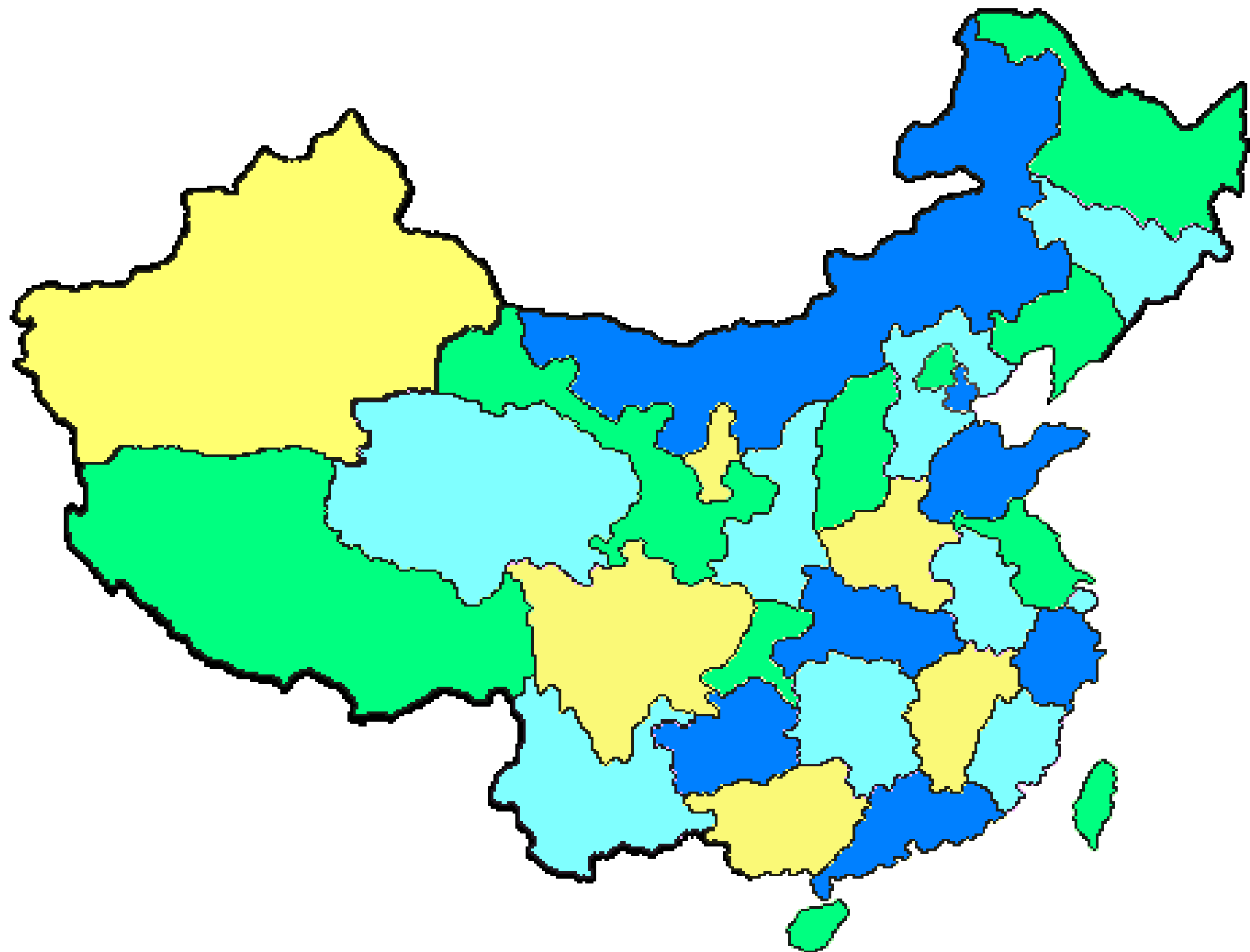
```
bool Place(int k) //考察皇后k放置在x[k]列是否发生冲突
{
    for (i=1; i<k; i++)
        if (x[k]==x[i] || abs(k-i)==abs(x[k]-x[i]))
            return false;
    return true;
}
```





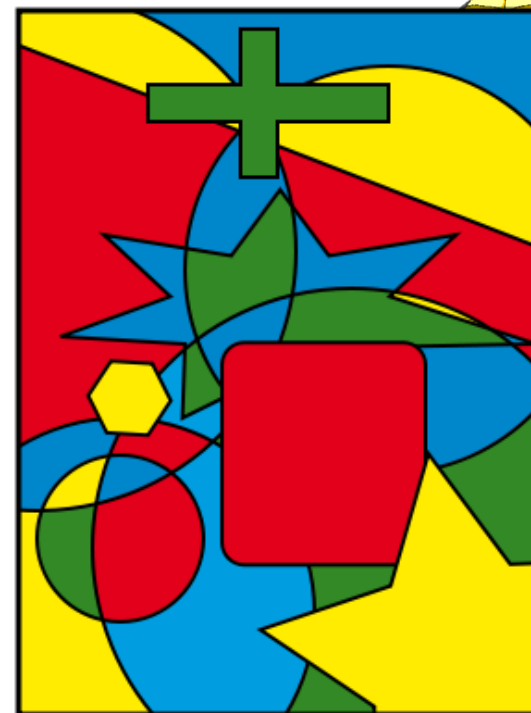
图着色问题







地图着色问题又称为“四色问题”，四色问题的内容是：
“任何连通的平面图，可以用不多于4种颜色对每一个区域着色，使相邻区域着不同颜色。”



在1852年，一个英国青年名叫盖思里（Francis Guthrie）提出的。

1878~1880年两年间，数学家肯普和泰勒两人分别宣布证明了四色定理。后来，人们发现他们实际上证明了一个较弱的命题——[五色定理](#)。就是说对平面图着色，用五种颜色就够了。

1976年美国伊利诺州立大学的两位教授，阿佩尔（Kenneth Appel）和海肯（Wolfgang Haken）宣布，用计算机证明了这个问题。他们的证明需要对1900多种情况进行详尽的分析，在一台高速计算机上耗时超过1200小时。对数学家来说总还是希望能找到数学方法的证明。

四色问题又称四色猜想，是世界近代三大数学难题之一。





- 物资存储问题:

设有 n 种物资要存放在仓库里，但有的物资不能放在同一房间里，否则引起损坏，甚至会发生危险。问存放这 n 种物资最少需要几个房间？





会场安排问题

- 问题描述:

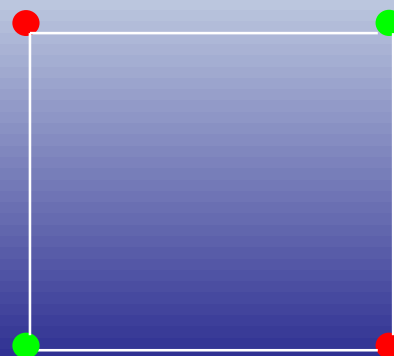
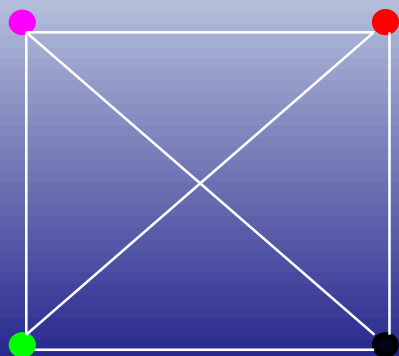
假设要在足够多的会场里安排一批活动，并希望使用尽可能少的会场。





图着色问题

图着色问题描述为：给定无向连通图 $G=(V, E)$ 和正整数 m ，求最小的整数 m ，使得用 m 种颜色对 G 中的顶点着色，使得任意两个相邻顶点着色不同。

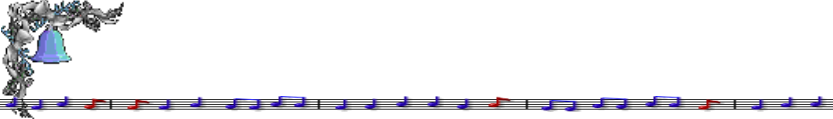




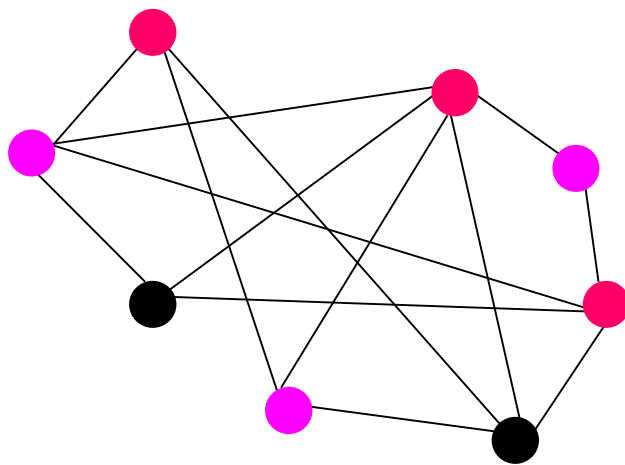
- 物资存储问题:

设有 n 种物资要存放在仓库里，但有的物资不能放在同一房间里，否则引起损坏，甚至会发生危险。问存放这 n 种物资最少需要几个房间？





- 用顶点 v_1, v_2, \dots, v_n 表示 n 种物资，若其中两种不能放在同一房间里，则 v_i, v_j 间连一条边。



顶点色数即为最少房间数





会场安排问题

- 问题描述:

假设要在足够多的会场里安排一批活动，并希望使用尽可能少的会场。

- 这个问题实际上是著名的图着色问题。若将每一个活动作为图的一个顶点，不相容活动间用边相连。使相邻顶点着有不同颜色的最小着色数，相应于要找的最小会场数。





交通信号灯问题

■ 1.1.1 问题

- (1) 每一车辆通过路口时都不会与其它车辆发生冲突；
- (2) 车辆在路口等待通过的时间尽可能少。

■ 1.1.2 实例

ab , ac , ad
ba , bc , bd
da , db , dc
ea , eb , ec , ed

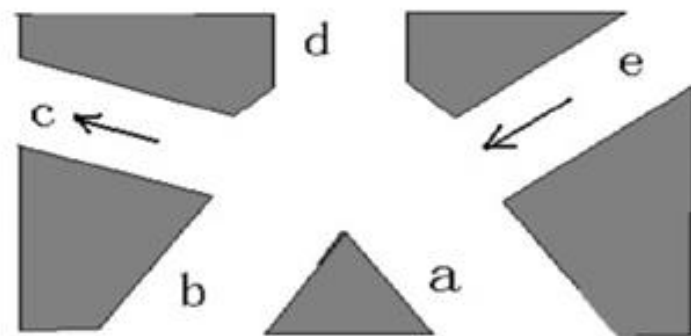


Fig. 1.1 一个五叉路口



交通信号灯问题

- Fig.1.2给出的无向图对应于五叉路口实例，该图有13个结点和20条边，其中ba, dc, ed三个顶点是孤立点，说明这三条路线与其它所有路线不相交，就是所谓的“拐小弯”

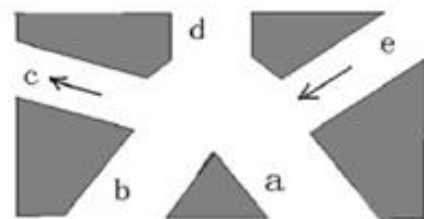
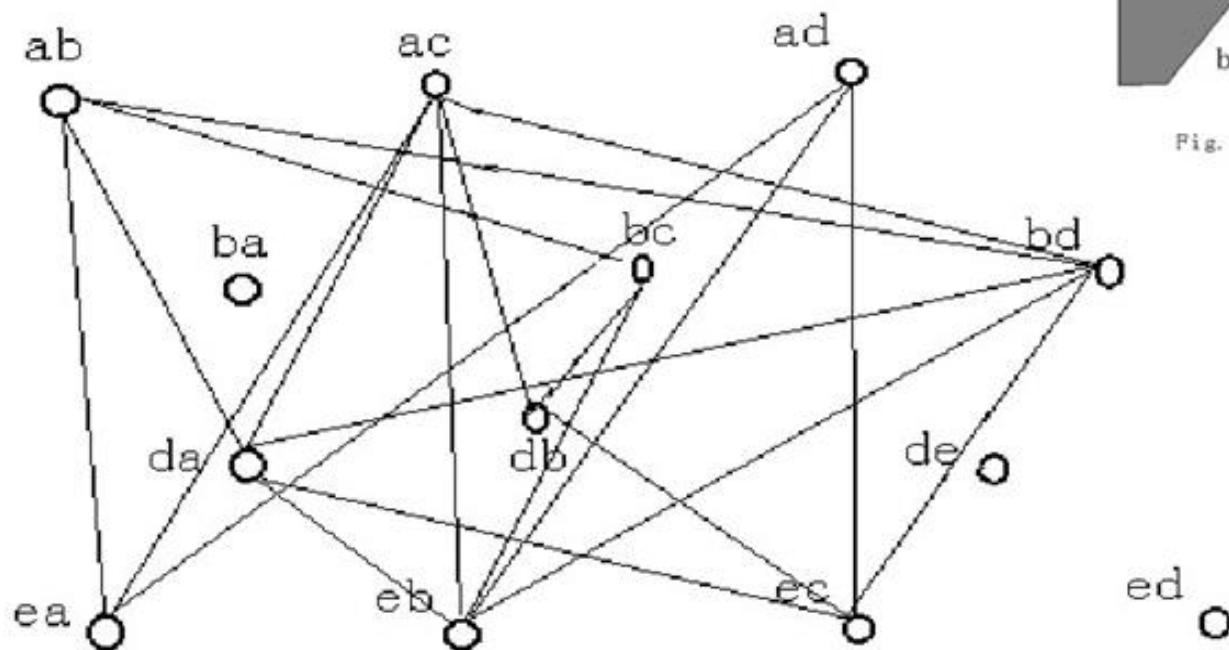


Fig. 1.1 一个五叉路口

图着色问题

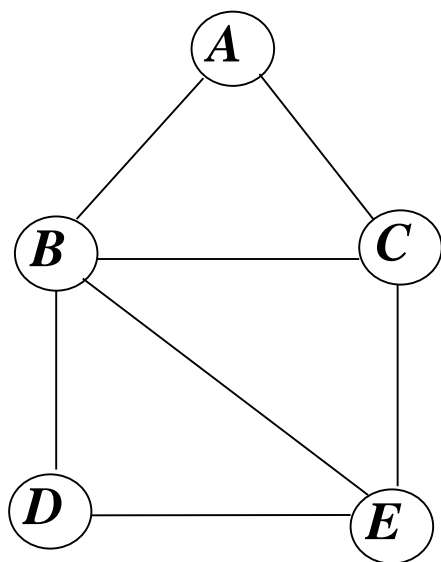
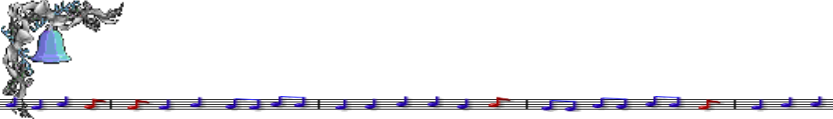


用 m 种颜色为无向图 $G=(V, E)$ 着色，其中， V 的顶点个数为 n ，可以用一个 n 元组 $C=(c_1, c_2, \dots, c_n)$ 来描述图的一种可能着色，其中， $c_i \in \{1, 2, \dots, m\}$ ($1 \leq i \leq n$)表示赋予顶点 i 的颜色。

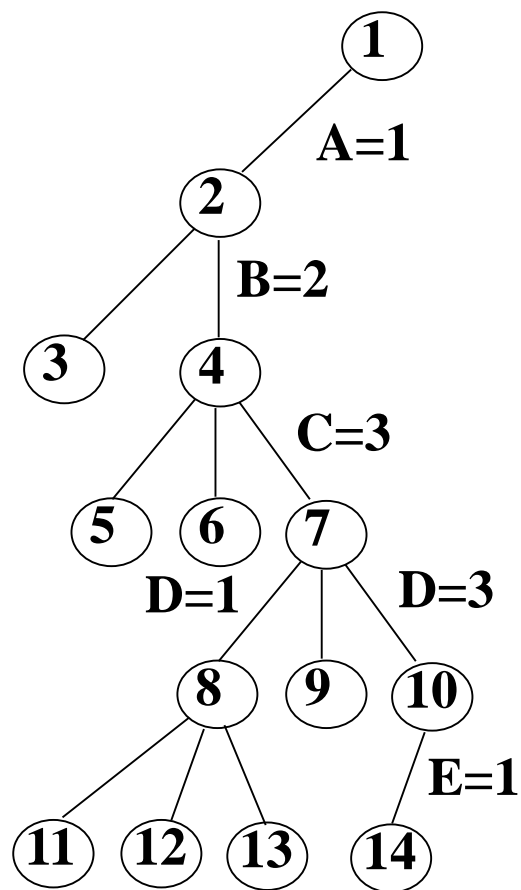
例如，5元组 $(1, 2, 2, 3, 1)$ 表示对具有5个顶点的无向图的一种着色。

如果在 n 元组 C 中，所有相邻顶点都不会着相同颜色，就称此 n 元组为可行解，否则为无效解。





(a) 一个无向图



(b) 回溯法搜索空间

图8 回溯法求解图着色问题示例





设数组 $\text{color}[n]$ 表示顶点的着色情况，回溯法求解 m 着色问题的算法如下：

算法1——图着色问题

伪代码

1. 将数组 $\text{color}[n]$ 初始化为0;
2. $k=1$;
3. while ($k \geq 1$)
 - 3.1 依次考察每一种颜色，若顶点 k 的着色与其他顶点的着色不发生冲突，则转步骤3.2；否则，搜索下一个颜色；
 - 3.2 若顶点已全部着色，则输出数组 $\text{color}[n]$ ，返回；
 - 3.3 否则，
 - 3.3.1 若顶点 k 是一个合法着色，则 $k=k+1$ ，转步骤3处理下一个顶点；
 - 3.3.2 否则，重置顶点 k 的着色情况， $k=k-1$ ，转步骤3回溯；

算法2——图着色问题

C++描述

```
void GraphColor(int n, int c[ ][ ], int m)
//所有数组下标从1开始
{
    for (i=1; i<=n; i++ )    //将数组color[n]初始化为0
        color[i]=0;
    k=1;
    while (k>=1)
    {
        color[k]=color[k]+1;
        while (color[k]<=m)
            if Ok(k) break;
        else color[k]=color[k]+1; //搜索下一个颜色
        if (color[k]<=m && k==n) //求解完毕，输出解
        {
            for (i=1; i<=n; i++)
                cout<<color[i];
            return;
        }
    }
}
```



```
else if (color[k]<=m && k<n)
    k=k+1;        //处理下一个顶点
else {
    color[k]=0;
    k=k-1;  //回溯
}
}
}
```

```
bool Ok(int k) //判断顶点k的着色是否发生冲突
{
    for (i=1; i<k; i++)
        if (c[k][i]==1 && color[i]==color[k])
            return false;
    return true;
}
```





思考：五个颜色各异的风房子里住着5个不同国籍的人，他们所养的宠物、



喜欢的饮料、拥有的汽车各不相同

- （1）英国人住在红房子里；
- （2）西班牙人养狗；
- （3）居住在绿房子里的人喜欢喝可乐；
- （4）乌克兰人喜欢喝蛋酒；
- （5）住在绿房子里的人是住在象牙色房子里人的右邻；
- （6）拥有老爷车的人养蜗牛；
- （7）拥有福特车的人住在黄房子里；
- （8）住在中间房子里的人喝牛奶；
- （9）挪威人住在最左边房子里；
- （10）拥有雪佛莱汽车的人与养狐狸的人是邻居；
- （11）拥有福特汽车的人与养马的人是邻居；
- （12）拥有奔驰汽车的人爱喝橙汁；
- （13）日本人开大众；
- （14）挪威人的邻居住在三房子里

— 问：斑马属于谁？ 谁爱喝矿泉水？





思考题

1. 亚瑟王打算请150名骑士参加宴会，但是有些骑士相互间会有口角，而亚瑟王知道谁与谁不和。亚瑟王希望能让他的客人围着一张圆桌坐下，而所有不和的骑士都不会挨着坐。请设计回溯算法求解亚瑟王问题。

