算法分析与设计

Analysis and Design of Algorithm

Lesson 13

要点回顾

- 回溯法
  - 一种组织得井井有条的(带有系统性)、能避免不必要搜索的(带有跳跃性)穷举式搜索法
- 回溯法的几个简单实例

问题	解性质	解向量	搜索空间	搜索方式	约束条件
n后问题	可行解	$\langle x_1, x_2, \ldots, x_n \rangle$ x_i:第i行列号	n叉树	深度优先	彼此 不攻击
0-1背包问题	最优解	$\langle x_1, x_2, \ldots, x_n \rangle$ x_i=0/1 x_i=1⇔选i	子集树	深度优先	不超过 背包重量
TSP问题	最优解	$\langle k_1, k_2, \ldots, k_n \rangle$ 1,2,…,n的排列	排列树	深度优先	选未经过 的城市
特点	搜索解	扩张 部分向量	树	跳跃式 遍历	约束条件 回溯判定

要点回顾(cont.)

- 回溯法设计要素
  - 适用对象：求解搜索问题和优化问题
  - 搜索空间：树，结点对应部分解向量，可行解在树叶上
  - 搜索过程：采用系统的方法隐含遍历搜索树
  - 搜索策略：深度/宽度优先、函数优先、宽深结合
  - 结点分支判定条件：
    - 满足约束条件——分支扩张解向量
    - 不满足约束条件——回溯到该结点的父结点
  - 结点状态：动态生成
    - 可以约定(黑：已访问；灰：正在访问；白：未访问)
  - 存储：当前路径

回溯法适用条件

- 在结点$\langle x_1, x_2, \ldots, x_k \rangle$处

  $P(x_1, x_2, \ldots, x_k)$为真

  $\Leftrightarrow$向量$\langle x_1, x_2, \ldots, x_k \rangle$满足某个性质（约束条件）

  （例如：n后中k个皇后放在彼此不攻击的位置）

- 多米诺性质

  $$P(x_1, x_2, \ldots, x_{k+1}) \rightarrow P(x_1, x_2, \ldots, x_k) \quad 0 < k < n$$

- 逆否命题：

  $$\neg P(x_1, x_2, \ldots, x_k) \rightarrow \neg P(x_1, x_2, \ldots, x_{k+1}) \quad 0 < k < n$$

  作用：k维向量不满足约束条件，扩张向量到k+1维仍旧不满足，可以回溯！

一个实例

例： 求不等式的整数解

$$5x_1+4x_2-x_3 \leqslant 10,\ 1 \leqslant x_k \leqslant 3,\ k=1,2,3$$

$P(x_1, x_2, ..., x_k)$：将$x_1, x_2, ..., x_k$代入原不等式的相应部分，部分算术值小于等于10

➔不满足多米诺性质：

$$5x_1+4x_2-x_3 \leqslant 10 \not\Rightarrow 5x_1+4x_2 \leqslant 10$$

变换使得问题满足多米诺性质：

令 $x_3=3-x_3^*$，那么原不等式变换为

$$5x_1+4x_2+x_3^* \leqslant 13$$

$$1 \leqslant x_1, x_2 \leqslant 3,\ 0 \leqslant x_3^* \leqslant 2$$

归纳一下

- 回溯法适用条件：多米诺性质
- 回溯法设计步骤
  1. 定义解向量和每个分量的取值范围
     - 解向量为$\langle x_1, x_2, ..., x_n \rangle$
     - 确定x_i的取值集合为X_i (显约束)
  2. 由$\langle x_1, x_2, ..., x_{k-1} \rangle$确定如何计算$x_k$取值集合$S_k$, $S_k \subseteq X_k$
  3. 确定结点儿子的排列规则
  4. 判断是否满足多米诺性质
  5. 确定每个结点分支的约束条件
  6. 确定搜索策略：深度优先/宽度优先，...
  7. 确定存储搜索路径的数据结构

回溯法的实现及实例

回溯法两种实现

- 递归实现
- 算法ReBack(k)

1. if $k>n$ then $\langle x_1, x_2, \ldots, x_n \rangle$是解
2. else while $S_k \neq \emptyset$ do
3. $x_k \leftarrow S_k$中最值
4. $S_k \leftarrow S_k - \{x_k\}$
5. 计算S_{k+1}
6. ReBack(k+1)

- 算法ReBacktrack(n)
- 输入：n
- 输出：所有解

1. for $k \leftarrow 1$ to n 计算X_k且$S_k \leftarrow X_k$
2. ReBack(1)

回溯法两种实现

- 迭代实现
- 算法Backtrack(n)
- 输入： n
- 输出： 所有解

1. 对于i=1,2,...,n确定X_i
2. $k \leftarrow 1$
3. 计算S_k
4. **while** $S_k \neq \emptyset$ **do**
5. $x_k \leftarrow S_k$中最值；$S_k \leftarrow S_k - \{x_k\}$
6.   **if** $k<n$ **then**
7. $k \leftarrow k+1$; 计算S_k
8.   **else** $\langle x_1, x_2, \ldots, x_n \rangle$是解
9. **if** $k>1$ **then** $k \leftarrow k-1$; **goto** 4

确定初始取值

满足约束分支搜索

回溯

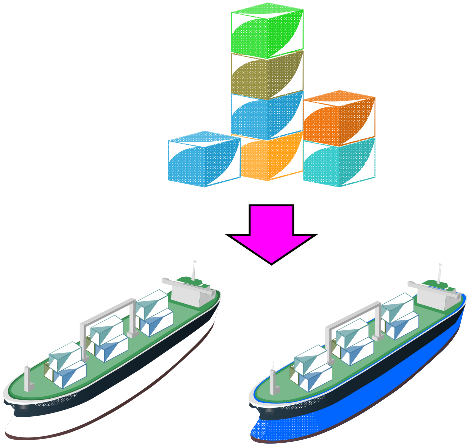
装载问题

问题： 有n个集装箱，需要装上两艘载重分别为c_1和c_2的轮船。w_i为第i个集装箱的重量，且$w_1+w_2+...+w_n \leqslant c_1+c_2$。问是否存在一种合理的装载方案把这$n$个集装箱装上船？如果有，给出一种方案。

实例：

W=⟨90, 80, 40, 30, 20, 12, 10⟩

c_1=152, c_2=130



解： 1,3,6,7装第1艘船，其余第2艘

求解思路

输入： W=⟨w_1, w_2, ..., w_n⟩为集装箱重量，c_1和c_2为船的最大载重

算法思想： 令第一艘船的载入量为W_1

1. 用回溯法求使得c_1-W_1达到最小的装载方案
2. 若满足

$$w_1+w_2+...+w_n-W_1 \leqslant c_2$$

则回答"YES"，否则回答"NO"

算法伪码

- 算法Loading(W,c_1)

1. Sort(W)
2. $B \leftarrow c_1$; $best \leftarrow c_1$; $i \leftarrow 1$
3. **while** $i \leqslant n$ **do**
4.   **if** 装入i后重量不超过c_1
5.   **then** $B \leftarrow B\text{-}w_i$; $x[i] \leftarrow 1$; $i \leftarrow i+1$
6.   **else** $x[i] \leftarrow 0$; $i \leftarrow i+1$
7. **if** $B<best$ **then** 记录解; $best \leftarrow B$
8. **Backtrack**(i) 回溯
9. **if** i=1 **then return** 最优解
10. **else goto** 3

*B*为当前空隙
*best*为最小空隙

子过程 Backtrack

- 算法**Backtrack**(*i*)

```
while i > 1 and x[i]=0 do
    i←i-1
if x[i]=1
then x[i]←0
    B←B+w_i
    i←i+1
```

沿右分支一直回溯发现左分支边或到根为止

实例

W=⟨90, 80, 40, 30, 20, 12, 10⟩

c_1=152, c_2=130

解：

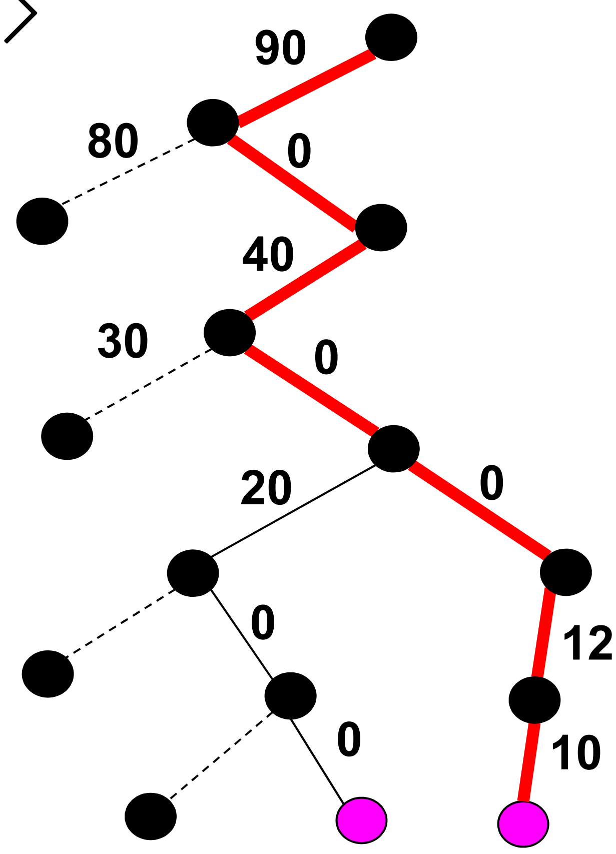
可以装！

方案如下：

1,3,6,7装第一艘船

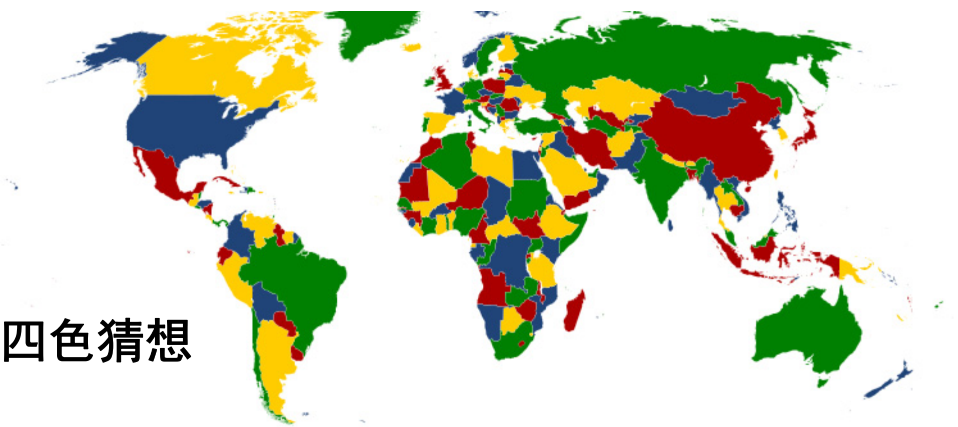
2,4,5装第二艘船

时间复杂度：O(2^n)

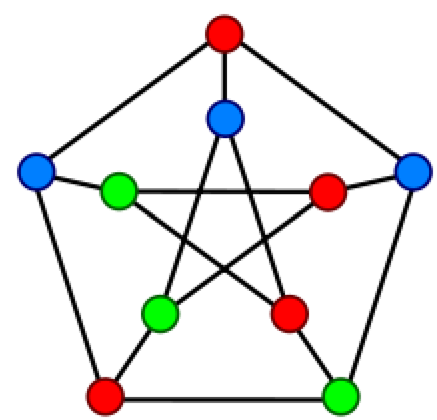


图的着色问题

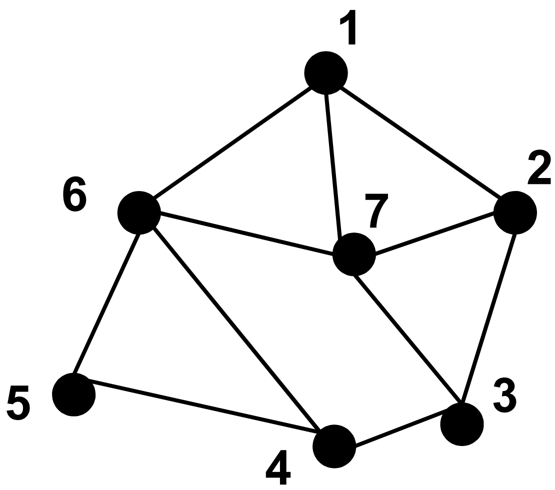
- **输入：** 无向连通图G和m种颜色的集合，用这些颜色给图的顶点着色，每个顶点一种颜色。要求是：G的每条边的两个顶点着不同颜色。
- **输出：** 所有可能的着色方案。如果不存在着色方案，回答“NO”

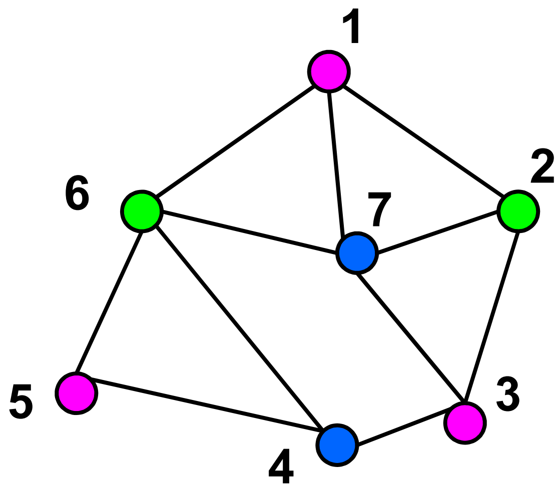


四色猜想



实例





n=7, m=3

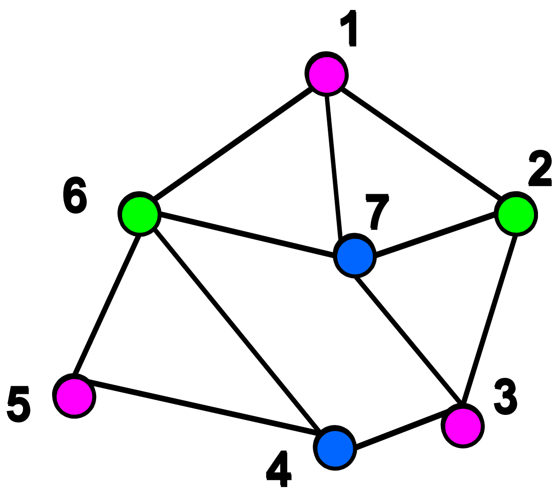
解向量

- 设G=(V, E), V={1,2,...,n}
- 颜色编号：1,2, ... , m
- **解向量：** $\langle x_1, x_2, ..., x_n\rangle$
  - $x_i \in \{1,2,...,m\}$
- 结点的部分向量$\langle x_1, x_2, ..., x_k\rangle$, $1 \leqslant k \leqslant n$
  - 表示只给顶点1,2,...,k着色的部分方案

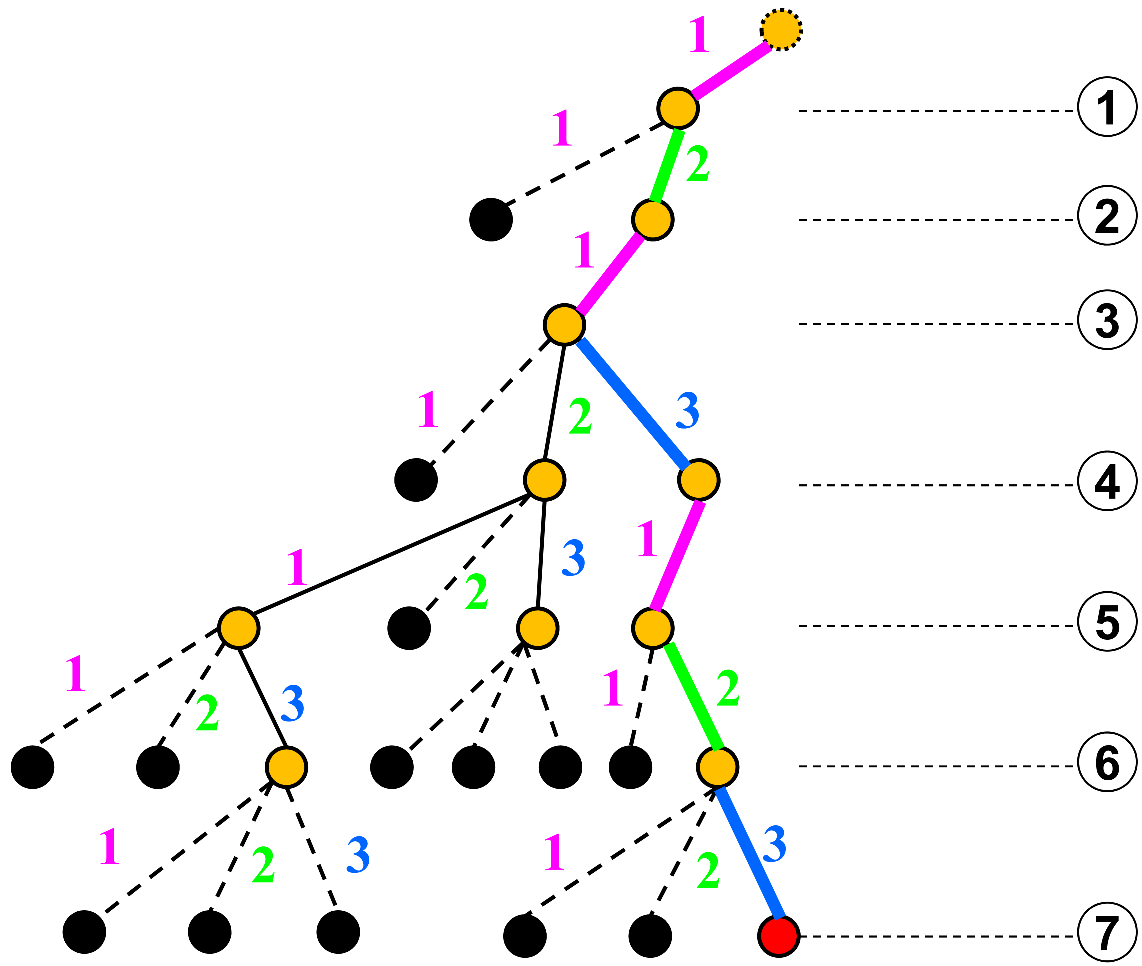
算法设计

- 搜索树：m叉树
- 约束条件：在结点$\langle x_1,\ldots, x_k\rangle$处，顶点$k+1$的邻接表中结点已用过的颜色不能再用。如果邻接表种结点已用过m种颜色，则结点$k+1$没法着色，从该结点回溯到其父节点。➔满足多米诺性质
- 搜索策略：深度优先
- 时间复杂度：$O(n \times m^n)$

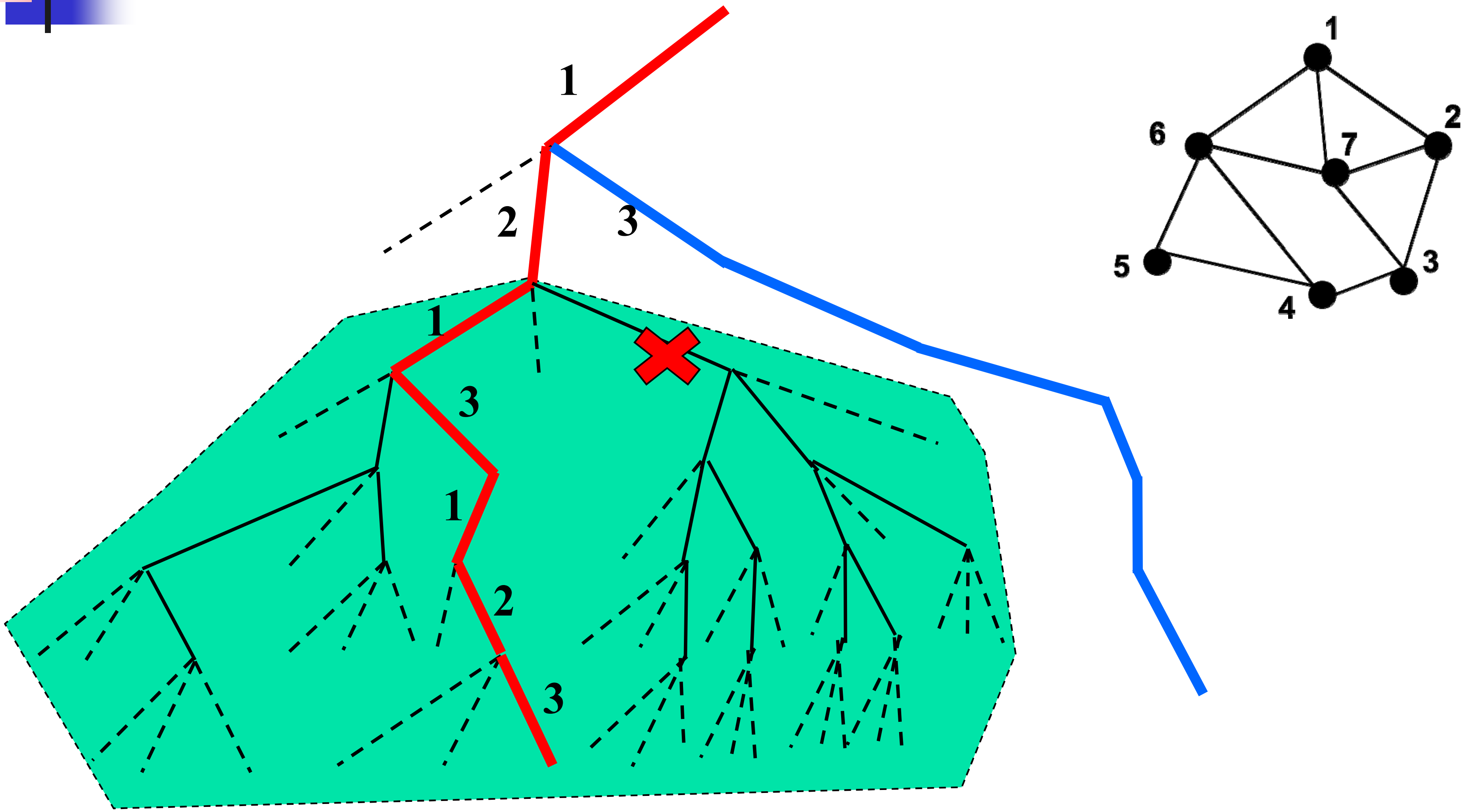
运行实例



n=7, m={1、2、3}



搜索树



第一个解向量：<1, 2, 1, 3, 1, 2, 3>

时间复杂度与改进途径

- 时间复杂度：$O(n \times m^n)$
- 根据对称性，只需搜索1/3的解空间。当1和2确定，即<1,2>以后，只有1个解，在<1,3>为根的子树中，也只有一个解。由于3个子树的对称性，总共有6个解。
- 在取定<1,2>后，不可扩张成<1,2,3>，因为7和1,2,3都相邻。7没法着色。可以从打叉的结点回溯，而不必搜索其子树。

着色问题的应用

- **会场分配问题：**
  - 有n项活动需要安排，对于活动i, j，如果i, j时间冲突，就说i, j不相容。如何分配这些活动，使得每个会场的活动相容且占用会场数最少？
- **建模：**
  - 活动作为图的顶点，如果i, j不相容，则在i和j之间加一条边，会场标号作为颜色标号。求图的一种着色方案，使得使用的颜色数最少。

第五章小结

- 理解回溯法的深度优先搜索策略
- 掌握用回溯法解题的算法框架
  - 递归回溯最优子结构性质
  - 迭代回溯贪心选择性质
  - 子集树算法框架
  - 排列树算法框架
- 通过应用范例学习回溯法的设计策略
  - n后问题、0-1背包问题 、旅行售货员问题
  - 装载问题
  - 图的着色问题

课程内容

NP完全性理论与近似算法				算法高级理论
随机化算法	线性规划与网络流			高级算法
递归分治	动态规划	贪心算法	回溯与分支限界	基础算法
算法分析与问题的计算复杂性				算法基础理论

第六章 分支限界法

学习要点

- 理解分支限界法的剪枝搜索策略
- 掌握分支限界法的算法框架
  - 队列式分支限界法
  - 优先队列式分支限界法
- 通过应用范例学习分支限界法的设计策略
  - 背包问题
  - 0-1背包问题
  - 非对称旅行商问题
  - 单源最短路径问题
  - 装载问题
  - 最大团问题

分支限界法概述

分支限界法与回溯法的区别

- **求解目标**：
  - 一般情况下，回溯法的求解目标是找出解空间树中满足约束条件的所有解，而分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出在某种意义下的最优解（可以理解为更细粒度的回溯法）。
- **搜索策略**：
  - 回溯法更多地以深度优先的方式搜索解空间树，而分支限界法则更多地以宽度优先或以最小耗费优先（函数优先）的方式搜索解空间树。
- **各自特点**：
  - 回溯法空间效率高；分支限界法往往更快。

分支限界法的基本思想

- 分支限界法常以宽度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树。
- 对已处理的各结点根据限界函数估算目标函数的可能取值，从中选取使目标函数取得极值（极大/极小）的结点优先进行宽度优先搜索➔不断调整搜索方向，尽快找到解。
- 特点：限界函数常基于问题的目标函数，适用于求解最优化问题。

常见的分支限界法

- 队列式分支限界法
  - 按照队列先进先出原则选取下一个结点为扩展结点。
- 优先队列式分支限界法
  - 按照规定的结点费用最小原则选取下一个结点为扩展结点（常采用优先队列实现）。
- 栈式分支限界法
  - 按照栈后进先出原则选取下一个结点为扩展结点。

组合优化问题的分支限界法

组合优化问题

- 组合优化问题的相关概念
  - 目标函数（极大化或极小化）
  - 约束条件（解满足的条件）
  - 可行解：搜索空间满足约束条件的解
  - 最优解：使得目标函数达到极大（极小）的可行解

- 一般背包问题

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$

$$s.t. \quad \begin{cases} 2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10 \\ x_i \in \mathbb{N}, i = 1,2,3,4 \end{cases}$$

代价函数

- 计算位置：搜索树的结点
- 估值：极大化问题是以该点为根的子树所有可行解的值的上界（极小化问题则为下界）
- 性质：对极大化问题父节点代价不小于子结点的代价（极小化问题则相反）