

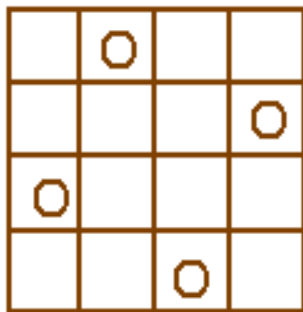
几个回溯算法 的例子

4后问题

4后问题：在 4×4 的方格棋盘上放置4个皇后，使得没有两个皇后在同一行、同一列、也不在同一条45度的斜线上。问有多少种可能的布局？

解是 4 维向量 $\langle x_1, x_2, x_3, x_4 \rangle$

解： $\langle 2, 4, 1, 3 \rangle$, $\langle 3, 1, 4, 2 \rangle$

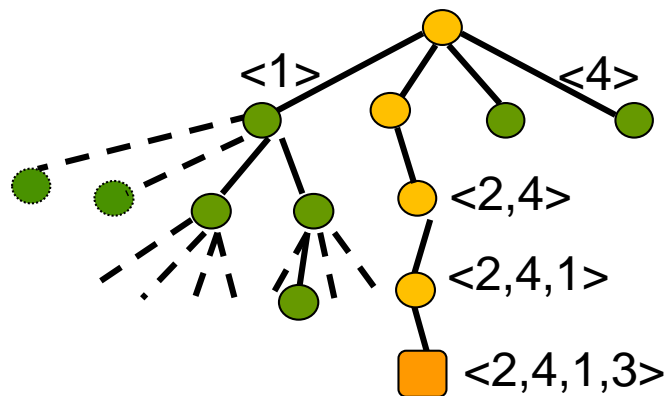


推广到8后问题

解：8维向量，有 92个。

例如： $\langle 1, 5, 8, 6, 3, 7, 2, 4 \rangle$ 是解。

搜索空间：4叉树



每个结点有4个儿子，分别代表选择
1,2,3,4列位置

第 i 层选择解向量中第 i 个分量的值

最深层的树叶是解

按深度优先次序遍历树，找到所有解

0-1背包问题

问题:

有 n 种物品，每种物品只有 1 个. 第 i 种物品价值为 v_i , 重量为 w_i , $i=1,2,\dots,n$.
问如何选择放入背包的物品，使得总重量不超过 B , 而价值达到最大?

实例:

$V=\{12,11,9,8\}$, $W=\{8,6,4,3\}$, $B=13$

最优解:

$\langle 0,1,1,1 \rangle$, 价值: 28, 重量: 13

算法设计

解： n 维0-1向量 $\langle x_1, x_2, \dots, x_n \rangle$,
 $x_i=1 \Leftrightarrow$ 物品 i 选入背包

结点： $\langle x_1, x_2, \dots, x_k \rangle$ （部分向量）

搜索空间： 0-1取值的二叉树，称为子集树，有 2^n 片树叶.

可行解：满足约束条件 $\sum_{i=1}^n w_i x_i \leq B$ 的解

最优解：可行解中价值达到最大的解

实例

输入:

$V=\{12,11,9,8\}$, $W=\{8,6,4,3\}$, $B=13$

2个可行解:

$\langle 0,1,1,1 \rangle$, 选入物品2,3,4, 价值为28,
重量为13

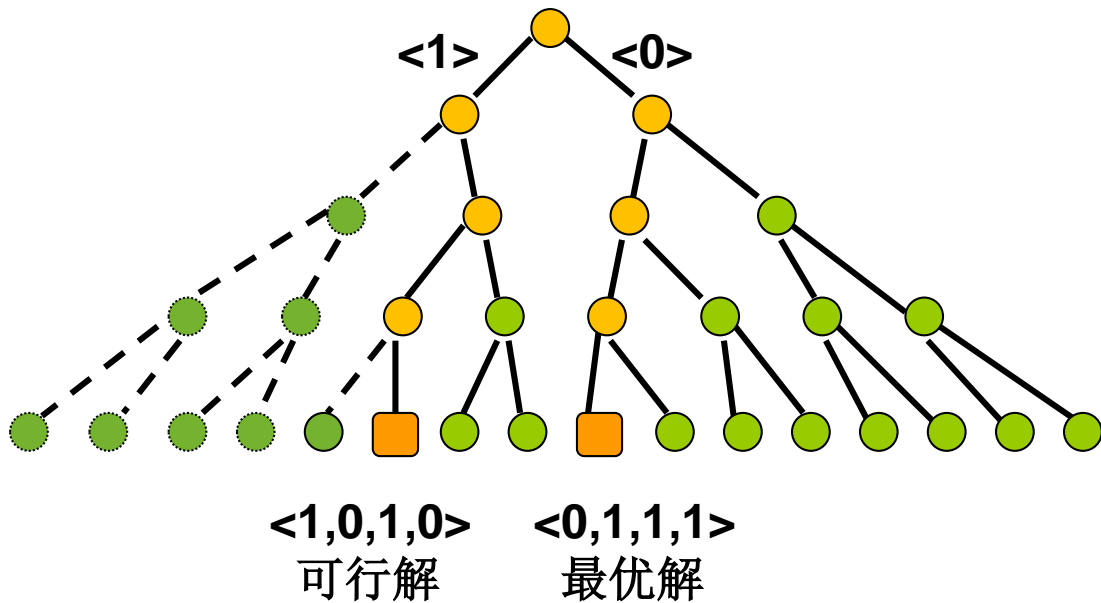
$\langle 1,0,1,0 \rangle$, 选入物品1,3, 价值为21,
重量为12

最优解: $\langle 0,1,1,1 \rangle$

搜索空间

实例: $V=\{12,11,9,8\}$, $W=\{8,6,4,3\}$, $B=13$

搜索空间: 子集树, 2^n 片树叶



货郎问题

问题：有 n 个城市，已知任两个城市之间的距离，求一条每个城市恰好经过一次的回路，使得总长度最小。

建模：城市集 $C=\{c_1, c_2, \dots, c_n\}$ ，距离

$$d(c_i, c_j) = d(c_j, c_i) \in \mathbb{Z}^+, 1 \leq i < j \leq n$$

求： $1, 2, \dots, n$ 的排列 k_1, k_2, \dots, k_n 使得

$$\min\left\{\sum_{i=1}^{n-1} d(c_{k_i}, c_{k_{i+1}}) + d(c_{k_n}, c_{k_1})\right\}$$

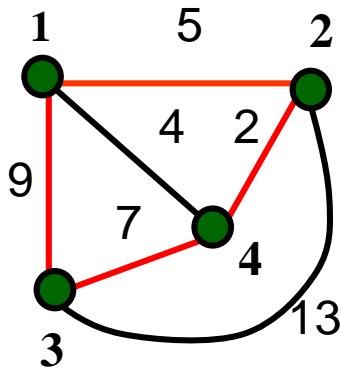
实例

$$C = \{1, 2, 3, 4\}$$

$$d(1, 2) = 5, d(1, 3) = 9,$$

$$d(1, 4) = 4, d(2, 3) = 13,$$

$$d(2, 4) = 2, d(3, 4) = 7$$

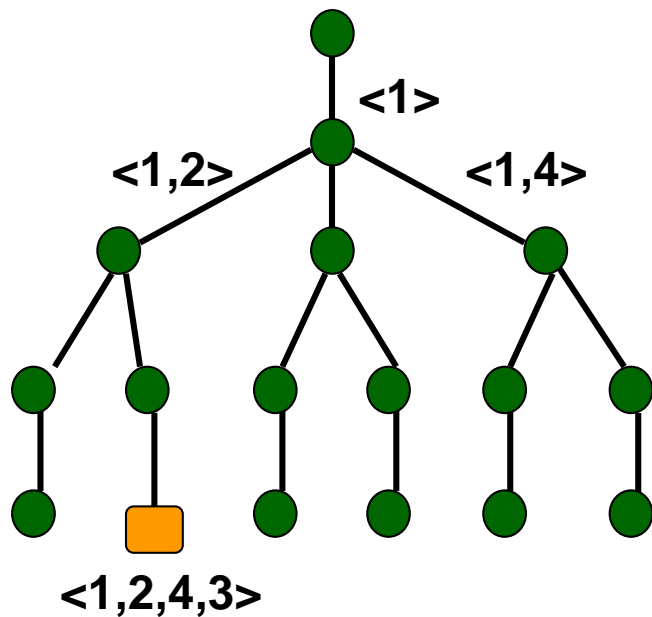


解: $\langle 1, 2, 4, 3 \rangle$,

$$\text{长度} = 5 + 2 + 7 + 9 = 23$$

搜索空间

排列树, 有 $(n-1)!$ 片树叶



小结

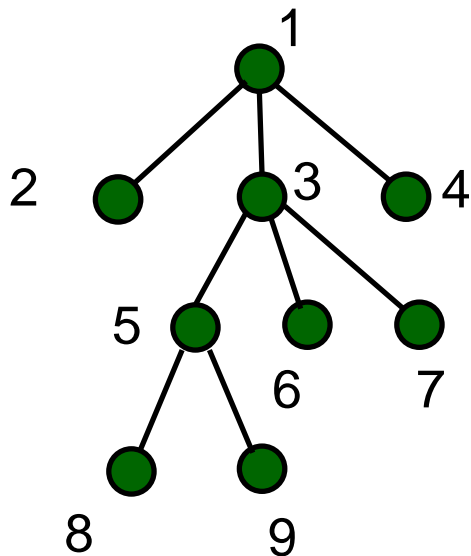
- 回溯算法的例子： n 后问题，0-1背包问题，货郎问题
- 解：向量
- 搜索空间：树，可能是 n 叉树、子集树、排列树等等，树的结点对应于部分向量，可行解在叶结点
- 搜索方法：深度优先，宽度优先，...
跳越式遍历搜索树，找到解

回溯算法的设计 思想和适用条件

问题分析

问题	解性质	解描述 向量	搜索 空间	搜索 方式	约束 条件
n 后	可行解	$\langle x_1, x_2, \dots, x_n \rangle$ x_i : 第 i 行列号	n 叉树	深度, 宽度优先	彼此不攻击
0-1背包	最优解	$\langle x_1, x_2, \dots, x_n \rangle$ $x_i = 0, 1$, $x_i = 1 \Leftrightarrow$ 选 i	子集树	深度, 宽度优先	不超背包重量
货郎	最优解	$\langle i_1=1, i_2, \dots, i_n \rangle$ $1, 2, \dots, n$ 的排列	排列树	深度, 宽度优先	选没有经过的城市
特点	搜索解	向量, 不断扩张部分向量	树	跳跃式遍历	约束条件回溯判定

深度与宽度优先搜索



深度优先访问顺序:

1→2→3→5→8→9
→6→7→4

宽度优先访问顺序:

1→2→3→4→5→6
→7→8→9

回溯算法基本思想

- (1) **适用**：求解搜索问题和优化问题.
- (2) **搜索空间**：树，结点对应部分解向量，可行解在树叶上.
- (3) **搜索过程**：采用系统的方法隐含遍历搜索树.
- (4) **搜索策略**：深度优先，宽度优先，函数优先，宽深结合等.

回溯算法基本思想(续)

(5) 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.

(6) 结点状态: 动态生成

白结点(尚未访问)

灰结点(正在访问该结点为根的子树)

黑结点(该结点为根的子树遍历完成)

(7) 存储: 当前路径

结点状态

深度优先

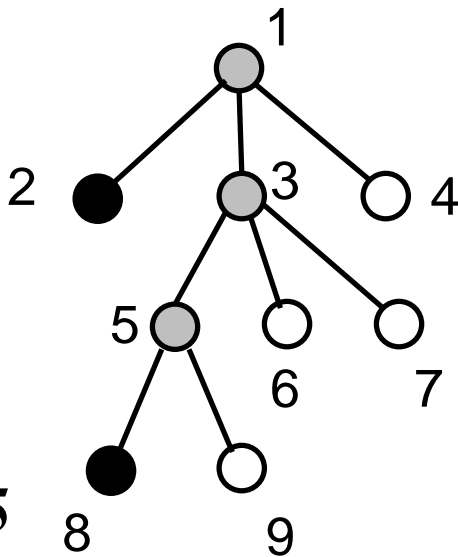
访问次序:

$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8$

已完成访问: 2, 8

已访问但未结束: 1, 3, 5

尚未访问: 9, 6, 7, 4



回溯算法的适用条件

在结点 $\langle x_1, x_2, \dots, x_k \rangle$ 处

$P(x_1, x_2, \dots, x_k)$ 为真

\Leftrightarrow 向量 $\langle x_1, x_2, \dots, x_k \rangle$ 满足某个性质
(n 后中 k 个皇后放在彼此不攻击的位置)

多米诺性质

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

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

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.

一个反例

例 求不等式的整数解

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

$P(x_1, \dots, x_k)$: 将 x_1, x_2, \dots, x_k 代入原不等式的相应部分, 部分和小于等于10

不满足多米诺性质:

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

变换使得问题满足多米诺性质:

令 $x_3 = 3 - x_3'$,

$$5x_1 + 4x_2 + x_3' \leq 13$$

$$1 \leq x_1, \quad x_2 \leq 3, \quad 0 \leq x_3' \leq 2$$

小结

- 回溯算法的适用条件：
多米诺性质
- 回溯算法的设计步骤
 - (1) 定义解向量和每个分量的取值范围
解向量 为 $\langle x_1, x_2, \dots, x_n \rangle$
确定 x_i 的取值集合为 $X_i, i = 1, 2, \dots, n$.

小结（续）

- (2) 在 $\langle x_1, x_2, \dots, x_{k-1} \rangle$ 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, 宽度优先等
- (7) 确定存储搜索路径的数据结构

回溯算法的 实现及实例

回溯算法递归实现

算法 **ReBack** (k)

1. if $k > n$ then $\langle x_1, x_2, \dots, x_n \rangle$ 是解
2. else while $S_k \neq \emptyset$ do
3. $x_k \leftarrow S_k$ 中最小值
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: 所有的解

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

迭代实现

迭代算法 Backtrack

输入: n

输出: 所有的解

确定初
始取值

1. 对于 $i=1, 2, \dots, n$ 确定 X_i

2. $k \leftarrow 1$

3. 计算 S_k

满足约束
分支搜索

4. while $S_k \neq \emptyset$ do

5. $x_k \leftarrow S_k$ 中最小值; $S_k \leftarrow S_k - \{x_k\}$

6. if $k < n$ then

7. $k \leftarrow k+1$; 计算 S_k

8. else $\langle x_1, x_2, \dots, x_n \rangle$ 是解

回溯

9. if $k > 1$ then $k \leftarrow k-1$; goto 4

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 $w_1+w_2+\dots+w_n \leq c_1+c_2$
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，请给出一种方案.

实例：

$W = < \underline{90}, 80, \underline{40}, 30, 20, \underline{12}, 10 >$
 $c_1=152, c_2=130$

解：1,3,6,7装第一艘船，其余第2艘船 4

求解思路

输入： $W = \langle w_1, w_2, \dots, w_n \rangle$ 为集装箱重量
 c_1 和 c_2 为船的最大载重量

算法思想： 令第一艘船的装入量为 W_1 ,

1. 用回溯算法求使得 $c_1 - W_1$ 达到最小的装载方案.
2. 若满足

$$w_1 + w_2 + \dots + w_n - W_1 \leq c_2$$

则回答 “Yes”, 否则回答 “No”

伪码

算法 Loading (W, c_1),

B为当前空隙
best 最小空隙

1. Sort(W);
2. $B \leftarrow c_1$; $best \leftarrow c_1$; $i \leftarrow 1$;
3. while $i \leq n$ do
4. if 装入 i 后重量不超过 c_1
5. then $B \leftarrow B - w_i$; $x[i] \leftarrow 1$; $i \leftarrow i + 1$;
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.

子过程 Backtrack

算法 Backtrack(i)

1. while $i > 1$ and $x[i] = 0$ do

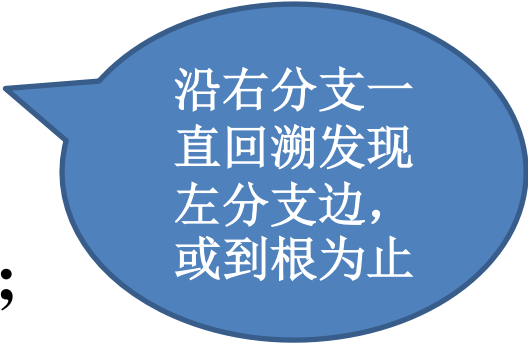
2. $i \leftarrow i - 1$;

3. if $x[i] = 1$

4. then $x[i] \leftarrow 0$;

5. $B \leftarrow B + w_i$;

6. $i \leftarrow i + 1$.



沿右分支一
直回溯发现
左分支边，
或到根为止

实例

$W = \langle 90, 80, 40, 30, 20, 12, 10 \rangle$

$c_1=152, c_2=130$

解:

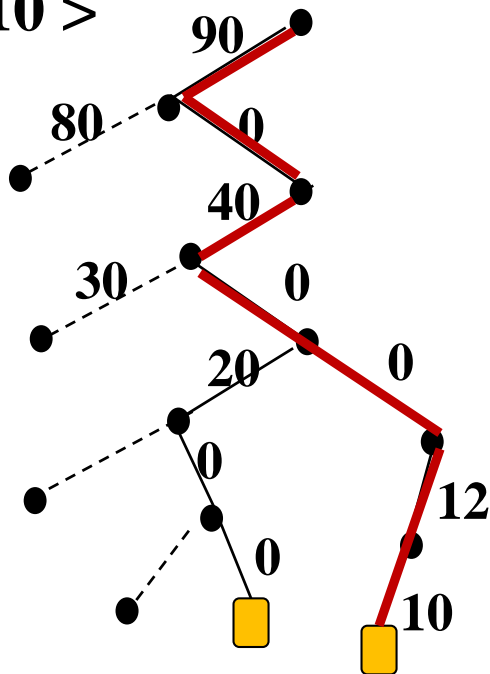
可以装, 方案如下:

1,3,6,7 装第一艘船

2,4,5 装第二艘船

时间复杂性:

$$W(n)=O(2^n)$$



小结

- 回溯算法的实现：
递归实现、迭代实现
- 装载问题
问题描述
算法伪码
最坏情况下时间复杂度 $O(2^n)$

图的着色

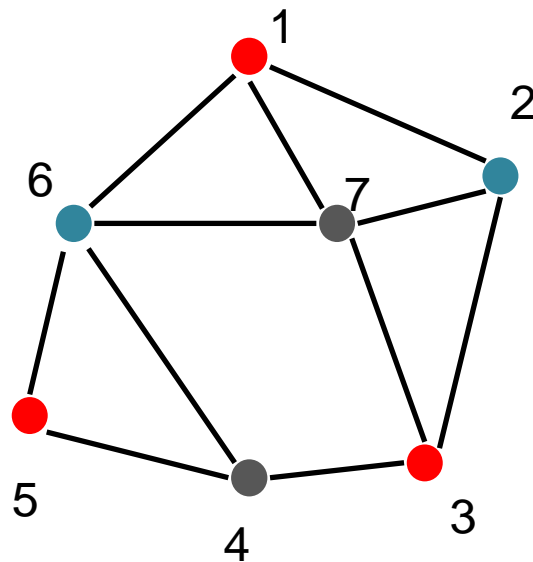
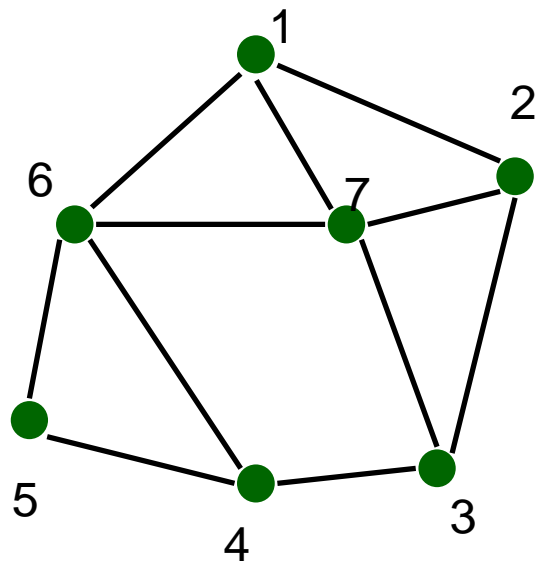
着色问题

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”.

实例



$n=7, m=3$

解向量

设 $G=(V,E)$, $V=\{1,2, \dots, n\}$

颜色编号: $1, 2, \dots, m$

解向量: $\langle x_1, x_2, \dots, x_n \rangle$,

$$x_1, x_2, \dots, x_n \in \{1, 2, \dots, m\}$$

结点的部分向量 $\langle x_1, x_2, \dots, x_k \rangle$

$$x_1, x_2, \dots, x_k, 1 \leq k \leq n,$$

表示只给顶点 $1, 2, \dots, k$ 着色的部分方案

算法设计

搜索树： m 叉树

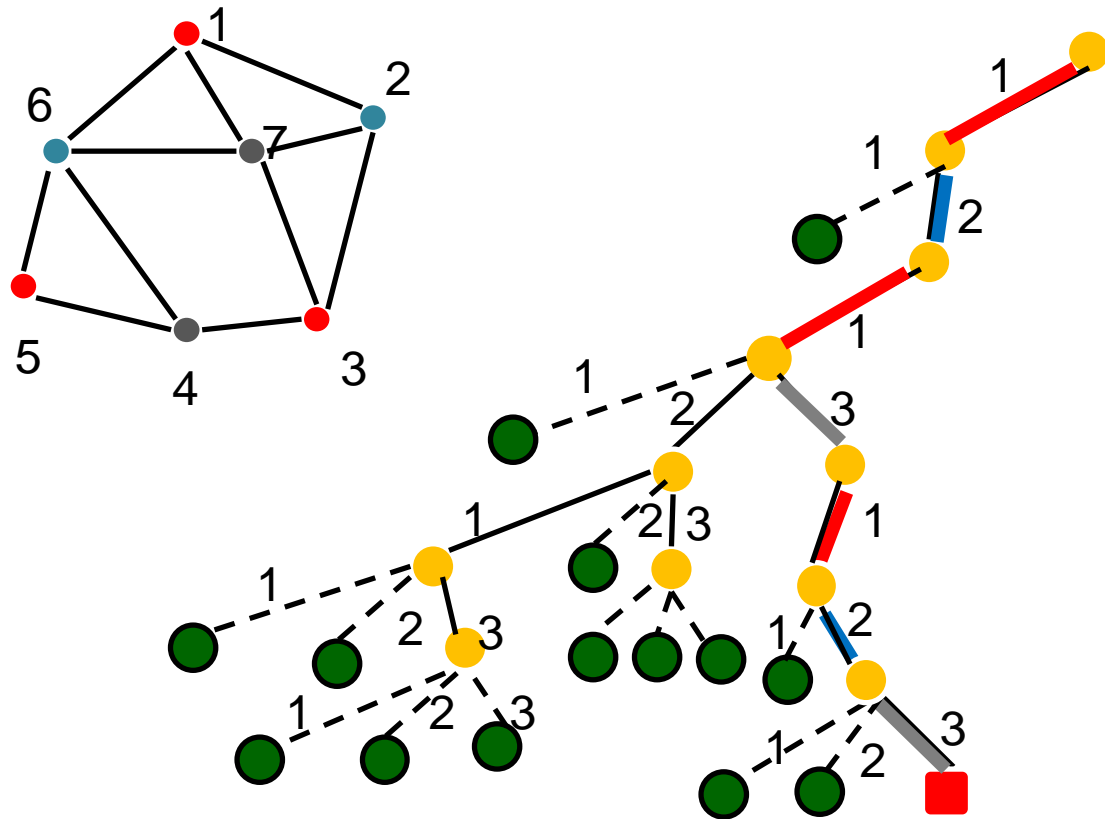
约束条件： 在结点 $\langle x_1, x_2, \dots, x_k \rangle$ 处，
顶点 $k+1$ 的邻接表中结点已用过的颜色
不能再使用

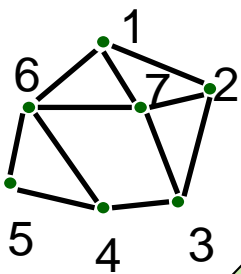
如果邻接表中结点已用过 m 种颜色，
则结点 $k+1$ 没法着色，从该结点回溯
到其父结点。满足多米诺性质

搜索策略： 深度优先

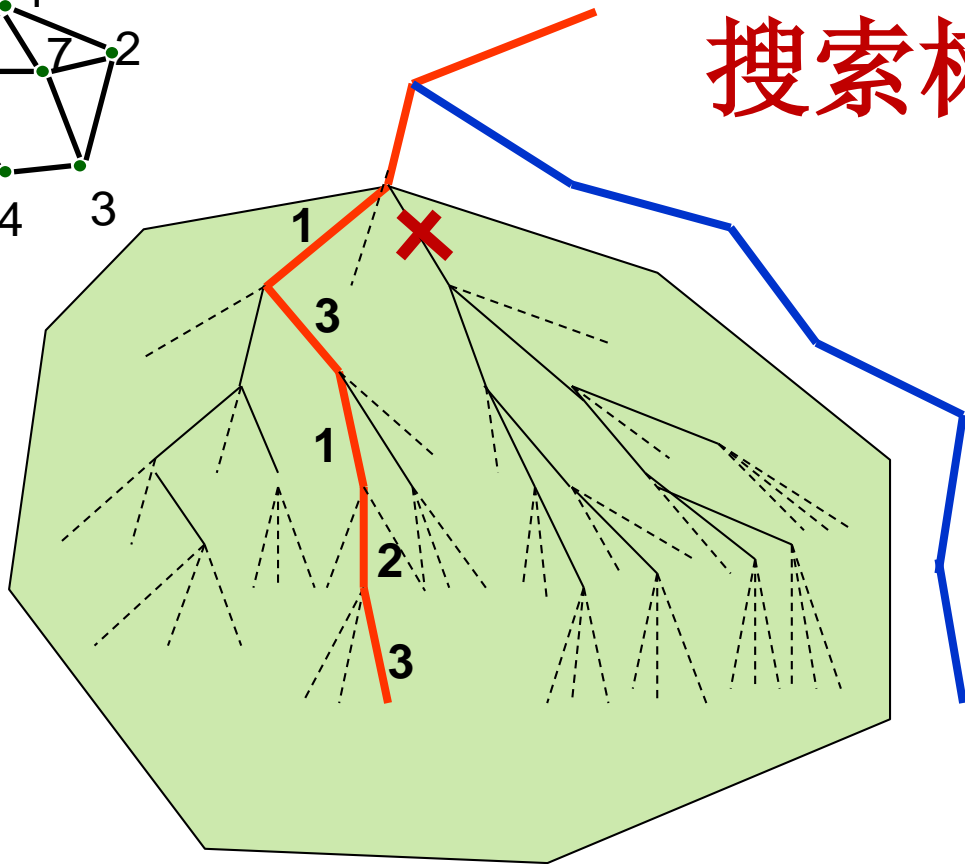
时间复杂度： $O(n m^n)$

运行实例





搜索树



第一个解向量: $\langle 1, 2, 1, 3, 1, 2, 3 \rangle_7$

时间复杂度与 改进途径

时间复杂度: $O(nm^n)$

根据对称性, 只需搜索 $1/3$ 的解空间. 当 1 和 2 确定, 即 $\langle 1, 2 \rangle$ 以后, 只有 1 个解, 在 $\langle 1, 3 \rangle$ 为根的子树中也只有 1 个解. 由于 3 个子树的对称性, 总共 6 个解.

在取定 $\langle 1, 2 \rangle$ 后, 不可扩张成 $\langle 1, 2, 3 \rangle$, 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 与 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 与 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.

小结

- 着色问题的描述
- 着色问题的算法设计
- 时间复杂度及改进途径
- 着色问题的应用

搜索树结点数 的估计

搜索树结点数估计

Monte Carlo方法

1. 从根开始，随机选择一条路径，直到不能分支为止，即从 x_1, x_2, \dots ，依次对 x_i 赋值，每个 x_i 的值是从当时的 S_i 中随机选取，直到向量不能扩张为止.
2. 假定搜索树的其他 $|S_i|-1$ 个分支与以上随机选出的路径一样，计数搜索树的点数.
3. 重复步骤 1 和 2，将结点数进行概率平均.


伪码

Monte Carlo

输入: n 为皇后数, t 为抽样次数

输出: sum , 即 t 次抽样路长平均值

1. $sum \leftarrow 0$
2. for $i \leftarrow 1$ to t do // 取样次数 t
3. $m \leftarrow \text{Estimate}(n)$ // m 为结点数
4. $sum \leftarrow sum + m$
5. $sum \leftarrow sum / t$



一次抽样结果

一次抽样

m 为本次取样得到的树结点总数

k 为层数

r_2 为上层结点数

r_1 为本层结点数

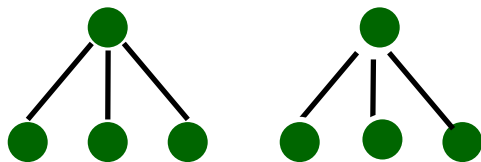
$r_1 = r_2 \cdot \text{分支数}$

n 为树的层数

从树根向下计算,随机选择,直到树叶.

$$r_2 = 2$$

$$r_1 = r_2 \cdot 3 = 6$$



子过程的伪码

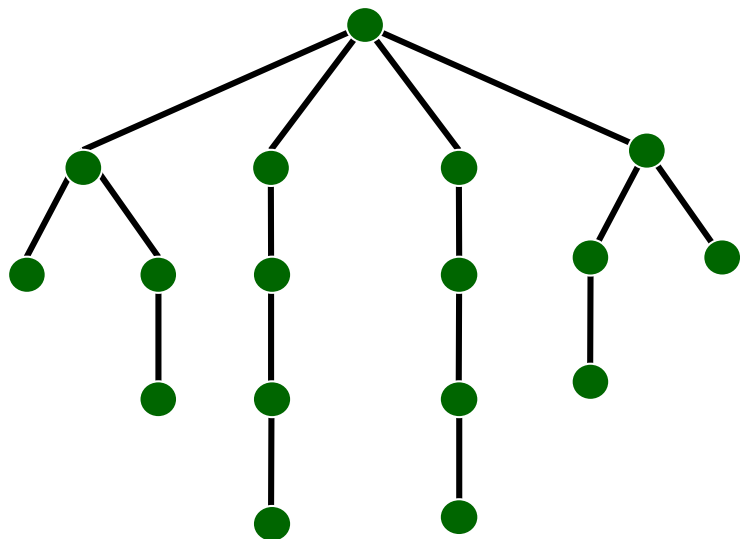
算法Estimate(n)

1. $m \leftarrow 1$; $r_2 \leftarrow 1$; $k \leftarrow 1$ // m 为结点总数
2. while $k \leq n$ do
3. if $S_k = \emptyset$ then return m
4. $r_1 \leftarrow |S_k| * r_2$ // r_1 为扩张后结点总数
5. $m \leftarrow m + r_1$ // r_2 为扩张前结点总数
6. $x_k \leftarrow$ 随机选择 S_k 的元素
7. $r_2 \leftarrow r_1$
8. $k \leftarrow k + 1$

不能
分支

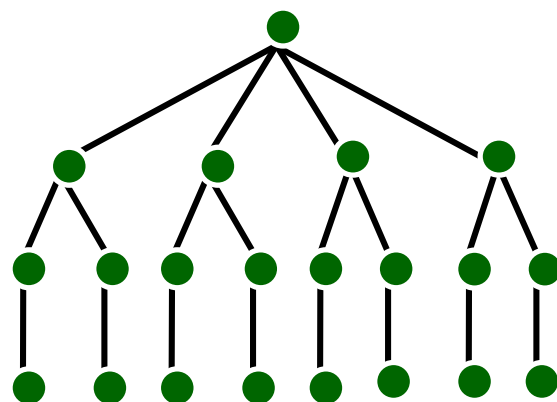
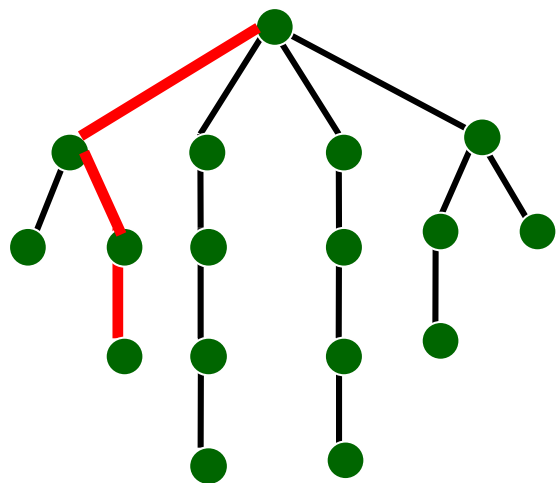
随机选
择一步

4后搜索树遍历的结点



结点数=17

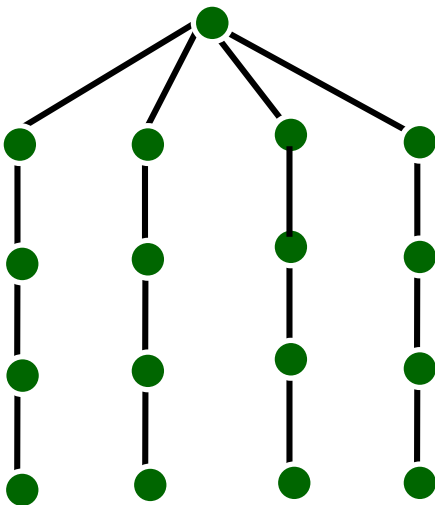
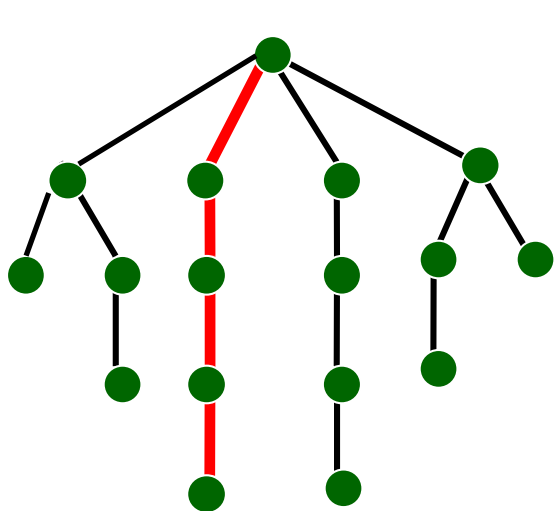
随机选择路径1



Case1: $\langle 1, 4, 2 \rangle$,

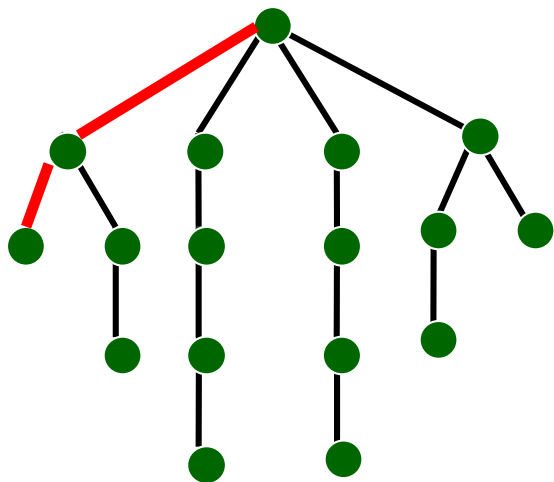
结点数 21

随机选择路径2

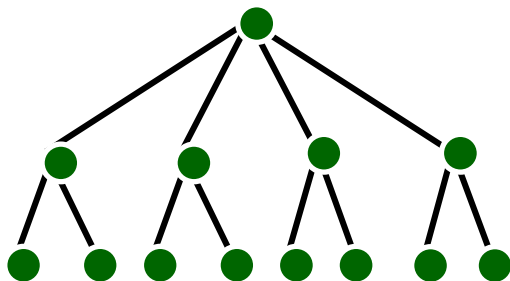


Case1: $\langle 2, 4, 1, 3 \rangle$, 结点数 17

随机选择路径3



Case3: $\langle 1, 3 \rangle$,



结点数 13

估计结果

假设 4 次抽样测试:

case1: 1次,

case2: 1次,

case3: 2次,

平均结点数 $= (21 \times 1 + 17 \times 1 + 13 \times 2) / 4 = 16$

搜索空间访问的结点数为 17

小结

- **Monte Carlo 方法**

目的： 估计搜索树真正访问结点数

步骤：

随机抽样，选择一条路径

用这条路径代替其他路径

逐层累加树的结点数

多次选择，取结点数的平均值