

# 回溯（backtrack）

---

- 回溯算法的基本思想和适用条件
- 回溯算法的设计步骤
- 估计回溯算法的效率
- 改进回溯算法的途径
- 分支估界
- 应用实例

# 基本思想和适用条件

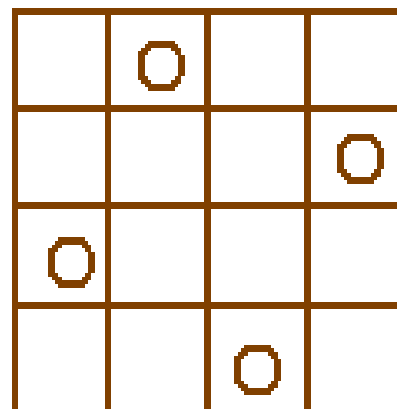
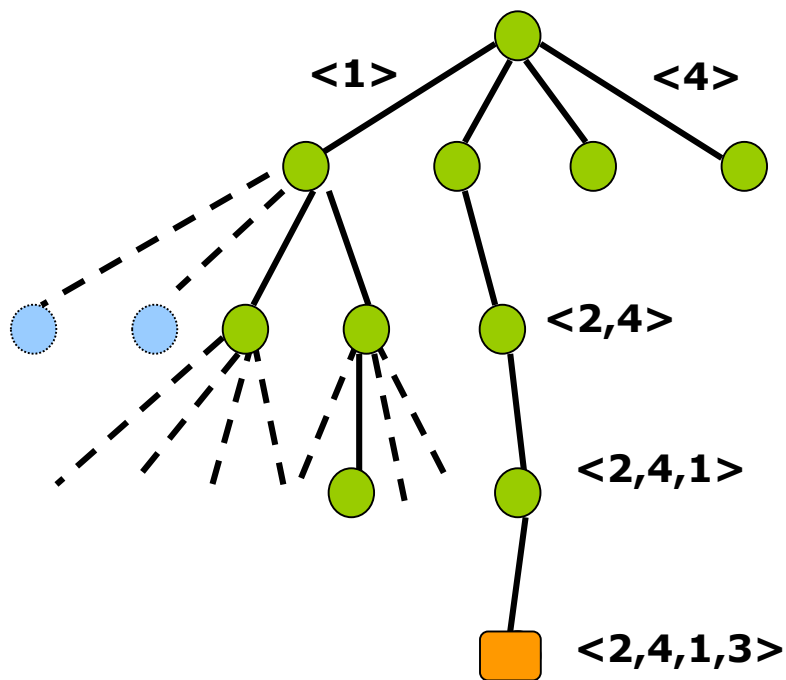
---

- 实例
- 基本思想
  - 搜索问题
  - 搜索空间
  - 搜索策略
  - 判定条件
  - 结点状态
  - 存储结构
- 必要条件

# 实例1: 四后问题

解表示成一个4维向量,  $\langle x_1, x_2, x_3, x_4 \rangle$  (放置列号)

## 搜索空间：4叉树

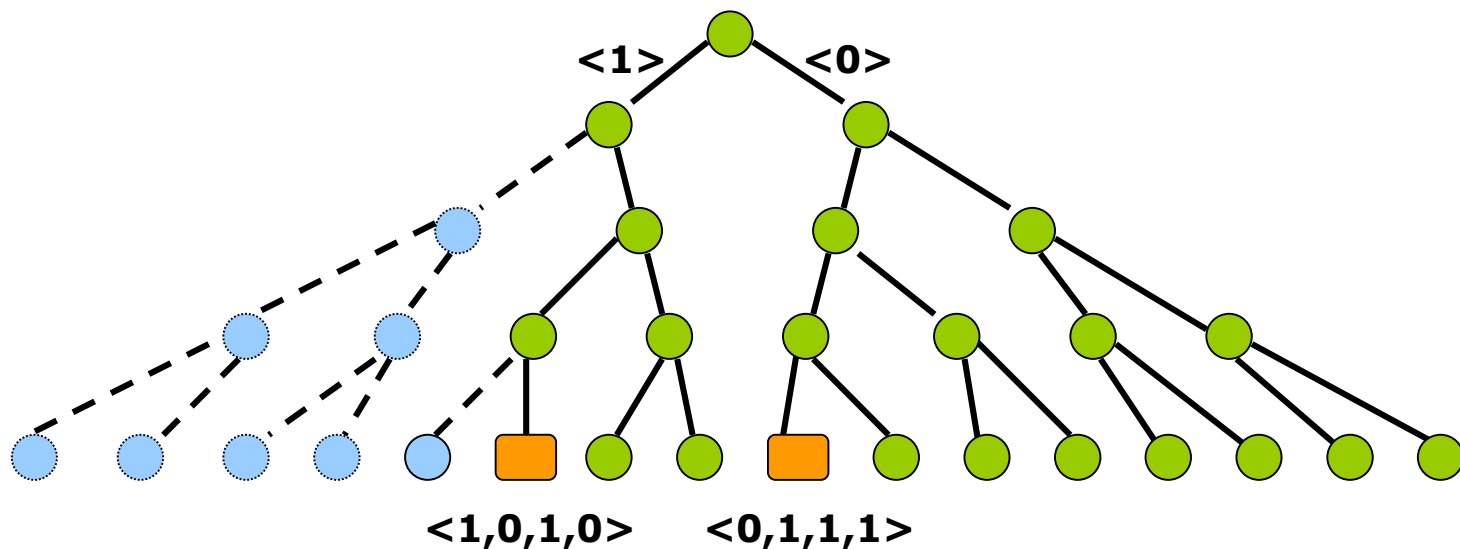


# 实例2：0-1背包问题

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

结点：向量 $\langle x_1, x_2, x_3, \dots, x_k \rangle$ （子集的部分特征向量）

搜索空间：子集树， $2^n$ 片树叶



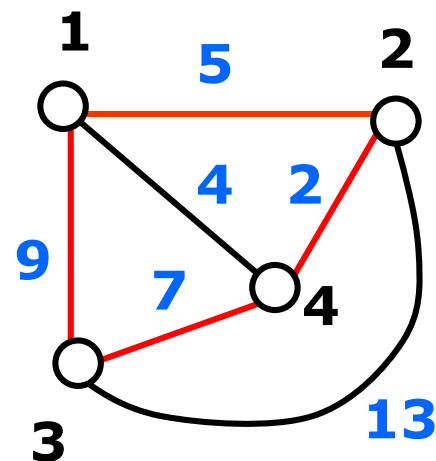
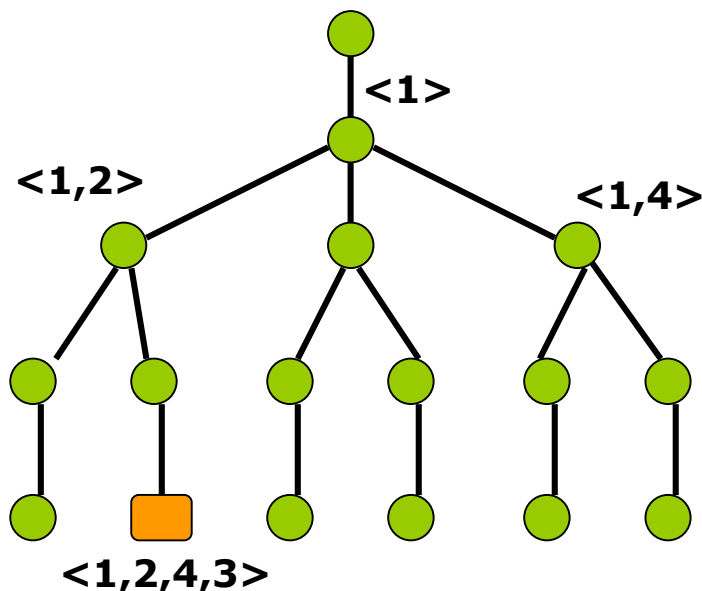
$\langle 0,1,1,1 \rangle$  可行解：  $x_1=0, x_2=1, x_3=1, x_4=1$ . 重量： 13, 价值： 28

$\langle 1,0,1,0 \rangle$ 可行解：  $x_1=1, x_2=0, x_3=1, x_4=0$ . 重量： 12, 价值： 21

# 实例3：货郎担问题

$\langle i_1, i_2, \dots, i_n \rangle$  为巡回路线

搜索空间：排列树,  $(n-1)!$  片树叶



$\langle 1, 2, 4, 3 \rangle$  对应于巡回路线:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

长度:  $5 + 2 + 7 + 9 = 23$

# 基本思想

---

- 适用问题：求解搜索问题
- 搜索空间：树，结点对应部分解向量，树叶对应可行解
- 搜索过程：采用系统的方法隐含遍历搜索树
- 搜索策略：深度优先，宽度优先，函数优先，宽深结合等
- 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
  - 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)
- 存储：当前路径

# 必要条件：多米诺性质

---

设  $P(x_1, x_2, \dots, x_i)$  为真表示向量  $\langle x_1, x_2, \dots, x_i \rangle$  中  $i$  个皇后放置在彼此不能攻击的位置

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

例4 求不等式的整数解

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

$P(x_1, \dots, x_k)$  : 意味将  $x_1, x_2, \dots, x_k$  代入原不等式的相应部分使得左边小于等于10

不满足多米诺性质

变换： 令  $x_3 = 3 - x_3'$ ,

$$5x_1 + 4x_2 + x_3' \leq 13, \quad 1 \leq x_1, x_2 \leq 3, \quad 0 \leq x_3' \leq 2$$

# 回溯算法的设计要素

---

- 定义搜索问题的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  $x_i$  的可能取值的集合为  $X_i, i = 1, 2, \dots, n$ .
- 当  $x_1, x_2, \dots, x_{k-1}$  确定以后计算  $x_k$  取值集合  $S_k, S_k \subseteq X_k$
- 确定结点儿子的排列规则
- 判断是否满足多米诺性质
- 搜索策略----深度优先、宽度优先等
- 确定每个结点分支约束条件
- 确定存储搜索路径的数据结构



# 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( $n$ )

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

# 迭代实现

---

## 迭代算法 Backtrack

1. 对于  $i = 1, 2, \dots, n$  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
9. if  $k > 1$  then  $k \leftarrow k - 1$ ; goto 4

# 估计搜索树的结点数

---

计数搜索树中遍历的结点，Monte Carlo方法

## 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。

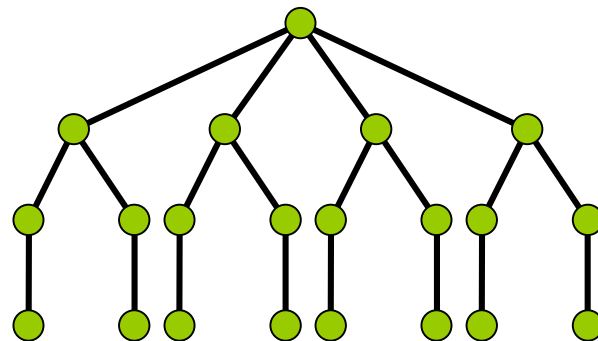
# 实例

例5 估计四后问题的效率

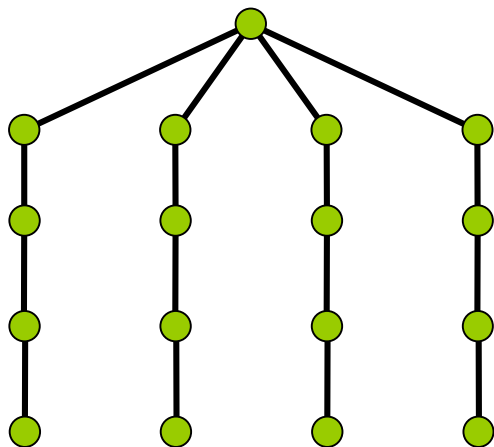
case1.  $\langle 1, 4, 2 \rangle$ :  $1 + 4 + 4 \times 2 + 4 \times 2 = 21$

case2.  $\langle 2, 4, 1, 3 \rangle$ :  $4 \times 4 + 1 = 17$

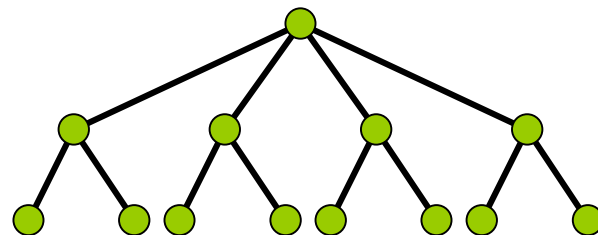
case3.  $\langle 1, 3 \rangle$ :  $1 + 4 \times 1 + 4 \times 2 = 13$



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



Case2:  $\langle 2, 4, 1, 3 \rangle$



Case3:  $\langle 1, 3 \rangle$

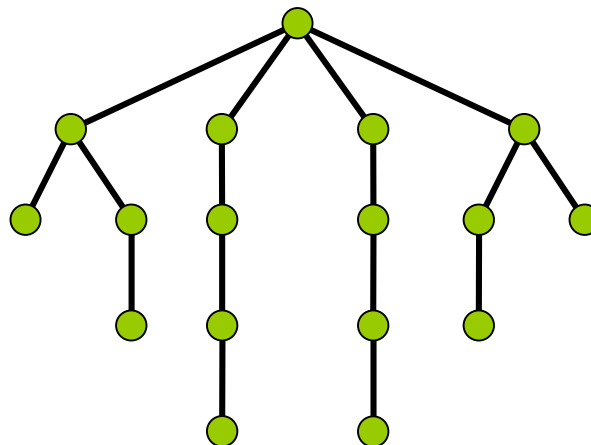
# 估计结点数

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



搜索空间

# 算法实现

---

## Monte Carlo

1.  $sum \leftarrow 0$       //  $sum$  为  $t$  次结点平均数
2. for  $i \leftarrow 1$  to  $t$  do    // 取样次数  $t$
3.      $m \leftarrow \text{Estimate}(n)$     //  $m$  为本次结点总数
4.      $sum \leftarrow sum + m$
5.  $sum \leftarrow sum / t$

# 结点数计算

$m$ 为输出——本次取样结点总数,  $k$  为层数,  $r_1$ 为本层分支数,  $r_2$ 为上层分支数,  $n$ 为树的层数

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.      $k \leftarrow k+1$

# 影响算法效率的因素

---

最坏情况下的时间  $W(n) = (p(n)f(n))$

其中  $p(n)$  为每个结点时间,  $f(n)$  为结点个数

影响回溯算法效率的因素

搜索树的结构

分支情况: 分支均匀否

树的深度

对称程度: 对称适合裁减

解的分布

在不同子树中分布多少是否均匀

分布深度

约束条件的判断: 计算简单



# 改进途径

---

- 根据树分支设计优先策略:

结点少的分支优先, 解多的分支优先

- 利用搜索树的对称性剪裁子树

- 分解为子问题:

求解时间  $f(n)=c2^n$ , 组合时间  $T=O(f(n))$

如果分解为  $k$  个子问题, 每个子问题大小为  $n/k$

求解时间为

$$kc^{\frac{n}{k}} + T$$

# 组合优化问题

---

## □ 相关概念

- 目标函数（极大化或极小化）
- 约束条件
- 搜索空间中满足约束条件的解称为可行解
- 使得目标函数达到极大(或极小)的解称为最优解

## □ 实例：背包问题

$$\begin{aligned}\max \quad & x_1 + 3x_2 + 5x_3 + 9x_4 \\ & 2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10 \\ & x_i \in N, i = 1, 2, 3, 4\end{aligned}$$

# 分支估界技术（极大化）

---

## □ 设立代价函数

- 函数值以该结点为根的搜索树中的所有可行解的目标函数值的上界
- 父结点的代价不小于子结点的代价

## □ 设立界

- 代表当时已经得到的可行解的目标函数的最大值
- 界的设定初值可以设为0
- 可行解的目标函数值大于当时的界，进行更新

## □ 搜索中停止分支的依据

- 不满足约束条件或者其代价函数小于当时的界