算法设计与分析

回溯法 (一)

1 经典例子
2 回溯法原理
3 装载问题
4 图着色问题
5 叶节点估计

回溯范式

递归方法本质上是遍历由递归关系定义的整棵树。

- 子树可能重复，因此需要缓存中间结果以提高效率。这正是动态规划的本质。

回溯范式

递归方法本质上是遍历由递归关系定义的整棵树。

- 子树可能重复，因此需要缓存中间结果以提高效率。这正是动态规划的本质。

对于某些问题，子树不会重叠。

- 在这种情况下，除了遍历整棵树之外没有更好的算法。但是，我们可以聪明地遍历整棵树。
- 这就是回溯技术所关注的：如果解不会出现在子树中，则停止访问该子树并回溯到父节点
  - 基本回溯策略：由问题约束定义的多米诺性质
  - 高级回溯策略：分支定界

目录

例 1：八皇后问题

八皇后问题：在 8×8 的棋盘上放置八个国际象棋皇后，使得任意两个皇后之间不能互相攻击。

- 一个解要求任意两个皇后不能在同一行、同一列或同一对角线上。

八皇后问题是更一般的 n 皇后问题的特例：在 $n \times n$ 的棋盘上放置 n 个不能互相攻击的皇后。

[见原文第 5 页图：8 皇后解的示例]

解的计数

解是一个 n 维向量，取值范围为 $[n]$：对所有自然数 n 存在解，除了 $n = 2, 3$。

- 八皇后问题有 92 个不同的解，整个解空间大小为 $C_{64}^{8} = 4,426,165,368$。
- 如果将仅通过棋盘的旋转和反射对称操作而不同的解算作一个，则该问题有 12 个解，称为基本解。

n	基本解	全部解
8	12	92
9	46	352
10	92	724
...	...	...
26	2,789,712,466,510,289	22,317,699,616,364,044
27	29,363,495,934,315,694	234,907,967,154,122,528

八皇后问题的背景

八皇后问题的起源

Max Bezzel 于 1848 年首次提出这个问题，Frank Nauck 于 1850 年给出了第一个解并将其推广到 n 皇后问题。包括 Carl Gauss 在内的许多数学家也研究过这个问题。
Edsger Dijkstra 通过这个问题展示了深度优先回溯算法的威力。

> 目前没有已知的精确解数量公式，甚至没有其渐近行为的公式。27×27 棋盘是已被完全枚举的最高阶棋盘。

如何求解?

- 将所有可能的解建模为树的 n 层叶节点
- 通过遍历树来遍历解空间

4 皇后问题演示

[见原文第 8 页图：4 皇后搜索树和棋盘示例]

按深度优先顺序遍历树以找到所有解

- 第 i 层节点表示解向量的前 i 个分量
- 在第 i 层，分支选择数少于 $n-(i-1)$
- n 层叶节点对应解

例 2：旅行商问题

问题：给定 n 个城市 $C = \{c_1, c_2, \ldots, c_n\}$ 和 $d(c_i, c_j) \in \mathbb{Z}^+$。找到一个访问每个城市恰好一次的最短环路。
解：$(1, 2, \ldots, n)$ 的一个排列 $(k_1, k_2, \ldots, k_n)$ 使得

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

[见原文第 9 页图：4 城市 TSP 示例图]

$C = \{1, 2, 3, 4\}$，$d(1, 2) = 5$，$d(1, 3) = 9$，$d(1, 4) = 4$，$d(2, 3) = 13$，$d(2, 4) = 2$，$d(3, 4) = 7$
解为 $(1, 2, 4, 3)$，环路长度为 $5 + 2 + 7 + 9 = 23$

TSP 的搜索空间

[见原文第 10 页图：TSP 搜索树]

任何节点都可以作为根节点，因为 TSP 定义在无向图上。

搜索空间：在第 i 层，分支选择数始终为 $n-i$

- 得到一棵有 $(n-1)!$ 个叶节点的树 $\rightsquigarrow$ 在循环移位下 $\{1,\ldots,n\}$ 的所有可能排列数

例 3：0-1 背包问题

问题：给定 n 个物品，价值为 v_i，重量为 w_i，以及一个容量为 W 的背包。每种物品数量为 1。找到一个使价值最大化的解。
解：n 维向量 $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$，$x_i = 1 \Leftrightarrow$ 选择物品 i
节点：$(x_1, x_2, \ldots, x_k)$ 对应部分解
搜索空间：在所有层，分支选择数始终为 2 $\rightsquigarrow$ 具有 2^n 个叶节点的完美二叉树
候选解：满足约束 $\sum_{i=1}^{n} w_i x_i \leq W$
最优解：达到最大价值的候选解。

背包问题演示

表: $n=4$，$W=13$

物品	1	2	3	4
价值	12	11	9	8
重量	8	6	4	3

[见原文第 12 页图：背包问题搜索树]

两个候选解

1. $(0,1,1,1)$：$v=28$，$w=13$
2. $(1,0,1,0)$：$v=21$，$w=12$

最优解是 $(0,1,1,1)$

总结

回溯的经典例子

- n 皇后问题、TSP、0-1 背包

解：向量

搜索空间：树，其中节点对应部分解，叶节点对应候选解

- n 皇后问题、TSP：排列树
- 0-1 背包：完美二叉树

搜索顺序：深度优先、广度优先、跳跃式

目录

回溯法的主要思想

适用范围：搜索或优化问题

搜索空间：树

- 叶节点：候选/可行解
- 节点：部分解

如何搜索：系统地遍历树：DFS、BFS、...

[见原文第 15 页图：DFS 和 BFS 遍历示例树]

DFS: $1 \to 2 \to 3 \to 5 \to 8 \to 9 \to 6 \to 7 \to 4$

BFS: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$

节点的状态

树是动态探索的。设 v 为候选节点（对应部分解），P 为检查 v 是否满足约束的谓词。

- $P(v) = 1 \Rightarrow$ 扩展
- $P(v) = 0 \Rightarrow$ 回溯到父节点

节点的状态

- 白色：未探索
- 灰色：正在访问其子树
- 黑色：已完成该子树的遍历

[见原文第 16 页图：节点状态示例]

DFS: $1 \to 2 \to 3 \to 5 \to 8$

- 已完成访问：2, 8
- 正在访问：1, 3, 5
- 未探索：9, 6, 7, 4

基本回溯技术：多米诺性质

在节点 $v = (x_1, \ldots, x_k)$

$$P(x_1, \ldots, x_k) = 1 \Leftrightarrow (x_1, \ldots, x_k) \text{ 满足某些性质}$$

示例：n 皇后问题，放置 k 个皇后在互不攻击的位置
多米诺性质 $\rightsquigarrow$ 允许安全回溯

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

逆否命题

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

k 维向量不满足约束 $\Rightarrow$ 其 $k+1$ 维扩展也不满足约束

- 保证回溯不会遗漏任何解
- 当 $P(x_1, x_2, \ldots, x_k) = 0$ 时可安全回溯

一个反例

求不等式的整数解

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

一个反例

求不等式的整数解

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

$P(x_1, \ldots, x_k) = 1$ 当且仅当 $\sum_{i=1}^{k} a_i x_i \le 10$ 不满足多米诺性质

$$5x_1 + 4x_2 - x_3 \le 10 \nRightarrow 5x_1 + 4x_2 \le 10$$

[见原文第 19 页图：挠头三连表情包]

一个反例

求不等式的整数解

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

$P(x_1, \ldots, x_k) = 1$ 当且仅当 $\sum_{i=1}^{k} a_i x_i \leq 10$ 不满足多米诺性质

$$5x_1 + 4x_2 - x_3 \leq 10 \nRightarrow 5x_1 + 4x_2 \leq 10$$

[见原文第 20 页图：挠头三连表情包]

修改以满足多米诺性质：设 $x_3' = 3 - x_3$

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

一个反例

求不等式的整数解

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

$P(x_1, \ldots, x_k) = 1$ 当且仅当 $\sum_{i=1}^{k} a_i x_i \leq 10$ 不满足多米诺性质

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

[见原文第 21 页图：挠头三连表情包]

修改以满足多米诺性质：设 $x_3' = 3 - x_3$

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

2022 级贾梦涵：3 可以推广到任意正整数 ≥ 3。

总结

使用回溯的前提条件：多米诺性质

回溯算法的一般步骤

- 定义解向量（包括每个元素的范围），$(x_1, x_2, \ldots, x_n) \in X_1 \times \cdots \times X_n$
- 在固定 $(x_1, x_2, \ldots, x_{k-1})$ 后，使用谓词 P 将 x_k 的可行范围更新为 $A_k \subseteq X_k$
- 判断是否满足多米诺性质
- 确定搜索策略：DFS、BFS
- 确定存储搜索路径的数据结构

回溯递归模板

Algorithm 1 BackTrack(n) // 输出所有解

for $k = 1$ **to** n **do**
　　$A_k \leftarrow X_k$ // 初始化
end for
ReBack(1)

Algorithm 2 ReBack(k) // k 是当前递归深度

if $k = n$ **then**
　　记录解 $(x_1, \ldots, x_n)$
else
　　while $A_k \neq \emptyset$ **do**
　　　　$x_k \leftarrow A_k$ // 按某种顺序
　　　　$A_k \leftarrow A_k - \{x_k\}$
　　　　更新 A_{k+1}， ReBack($k+1$)
　　end while

目录

装载问题

问题：给定 n 个集装箱，重量为 w_i，两艘船的载重能力分别为 W_1 和 W_2，满足 $w_1 + \cdots + w_n \leq W_1 + W_2$。
目标：判断是否存在将 n 个集装箱装上两艘船的方案。如果可解，请给出一个方案。

装载问题

问题：给定 n 个集装箱，重量为 w_i，两艘船的载重能力分别为 W_1 和 W_2，满足 $w_1 + \cdots + w_n \leq W_1 + W_2$。
目标：判断是否存在将 n 个集装箱装上两艘船的方案。如果可解，请给出一个方案。

示例

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$， $W_1 = 152$， $W_2 = 130$

解：将 1, 3, 6, 7 装上船 1，其余装上船 2

装载问题

问题：给定 n 个集装箱，重量为 w_i，两艘船的载重能力分别为 W_1 和 W_2，满足 $w_1 + \cdots + w_n \leq W_1 + W_2$。
目标：判断是否存在将 n 个集装箱装上两艘船的方案。如果可解，请给出一个方案。

示例

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$，$W_1 = 152$，$W_2 = 130$

解：将 1, 3, 6, 7 装上船 1，其余装上船 2
主要思想：设总重量为 W。

1. 先装船 1。使用回溯找到一个使 W_1^* 最大化的解，其中 W_1^* 是实际装载量。
2. 然后检查 $W - W_1^* \leq W_2$。如果为真返回" 是"，否则返回" 否"。

装载问题

问题：给定 n 个集装箱，重量为 w_i，两艘船的载重能力分别为 W_1 和 W_2，满足 $w_1 + \cdots + w_n \leq W_1 + W_2$。
目标：判断是否存在将 n 个集装箱装上两艘船的方案。如果可解，请给出一个方案。

示例

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$， $W_1 = 152$， $W_2 = 130$

解：将 1, 3, 6, 7 装上船 1，其余装上船 2
主要思想：设总重量为 W。

1. 先装船 1。使用回溯找到一个使 W_1^* 最大化的解，其中 W_1^* 是实际装载量。
2. 然后检查 $W - W_1^* \leq W_2$。如果为真返回" 是"，否则返回" 否"。

时间复杂度：$W(n) = O(2^n)$

伪代码（搜索完美二叉树）

装载演示

$W_1 = 152$，$W_2 = 130$

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$

[见原文第 30 页图：装载问题搜索树]

装载演示

$W_1 = 152$， $W_2 = 130$

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$

[见原文第 31 页图：装载问题搜索树（继续）]

装载演示

$W_1 = 152$， $W_2 = 130$

$w_1 = 90, w_2 = 80, w_3 = 40, w_4 = 30, w_5 = 20, w_6 = 12, w_7 = 10$

[见原文第 32 页图：装载问题搜索树（最终）]

可装载：1, 3, 6, 7 装船 1；2, 4, 5 装船 2

目录

图着色问题

问题：无向图 G 和 m 种颜色。对顶点着色以确保相连的两个顶点颜色不同。
目标：输出所有可能的着色方案。如果没有则输出" 否"。

图着色问题

问题：无向图 G 和 m 种颜色。对顶点着色以确保相连的两个顶点颜色不同。
目标：输出所有可能的着色方案。如果没有则输出" 否"。

[见原文第 35 页图：7 顶点图的 3 着色示例]

$$n = 7, \quad m = 3$$

算法设计

输入：$G=(V,E)$，$V=\{1,2,\ldots,n\}$，颜色集合 $\{1,2,\ldots,m\}$
解向量：$(x_1,x_2,\ldots,x_n)$，$x_i\in[m]$
$(x_1,\ldots,x_k)$ 给出顶点集 $\{1,2,\ldots,k\}$ 的部分解
搜索树：m 叉树
约束：在节点 $(x_1,\ldots,x_k)$，节点 $k+1$ 的可用颜色集合不能为空。

- 如果邻接表中的节点已用完 m 种颜色，则节点 $k+1$ 不可着色。此时回溯到父节点。（多米诺性质显然成立）

搜索策略：DFS
时间复杂度：$O(nm^n)$

- 树的深度为 $n \Rightarrow$ 最多 m^n 个节点
- 每步需要找可用颜色 $\Rightarrow$ 需要 $O(n)$ 代价

演示

[见原文第 37 页图：图着色搜索树和图示例]

第一个解向量：$(1, 2, 1, 2, 1, 2, 3)$

搜索树的结构

[见原文第 38 页图：搜索树结构 - 第一步]

第一个解向量：$(1,2,1,2,1,2,3)$

搜索树的结构

[见原文第 39 页图：搜索树结构 - 第二步]

第一个解向量：$(1, 2, 1, 2, 1, 2, 3)$

搜索树的结构

[见原文第 40 页图：搜索树结构 - 最终]

第一个解向量：$(1, 2, 1, 2, 1, 2, 3)$

复杂度分析

时间复杂度：$O(nm^n)$

对称性 $\rightsquigarrow$ 只需搜索最多 $1/6$ 的解空间

- $(1,2,3)$ 的排列数是 6 $\rightsquigarrow$ 对于任何特定解，存在 6 个同构解
- 第 2 层有 2 倍解（例如蓝色和绿色可互换），第 1 层有 3 倍解（节点 1 可以选红、绿或蓝）；越接近根，替换选择越多。

复杂度分析

时间复杂度：$O(nm^n)$

对称性 $\rightsquigarrow$ 只需搜索最多 $1/6$ 的解空间

- $(1,2,3)$ 的排列数是 6 $\rightsquigarrow$ 对于任何特定解，存在 6 个同构解
- 第 2 层有 2 倍解（例如蓝色和绿色可互换），第 1 层有 3 倍解（节点 1 可以选红、绿或蓝）；越接近根，替换选择越多。

额外推理也有助于减少搜索范围

- 示例：如果节点 1, 2, 3 已被不同着色，则节点 7 肯定不可着色，因为它与节点 1, 2, 3 相连 $\rightsquigarrow$ 从该节点回溯
- 需要在搜索和判断之间权衡

图着色的应用

会议室安排

有 n 个活动需要安排，如果活动 i 和活动 j 的时间段重叠，则称 i 和 j 不兼容。如何用最少的会议室安排这些活动?

图着色的应用

会议室安排

有 n 个活动需要安排，如果活动 i 和活动 j 的时间段重叠，则称 i 和 j 不兼容。如何用最少的会议室安排这些活动?

建模

- 将活动视为节点，如果 i, j 不兼容，则在 i 和 j 之间添加一条边。
- 将会议室视为颜色。

安排问题转化为找到一个使用最少颜色的着色方案。

目录

叶节点估计

有时，我们需要知道问题的规模（用节点数来衡量）

- 找到精确数量可能需要穷举遍历整棵树，这等价于解决问题本身。

蒙特卡洛方法

1. 选择一条从根到叶节点的随机路径，即随机顺序地给 $x_1, x_2, \ldots$ 赋值，直到向量无法继续扩展。
2. 假设其他 $|A_i| - 1$ 个分支与所选分支具有相同结构，计算搜索树的节点数
3. 重复步骤 1 和 2，计算平均节点数。

估计 n 皇后问题

一次采样

参数

- ℓ 是节点总数
- k 是深度
- A_k 是由节点 $(x_1, \ldots, x_{k-1})$ 确定的可用分支节点集合
- r_{prev}： 上一层的节点数
- r_{current}： 当前层的节点数
- $r_{\text{current}} = r_{\text{prev}} \times |A_k|$，其中 $|A_k| =$ 分支数
- n 是树的深度

计算顺序： 随机选择直到到达叶节点

[见原文第 48 页图： 两层树示例]

$$r_{\text{prev}} = 2，\ r_{\text{current}} = r_{\text{prev}} \cdot 3 = 6$$

伪代码

Algorithm 3 Estimate(n)

$\ell \leftarrow 1$; $r_{\text{prev}} \leftarrow 1$; $k \leftarrow 1$; $x_0 =$ 根节点
while $k \leq n$ **do**
    从 $(x_1, \ldots, x_{k-1})$ 确定 A_k
    if $A_k = \emptyset$ **then**
        return ℓ // 没有更多分支，到达叶节点
    end if
    $x_k \xleftarrow{R} A_k$ // 随机选择一个分支
    $r_{\text{current}} \leftarrow r_{\text{prev}} \times |A_k|$ // 第 k 层的节点数
    $\ell \leftarrow \ell + r_{\text{current}}$
    $r_{\text{prev}} \leftarrow r_{\text{current}}$
    $k \leftarrow k + 1$
end while

实际案例：4 皇后问题

[见原文第 50 页图：4 皇后搜索树 - 17 个节点]

17 个节点

随机选择路径 1

[见原文第 51 页图：路径 $(1,4,2)$]

情况 1：$(1,4,2)$　21 个节点

随机选择路径 2

[见原文第 52 页图：路径 $(2,4,1,3)$]

情况 2：$(2,4,1,3)$ 17 个节点

随机选择路径 3

[见原文第 53 页图：路径 $(1,3)$]

情况 3：$(1,3)$　13 个节点

估计结果

假设采样四次

- 情况 1：1 次
- 情况 2：1 次
- 情况 3：2 次

平均节点数：$(21 \times 1 + 17 \times 1 + 13 \times 2)/4 = 16$
实际节点数：17

- 更多的采样会使估计值接近实际值