

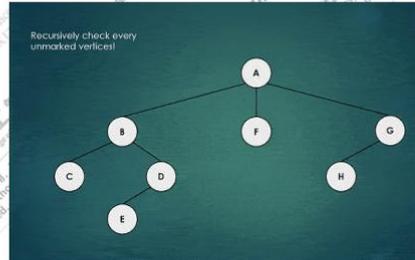
搜索求解

一. 盲目搜索 (无信息搜索)

1. 深度优先搜索 (DFS)

深度优先搜索 (Depth First Search, DFS)

- **基本思想: 优先扩展深度最深的节点**
- **从根节点 (或任何任意节点为根节点) 开始, 在回溯之前沿每个分支搜索至深度界限**
 - 1) 访问根节点
 - 2) 检查根是否有邻居/子节点, 如果是, 那就去看子节点
 - 3) 检查当前节点是否有邻居/子节点。如果是, 则搜索其子节点
 - 4) 重复此过程 (第3步), 直到到达没有任何子节点的节点 (即叶节点)
 - 5) 在叶节点上, 将其后一级遍历返到其父节点, 并检查是否有未访问的子节点。如果是, 则访问该节点并对该节点重复步骤3。如果不是, 则返回重复步骤4
 - 6) 如果访问了所有节点, 则结束



深度优先搜索从树的根 (顶部) 节点开始, 尽可能沿着给定的分支 (路径) 向下移动, 然后回溯直到找到一条未探索的路径, 然后对其进行探索直到探索了整个图。对每一个可能的分支路径深入到不能再深入为止, 而且每个节点只能访问一次

深度优先搜索的特点

- 用于树遍历算法, 也称为树搜索, 与问题无关, 具有**通用性**, 一般仅适用于求解比较简单的问题
- **每次选择一个深度最深的节点进行扩展**, 如果有相同深度的多个节点, 则按照事先的规定从中选择一个; 若该节点没有子节点, 则会选择一个深度最深而又不包含该节点的节点进行扩展, 以此类推, 直到找到问题的解结束; 或再没有可扩展节点结束, 即没有找到问题的解
- 对于许多问题可能会导致沿着一个“错误”路线搜索而陷入“深渊”, **一般不能保证找到最优解**, 可能遇到“死循环”, 是**不完备搜索**
- 为避免重复状态和冗余的图搜索, 根据具体问题可加入合理的**深度限制**: 如一个节点的深度达到深度限制, **强制进行回溯**, 选择一个稍浅的节点扩展, 而不是沿着最深的节点继续扩展, **限制过深则影响求解效率, 限制过浅则可能找不到解**
- 也用于树遍历算法, 也称为树搜索要多次遍历, 搜索所有可能路径, 在深度很大的情况下效率不高, **通常占用大量时间和空间**
- 存在**搜索与回溯交替出现**的现象

八数码问题—深度优先搜索求解



DFS 特例：回溯搜索

特点：不保留完整树结构、适用于没有明确动态规划和递归解法的问题

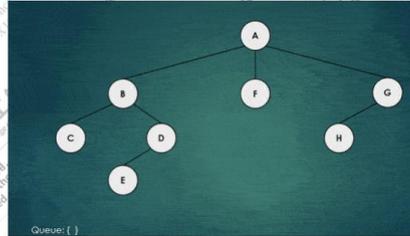
例子：N 皇后问题

2. 广度优先搜索 (BFS)

广度 (宽度) 优先搜索 (Breadth First Search, BFS)

- **基本思想：优先扩展同级直接相连的节点**
- **以接近起始节点的程度依次扩展节点、逐层搜索：从根节点开始，并在移动到下一个深度级别的节点之前，探索当前深度的所有邻居节点**

- 1) 选择图的根
- 2) 将根节点插入队列
- 3) 从队列中弹出一个顶点，将其标记为已访问，然后输出其值
- 4) 访问其未访问的邻居顶点，将其推入队列，并标记为已访问
- 5) 重复步骤3，直到队列为空
- 6) 当队列为空时，结束程序



黄色：已将节点添加到队列中，以便下次访问
红色：标记为已访问并已从队列中删除的节点

时间复杂度等效于 $O(V + E)$ ，其中 V 是顶点的数量， E 是图的边数

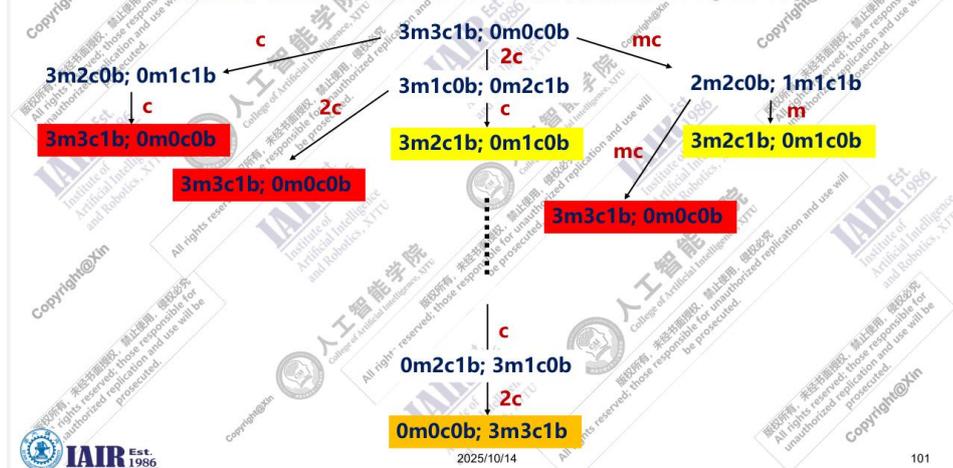
也称为级别顺序遍历

通常用于 GPS 导航，查找路径，周期检测等

广度优先搜索的特点

- **优先搜索深度浅的节点，每次选择深度最浅的叶节点进行扩展，如果有相同深度的多个节点，则按照事先的规定从深度最浅的几个节点中选择一个**
- **与问题无关，具有通用性**
- **当问题有解时，一定能找到解，而且一定能找到最优解**
- **对于解决最短或最少问题特别有效，而且寻找深度小，是完备的搜索**
- **若找到目标节点，一定是最浅的目标节点，最浅的目标节点不一定是目标节点**
- **如果路径是非递减函数，广度搜索是最优的**

传教士与野人问题：广度优先搜索 (BFS)



3. DFS 与 BFS 的对比

深度优先搜索与广度优先搜索比较

	深度优先 (DFS)	广度优先 (BFS)
完备性	不一定 (若解不在某分支, 且该分支又是个无穷分支, 则不可能得到解)	完备 (在分支因子 b 有限情况下)
最优性	不具备	<ul style="list-style-type: none"> ● 最优 (如果路径代价是节点深度的非递增函数的情况下); ● 不一定最优 (通常情况下)
时间复杂性	b^m (如果 $m > b$ 时间复杂度会很高, 但如果解决方案密集, 可能比BFS快得多)	b^d
空间复杂性	b^m	b^d

b : 分支因子; d : 解的深度
 m : 搜索树的最大深度 l : 深度限制

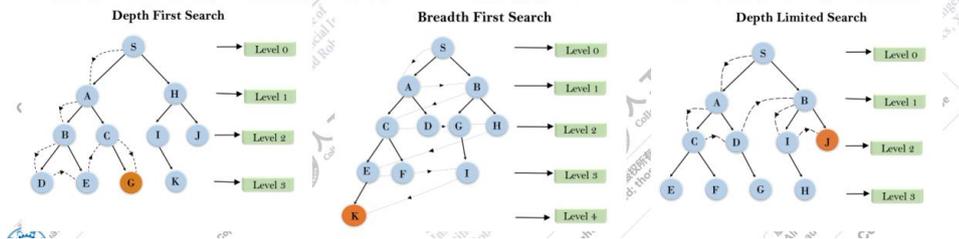
二. 启发式搜索 (有信息搜索)

1. 一致代价搜索 / Dijkstra 算法

以广度优先搜索为基础, 每次扩张代价最小的节点

一致代价搜索 (Uniform-cost Search) / Dijkstra算法

- **一致代价搜索**: 总是扩展路径消耗最小的节点 N , N 点的路径消耗等于前一节点 $N-1$ 的路径消耗加上 $N-1$ 到 N 节点的路径消耗
- 一致代价搜索是在广度优先搜索上进行扩展的, 使用优先级队列并在边缘中的状态发现更小代价的路径时引入的额外的检查, 也称为代价一致搜索
- 一致代价搜索实际上是在广度优先搜索的基础上进行扩展, 如果每一步的代价全部相等, 则相当于广度优先



2. 爬山搜索

每次随机选择一个位置, 与邻节点比较并前往更大值。(局部最优解, 能否得到全局最优解取决于初值)

3. 贪婪搜索

A. 启发函数和评价函数

启发函数 $h(n)$ 评估状态与目标状态的接近程度 (越小表示越接近)

已付出代价记为 $g(n)$

评价函数 $f(n) = g(n) + h(n)$ 评价节点的重要程度

B. 贪婪最佳优先搜索

令 $f(n) = h(n)$, 即只考虑与目标状态的接近程度, 优先扩展离目标状态最近的节点

- 1) 局部择优, 不是最优搜索
- 2) 对错误起点敏感, 可能有死循环

- 3) 不完备
- 4) 时间/空间复杂度为 $O(b^m)$
- C. A 搜索
 - 令 $f(n)=g(n)+h(n)$, 每次选择具有最小评价函数值的节点扩展
- D. A*搜索
 - 对 A 搜索的启发函数 $h(n)$ 作了规定:

最佳优先搜索——A*搜索

- **A*搜索**: 最小化总的解决方案代价估计值的**最佳优先搜索**
- **A**算法对评价函数中的启发函数未做出任何规定, **不能**评价搜索结果的**优劣**
- **A***算法评估函数 $f^*(n)$:

$$f^*(n) = g^*(n) + h^*(n)$$

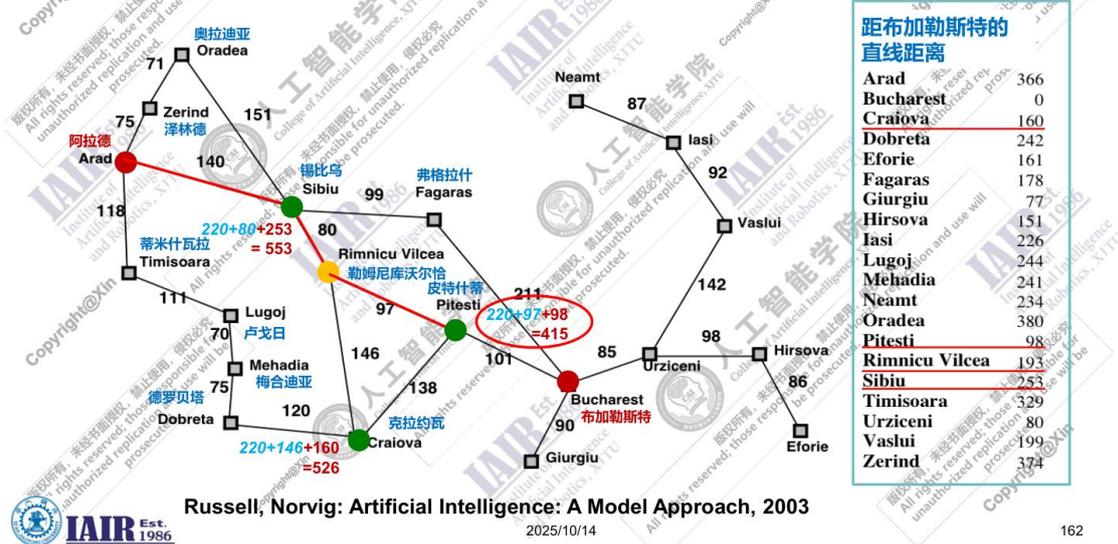
其中, $g^*(n)$: 从起始点到节点n的**路径最低代价**

$h^*(n)$: 从节点n到目标节点的**最低代价路径的代价**

$f^*(n)$: 从起始点出发、经过节点n、到达目标节点的**最佳路径的估计总代价**

- 启发函数 $h(n)$ 满足某些条件确保A*搜索的**完备性**、**最优性**:
 - 1) $g(n)$ 是对 $g^*(n)$ 的估计, 且 $g(n) > 0$
 - 2) $h(n)$ 是 $h^*(n)$ 的下界, 即对于任意节点均有 $h(n) \leq h^*(n)$

罗马尼亚旅行问题——A*搜索

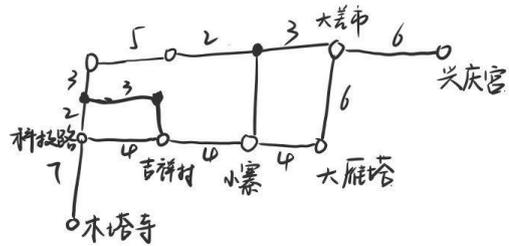


2024 年真题

(选择) 盲目搜索包含: 深度优先搜索和广度优先搜索

(简答) 贪婪搜索和 A*搜索

三. 贪婪搜索及 A*搜索 (兴庆宫版)



(记不住地名 emmm)

每个点有一个启发函数 $f(n)$

每条 edge 有一个长度 l .

图并不复杂, 可以参考 ppt 上类似题目的求解过程