西安电子科技大学 XIDIAN UNIVERSITY

人工智能概论

刘若辰
西安电子科技大学 人工智能学院

第二章 状态空间表示以及问题求解

- 2.1 问题求解与状态空间表示法
- 2.2 图搜索策略
- 2.3 盲目式搜索
- 2.4 启发式搜索

2.3 盲目式搜索

1. 宽度优先搜索
2. 深度优先搜索
3. 等代价搜索

盲目式搜索

➢ 概念

- 按预定的控制策略进行搜索，在搜索过程中获得的中间信息不用来改进控制策略（没有启发信息的一种搜索形式）。
- 一般只适用于求解比较简单的问题。

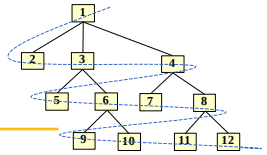
➢ 种类

- 宽度优先
- 深度优先
- 等代价搜索

宽度优先搜索

➢ **宽度/广度优先搜索基本思想：**

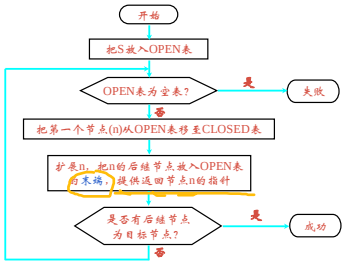
- 首先扩展根节点；
- 接着扩展根节点的所有后继节点；
- 然后再扩展后继节点的后继，依此类推；
- 在第n层节点还没有全部搜索完之前，不进入第n+1层节点的搜索。



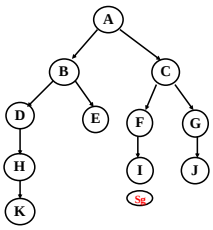
先生成的节点先扩展！

宽度优先搜索

宽度优先搜索流程图

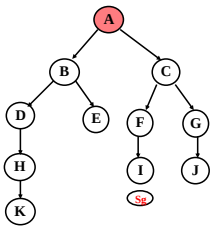


宽度优先搜索



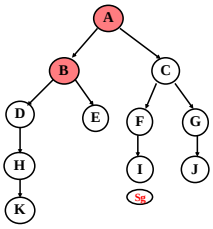
Open 表	Close 表
{A}	{}

宽度优先搜索



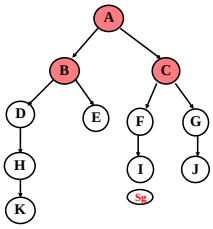
Open 表	Close 表
{A}	{}
{B,C}	{A}

宽度优先搜索



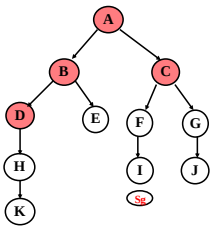
Open 表	Close 表
{A}	{}
{B,C}	{A}
{C,D,E}	{A,B}

宽度优先搜索



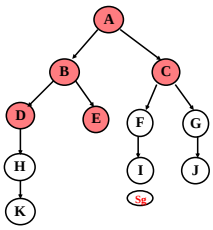
Open 表	Close 表
{A}	{}
{B,C}	{A}
{C,D,E}	{A,B}
{D,E,F,G}	{A,B,C}

宽度优先搜索

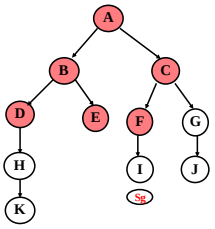


Open 表	Close 表
{A}	{}
{B,C}	{A}
{C,D,E}	{A,B}
{D,E,F,G}	{A,B,C}
{E,F,G,H}	{A,B,C,D}

宽度优先搜索



Open 表	Close 表
{A}	{}
{B,C}	{A}
{C,D,E}	{A,B}
{D,E,F,G}	{A,B,C}
{E,F,G,H}	{A,B,C,D}
{F,G,H}	{A,B,C,D,E}

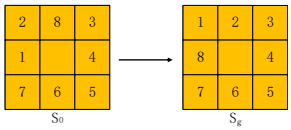


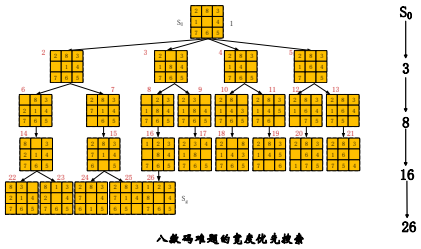
Open 表	Close 表
{A}	{}
{B,C}	{A}
{C,D,E}	{A,B}
{D,E,F,G}	{A,B,C}
{E,F,G,H}	{A,B,C,D}
{F,G,H}	{A,B,C,D,E}
{G,H,I}	{A,B,C,D,E,F}

停机：n放进close表中，而且子节点中包含目标节点

➢ **八数码难题**

在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。



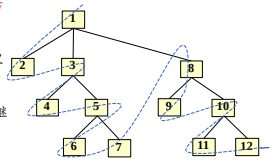


八数码难题的宽度优先搜索

- 属于图搜索方法：要求新扩展的节点排在OPEN表的末端。
- 当问题有解时，一定能找到解
- 方法与问题无关，具有通用性
- 效率较低

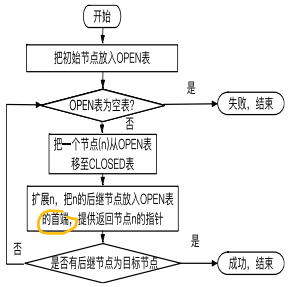
➢ **深度优先搜索基本思想：**

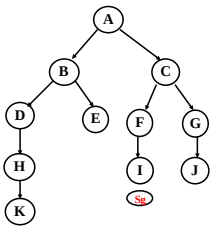
- 总是扩展搜索树的当前扩展分支中最深的节点；
- 搜索直接伸展到搜索树的最深层，直到那里的节点没有后继节点；
- 然后搜索算法回退到下一个还有未扩展后继节点的上层节点继续扩展。



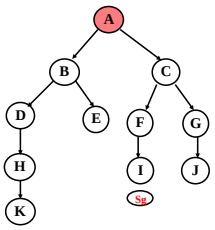
先扩展最新产生的(即最深的)节点的！

深度优先搜索流程图



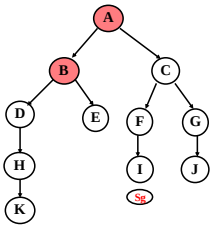


Open 表	Close 表
{A}	{}



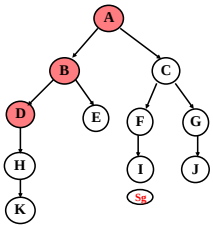
Open 表	Close 表
{A}	{}
{B,C}	{A}

深度优先搜索



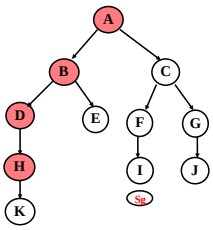
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}

深度优先搜索



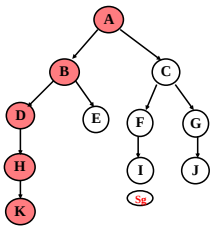
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}

深度优先搜索



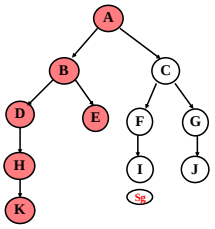
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}
{K,E,C}	{A,B,D,H}

深度优先搜索



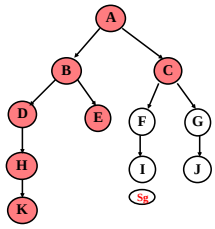
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}
{K,E,C}	{A,B,D,H}
{E,C}	{A,B,D,H,K}

深度优先搜索



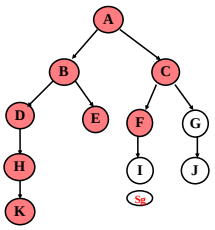
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}
{K,E,C}	{A,B,D,H}
{E,C}	{A,B,D,H,K}
{C}	{A,B,D,H,K,E}

深度优先搜索



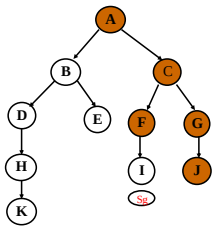
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}
{K,E,C}	{A,B,D,H}
{E,C}	{A,B,D,H,K}
{C}	{A,B,D,H,K,E}
{F,G}	{A,B,D,H,K,E,C}

深度优先搜索



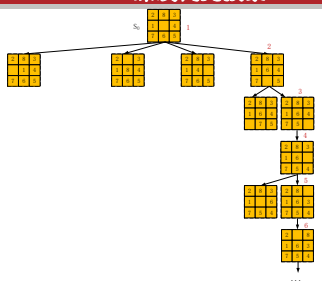
Open 表	Close 表
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{H,E,C}	{A,B,D}
{K,E,C}	{A,B,D,H}
{E,C}	{A,B,D,H,K}
{C}	{A,B,D,H,K,E}
{F,G}	{A,B,D,H,K,E,C}
{I,G}	{A,B,D,H,K,E,C,F}

深度优先搜索



Open 表	Close 表
{A}	{}
{C,B}	{A}
{G,F,B}	{A,C}
{J,F,B}	{A,C,G}
{F,B}	{A,C,G,J}
{I,B}	{A,C,G,J,F}

深度优先搜索



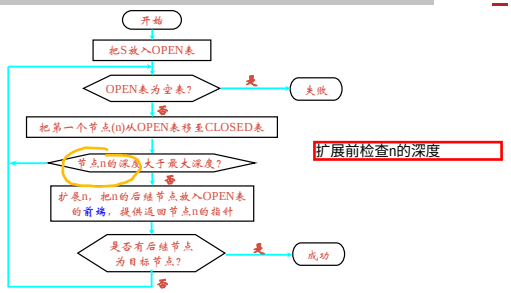
深度优先搜索

- 防止搜索过程沿着无益的路径扩展下去，往往给出一个节点扩展最大深度——深度界限。

- 定义-节点的深度:
  - (1) 起始节点的深度为0。
  - (2) 任何其他节点的深度等于其父辈节点的深度加1。

深度优先搜索

有界深度优先搜索流程图

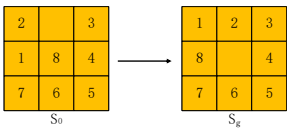


扩展前检查n的深度

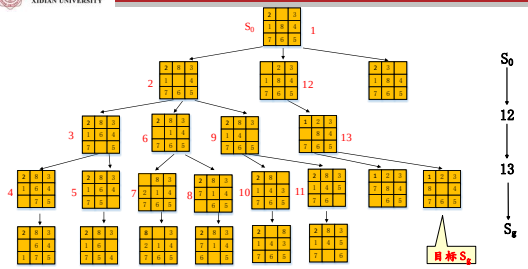
深度优先搜索

- **八数码难题**

在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。



深度优先搜索



八数码难题的深度优先搜索（深度界限：4）

深度优先搜索

- 一般不能保证找到最优解
- 当深度限制不合理时，可能找不到解，可以将算法改为可变深度限制
- 最坏情况时，搜索空间等同于穷举

等代价搜索

- **等代价搜索基本思想：**
  - 等代价搜索是宽度优先搜索的一种推广，不是沿着等长度路径断层进行扩展，而是沿着等代价路径断层进行扩展。
  - 状态图中每条连接弧线上的有关代价，表示时间、距离等花费。
  - 节点代价的定义：

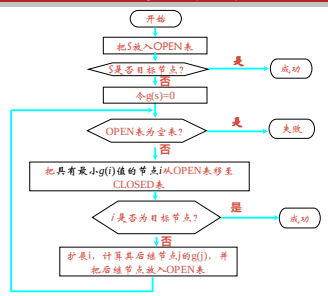
    $g(n)$：表示从初始节点 S_0 到节点 n 的代价；

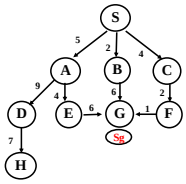
    $c(n_1, n_2)$：表示从父节点 n_1 到其子节点 n_2 的代价；

    $g(n_2) = g(n_1) + c(n_1, n_2)$

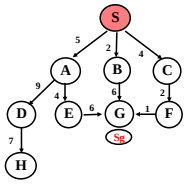
等代价搜索

等代价搜索流程图

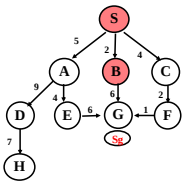




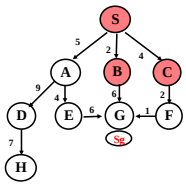
Open 表	Close 表
{S:0}	{}



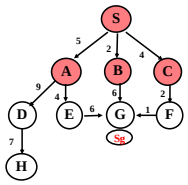
Open 表	Close 表
{S:0}	{}
{B:2,C:4,A:5}	{S:0}



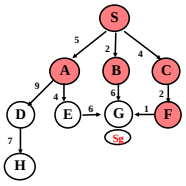
Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G:(2+6)}	{S:0,B:2}



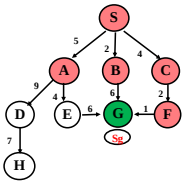
Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G:(2+6)}	{S:0,B:2}
{A:5,F:(4+2),G:(2+6)}	{S:0,B:2,C:4}



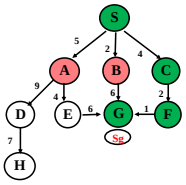
Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G: (2+6)}	{S:0,B:2}
{A:5,F: (4+2),G: (2+6)}	{S:0,B:2,C:4}
{F: (4+2),G: (2+6),E: (5+4),D:(5+9)}	{S:0,B:2,C:4, A:5}



Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G:(2+6)}	{S:0,B:2}
{A:5,F:(4+2),G:(2+6)}	{S:0,B:2,C:4}
{F: (4+2),G: (2+6),E: (5+4),D:(5+9)}	{S:0,B:2,C:4, A:5}
{G: (4+2+1),G: (2+6),E(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F: (4+2)}



Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G:(2+6)}	{S:0,B:2}
{A:5,F:(4+2),G:(2+6)}	{S:0,B:2,C:4}
{F(4+2),G(2+6),E(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5}
{G: (4+2+1),G: (2+6),E: (5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F: (4+2)}
{E: (5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F: (4+2), G: (4+2+1),}



注意删掉某个节点，修改指针！

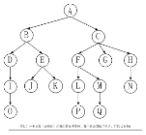
Open 表	Close 表
{S(0)}	{}
{B:2,C:4,A:5}	{S:0}
{C:4,A:5,G:(2+6)}	{S:0,B:2}
{A:5,F:(4+2),G:(2+6)}	{S:0,B:2,C:4}
{F:(4+2),G:(2+6),E:(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5}
{G:(4+2+1),G:(2+6),E:(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F(4+2)}
{G:(4+2+1),E:(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F:(4+2)}
{E:(5+4),D:(5+9)}	{S:0,B:2,C:4, A:5, F:(4+2), G:(4+2+1)}

盲目式搜索小结

- 宽度优先搜索是按“层”进行搜索的，先进入OPEN表的节点先被考察；
- 深度优先搜索是沿着纵深方向进行搜索的，后进入OPEN表的节点先被考察；
- 等代价搜索首先扩展最小代价节点；

按预先设置的策略或已付出的代价进行搜索，没有利用搜索过程中的信息指导搜索！

练习一



- 使用宽度/深度优先算法对图中的树进行搜索的如图所示，目标节点是标有Q的那个节点。写出其搜索过程中的OPEN表和CLOSED表.

练习二

(1)请根据深度优先搜索策略列出图中树的节点的访问序列(在所有情况中都选择最左分支优先访问)。
(2)请根据深度优先搜索策略搜索对从初始状态迭代至搜索结束的过程中OPEN表和CLOSED表所存储的内容