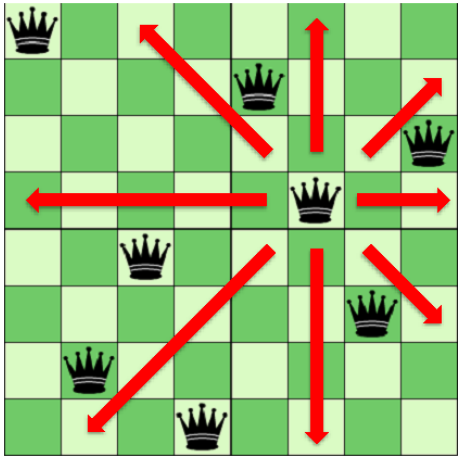
人工智能基础

第三章 经典人工智能



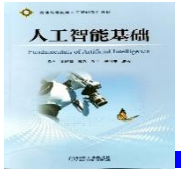
经典人工智能-主要内容

- 知识表示方法
- 搜索技术
- 知识推理
- 不确定性推理

知识表示方法-知识、信息与数据三者的关系

知识表示方法-知识的特性

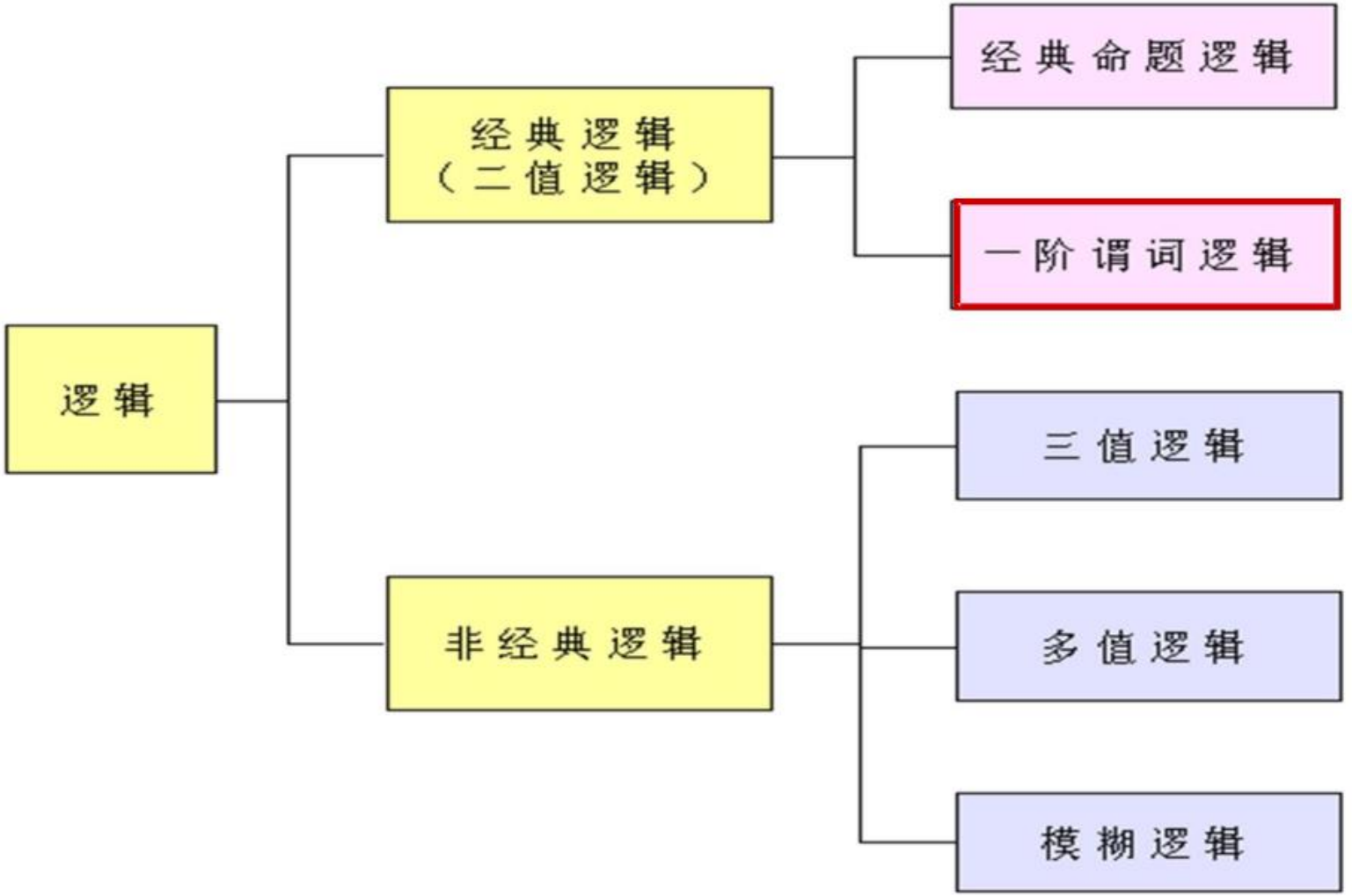
- 相对正确性
- 不确定性
- 可表示性
- 可利用性

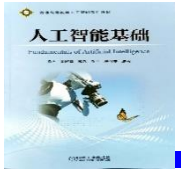


知识表示方法-知识分类

- 知识分类
  - 按作用范围分
    - 通用性知识
    - 领域性知识
  - 按作用与表示分
    - 事实性知识
    - 规则性知识
    - 控制性知识
    - 元知识
  - 按确定性分
    - 确定知识
    - 不确定知识
  - 按人类的思维及认识方法分
    - 逻辑性知识
    - 形象性知识

知识表示的方法-逻辑表示法





知识表示的方法-一阶谓词逻辑表示法-初步印象

例1：定义谓词IsStudent（x），事实性知识“小明是学生，小红也是学生”的谓词逻辑表示(谓词公式)

IsStudent(Xiaoming) ∧ IsStudent(Xiaohong)

例2：规则性知识“如果a，则b”用蕴含式表示：

$a \rightarrow b$

- 自然性
- 精确性
- 易实现

具体细节在知识推理中介绍

知识表示的方法-产生式表示法

- 可表示知识的种类：
  - ✓表达具有因果关系的知识，包括事实性知识和规则性知识
  - ✓根据知识是确定性的还是不确定性的分别进行表示
- 产生式的基本形式：

  $P \rightarrow Q$ 或 IF P THEN Q

  - ✓P是产生式前提，Q是一组结论或操作
- 产生式与谓词逻辑中蕴涵式的区别：
  - ✓产生式可表示精确与不精确知识，蕴涵式只表示精确知识
  - ✓产生式没有真值，蕴涵式有真值
  - ✓产生式可以不精确匹配，蕴涵式需要精确匹配

知识表示的方法-产生式表示法

(1) 确定性规则知识

P→Q 或 IF P THEN Q

例如，所有动物都会死，甲是动物，所以甲也会死

产生式表达的知识：所有动物都会死∧甲是动物→甲会死

(2) 不确定性规则知识

P→Q（置信度）或IF P THEN Q（置信度）

例如，如果乌云密布，那么将要下雨的置信度是80%

产生式表达式：乌云密布→ 将要下雨 （0.8）

知识表示的方法-产生式表示法

(3) 确定性事实性知识

三元组（对象，属性，值）

三元组（关系，对象1，对象2）

例如“老李年龄是55岁”，可以写成（李，年龄，55）

“老李、老王是朋友”则可写成（朋友，李，王）

(4) 不确定性事实性知识

四元组（对象，属性，值，可信度）

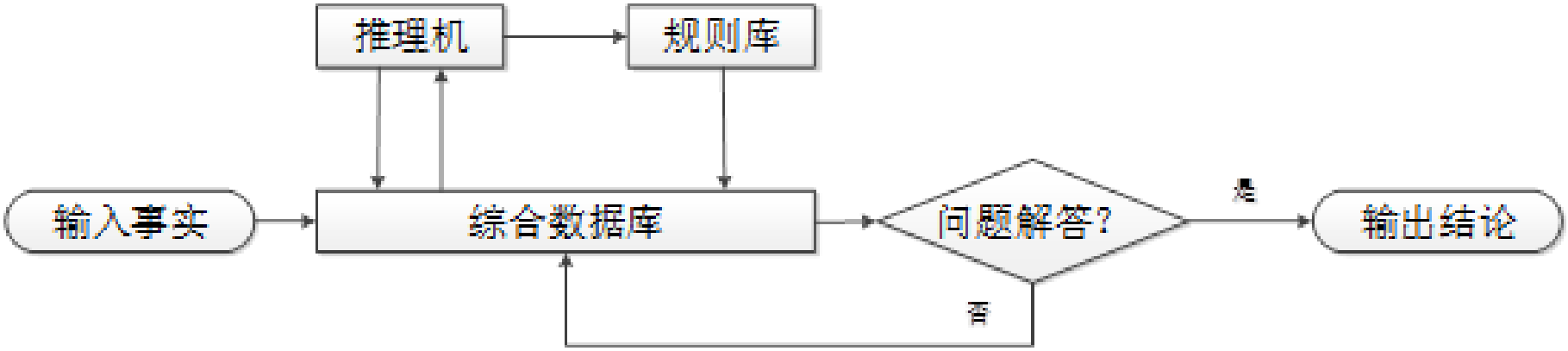
四元组（关系，对象1，对象2 ，可信度）

例如 “老李年龄可能是30岁，可能性70%” ，可以用四元组（李，年龄，30，0.7）来表示

知识表示的方法-产生式表示法

(5) 产生式系统

- ✓以产生式表示知识的系统称作产生式系统
- ✓由规则库、推理机和综合数据库三部分组成



知识表示的方法-产生式表示法

规则库：规则是以产生式表示的，规则集蕴涵着将问题从初始状态转换成目标状态的那些变换规则。

综合数据库：存放输入的事实、外部数据库输入的事实以及推理的中间结果（事实）和最后结果。

推理机：控制和协调规则库与数据库运行，包含推理方式和控制策略，控制策略就是确定如何选择或应用规则，包括匹配、冲突解决和操作步骤，推理方式有正向推理、反向推理和双向推理。

知识表示的方法-产生式表示法--一个实例

- 动物分类问题的产生式系统描述及其求解
- 规则库

r1：若某动物有奶，则它是哺乳动物。

r2：若某动物有毛发，则它是哺乳动物。

r3：若某动物有羽毛，则它是鸟。

r4：若某动物会飞且生蛋，则它是鸟。

r5：若某动物是哺乳动物且有爪且有犬齿且目盯前方，则它是食肉动物。

r6：若某动物是哺乳动物且吃肉，则它是食肉动物。

r7：若某动物是哺乳动物且有蹄，则它是有蹄动物。

知识表示的方法-产生式表示法

- 规则库（续）

r8：若某动物是有蹄动物且反刍食物，则它是偶蹄动物。

r9：若某动物是食肉动物且黄褐色且有黑色条纹，则它是老虎。

r10：若某动物是食肉动物且黄褐色且有黑色斑点，则它是金钱豹。

r11：若某动物是有蹄动物且长腿且长脖子且黄褐色且有暗斑点，则它是长颈鹿。

r12：若某动物是有蹄动物且白色且有黑色条纹，则它是斑马。

r13：若某动物是鸟且不会飞且长腿且长脖子且黑白色，则它是驼鸟。

r14：若某动物是鸟且不会飞且会游泳且黑白色，则它是企鹅。

r15：若某动物是鸟且善飞且不怕风浪，则它是海燕。

知识表示的方法-产生式表示法

- 综合数据库

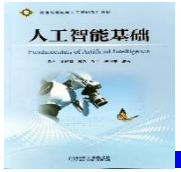
  f1：某动物有毛发。

  f2：吃肉。

  f3：黄褐色。

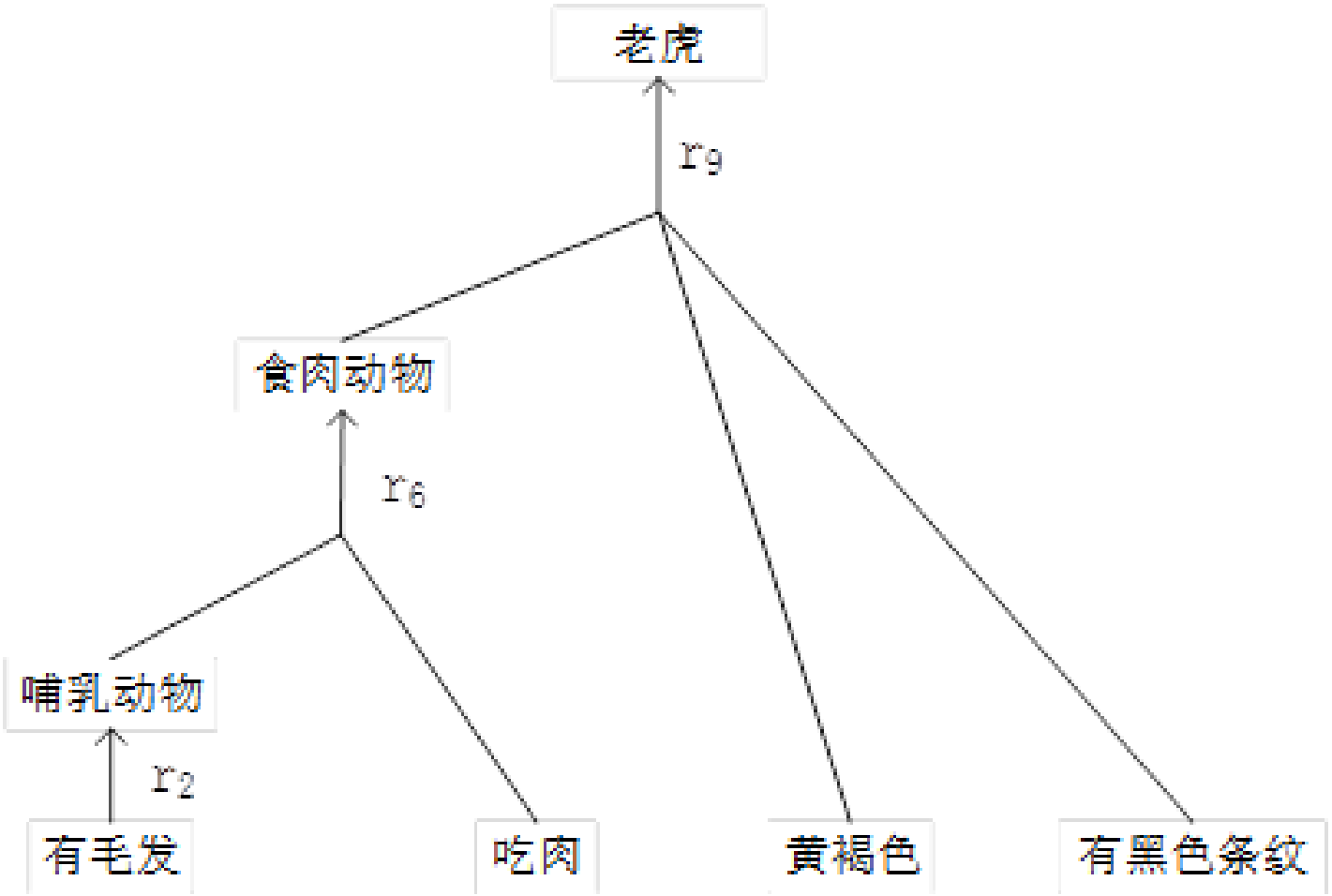
  f4：有黑色条纹。

- 推理目标：该动物是什么？



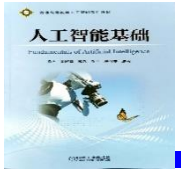
知识表示的方法-产生式表示法

- 解：该动物分类问题的正向推理树



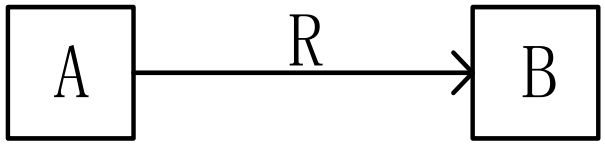
知识表示的方法-语义网络表示法

语义网络是1968年J.R.Quillian在研究人类联想记忆时提出的心理学模型。1972年Simon将语义网络用于自然语言理解系统。

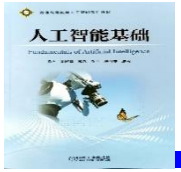


知识表示的方法-语义网络表示法

- ✓ 通过概念及其语义关系来表示知识的一种带标注的有向网络图
- ✓ 最简单的语义网络可由一个三元组表示：（结点A，弧R，结点B）或由有向图表示



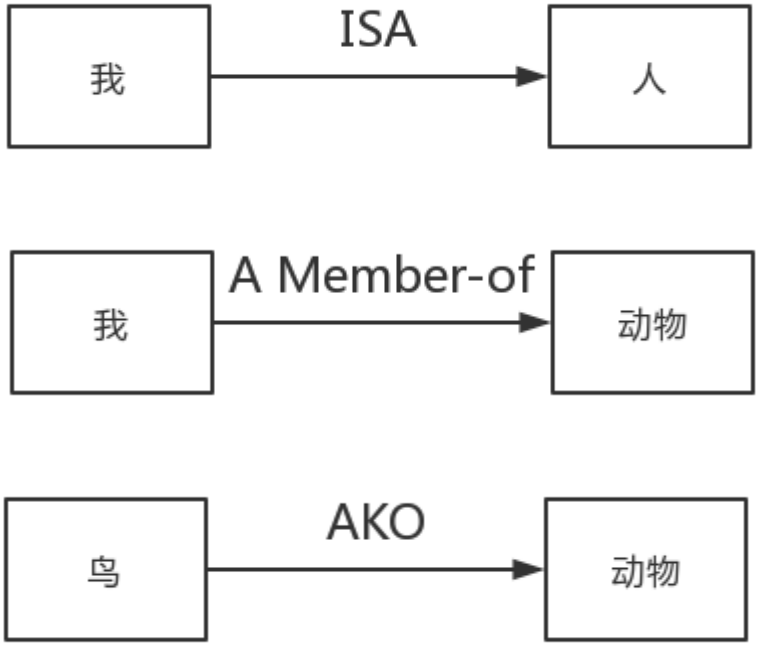
- ✓ 结点表示概念、事物、事件、情况等
- ✓ 弧有方向、有标注的。方向体现主次，结点A为主，结点B为辅
- ✓ 弧上的标注表示结点A的属性或结点A和结点B之间的关系。
- ✓ 当将多个基本网元用相应的语义联系关联起来即可构成语义网络。

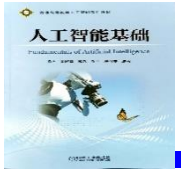


知识表示的方法-语义网络表示法

语义网络的基本语义关系：

- ✓ 类属关系：指不同事物间的分类关系、成员关系或实例关系，常用有三种：ISA：(Is a)、AMO：(A-Member-Of)、AKO：(A-Kind-Of)，具有继承性。

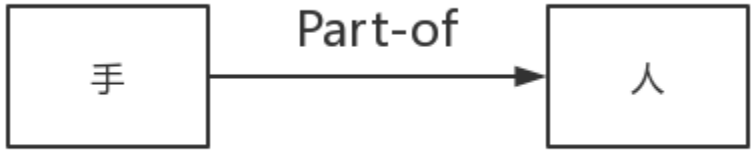


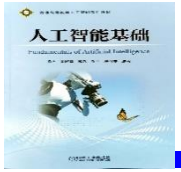


知识表示的方法-语义网络表示法

语义网络的基本语义关系：

- ✓ 包含关系：又叫聚类关系，是指部分与整体之间的关系。它和类属关系最主要的区别是包含关系一般不具备属性的继承性。常用的包含关系是：Part-of ，含义为“是一部分”

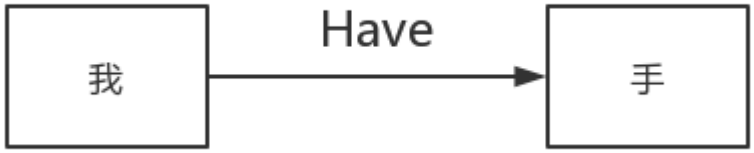




知识表示的方法-语义网络表示法

语义网络的基本语义关系：

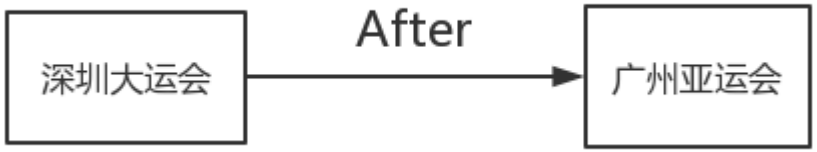
- ✓ 占有关系：占有关系是事物或属性之间的“具有”关系。常用的占有关系是Have

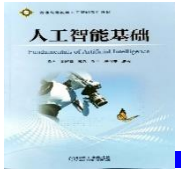


知识表示的方法-语义网络表示法

语义网络的基本语义关系：

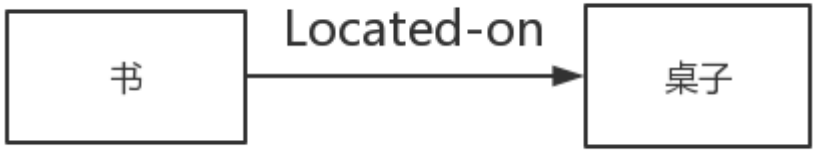
✓ 时间关系：时间关系是指不同事件在其发生时间方面的先后次序关系，节点间的属性不具有继承性。常用的时间关系有三种：Before，After，During

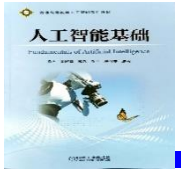




知识表示的方法-语义网络表示法

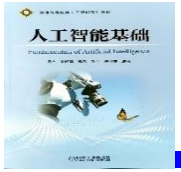
- ✓ 位置关系：位置关系是指不同事物在位置方面的空间关系，常用有Located-on（at，under，inside，outside）





知识表示的方法-语义网络表示法

- ✓ 因果关系：因果关系是用于表示规则性知识，常用If-then联系表示两个节点间的因果关系
- ✓ 相近关系：相近关系指不同事物的某些特征的相似和接近。常用的相近关系有：Similar-to，Near-to
- ✓ 推论关系：推论关系是指从一个概念推出另一个概念的语义关系
- ✓ 组成关系：组成关系是一种一对多联系，用于表示某一事物由其他一些事物构成，常用Composed of联系表示
- ✓ 属性关系：属性关系用于表示一个节点是另一个节点的属性关系



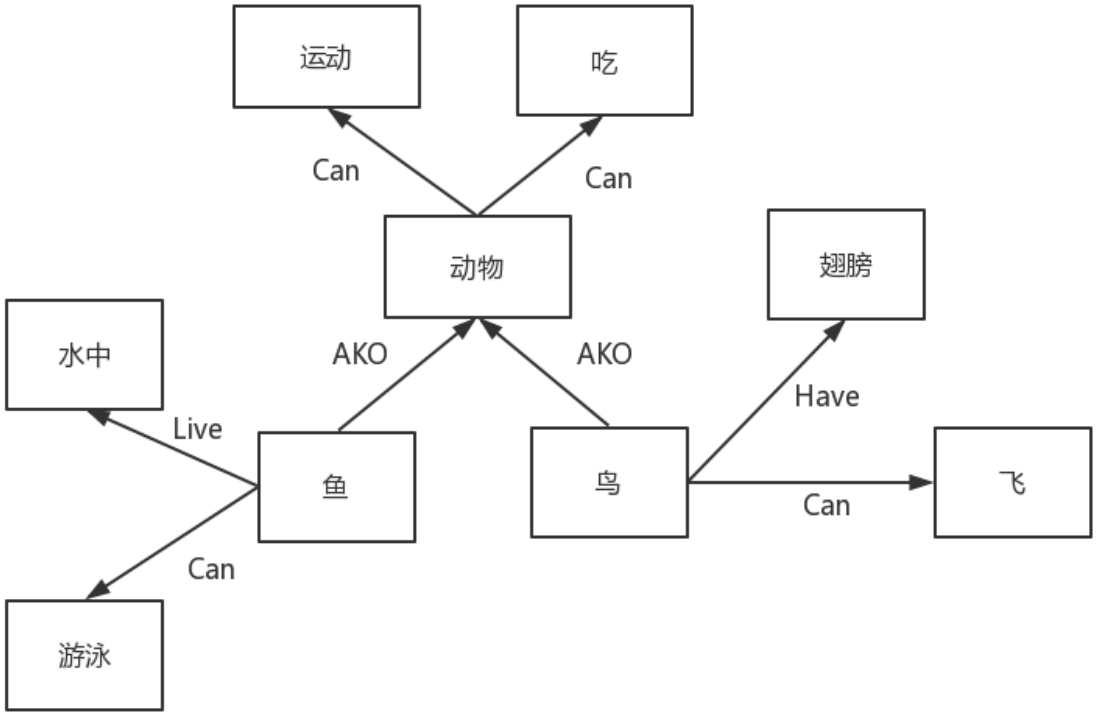
知识表示的方法-语义网络表示法-实例（网络及推理）

1、动物能吃、能运动

2、鸟是一种动物，鸟有翅膀、会飞

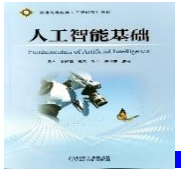
3、鱼是一种动物，鱼生活在水中、会游泳

利用继承与匹配求出鸟（待求解节点）可以（Can）进行的活动



搜索技术

- 盲目搜索
  - ✓状态空间图的一般搜索算法
  - ✓宽度优先搜索策略
  - ✓深度优先搜索策略
  - ✓代价树的宽度优先搜索策略
  - ✓代价树的深度优先搜索策略
- 启发式搜索
  - ✓局部最佳优先搜索
  - ✓全局最佳优先搜索
  - ✓A*启发式搜索算法



搜索技术-搜索的概念及种类

- **搜索：**

找到从初始事实到问题最终答案的一条推理路线，而且是时间和空间复杂度最小的求解路线

- 搜索种类：
  - ✓ **盲目搜索：**

    系统根据事先确定好的某种固定排序（依次或随机）调用规则
  - ✓ **启发式搜索：**

    考虑问题领域可应用的知识，根据具体情况动态地确定规则的排序，优先调用较合适的规则使用

搜索技术-盲目搜索-状态空间表示法

- 状态空间表示法：用来表示问题及其搜索过程的一种方法
- 状态空间由一个四元组构成

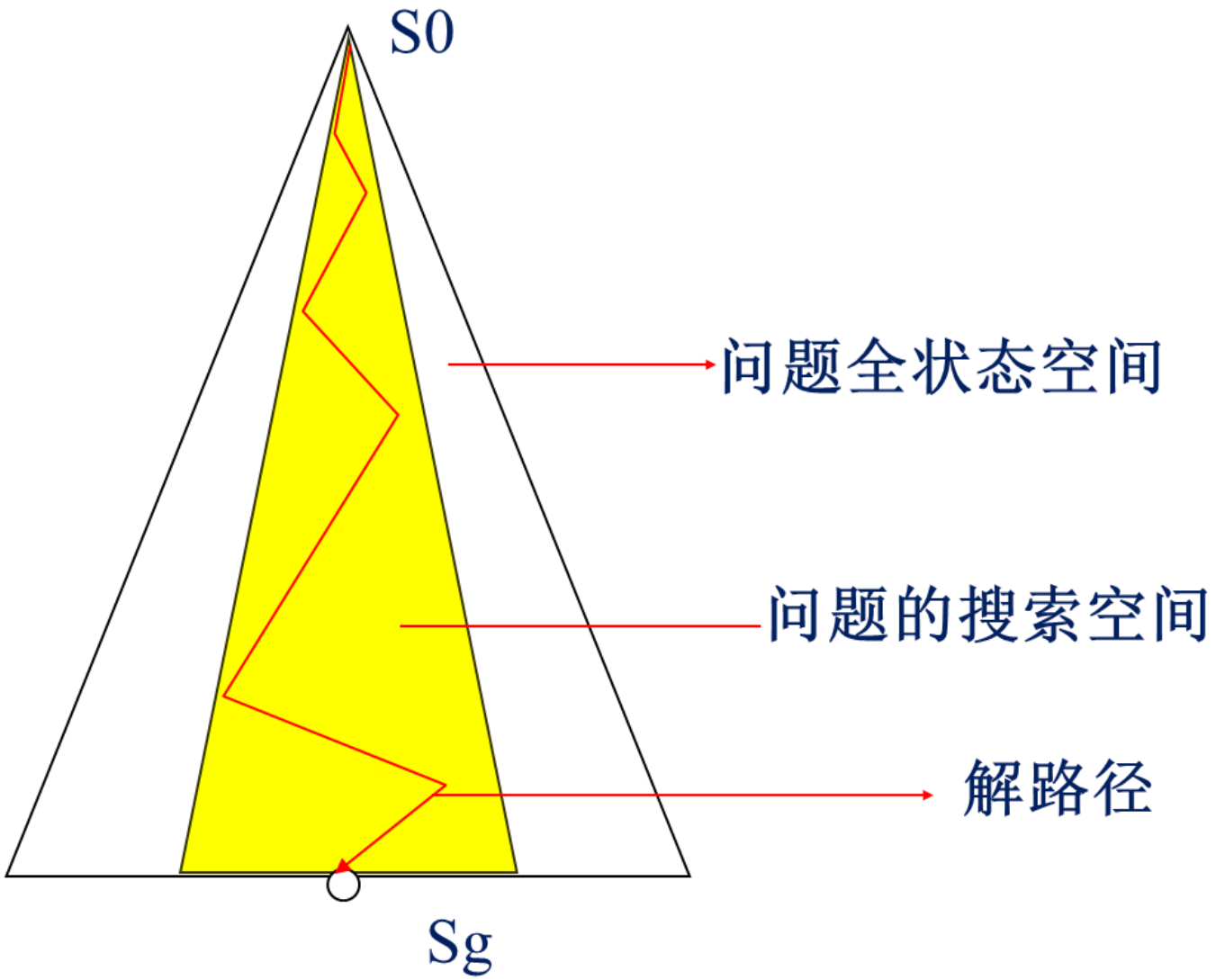
（1）状态，描述问题求解过程中不同时刻状况的数据结构。

（2）算符：使问题由一个状态变为另一个状态的操作。

（3）状态空间：一个问题的全部状态及一切可用算符构成的集合。一般包括3部分（初始状态集合S，算符集合F，目标状态集合G）　(S, F, G)

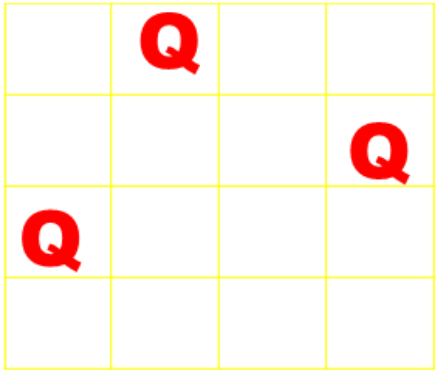
（4）问题的求解：从S出发经过一系列的算符运算，到达目标状态。由初始状态到目标状态所用算符的序列构成了问题的一个求解。

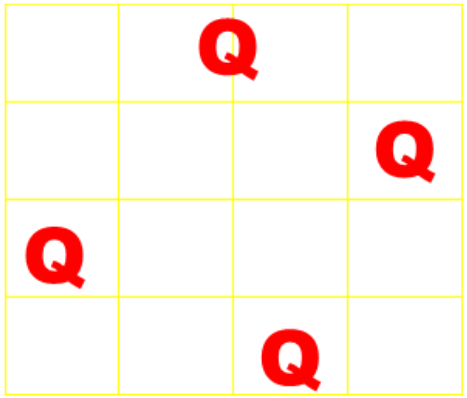
搜索技术-盲目搜索-状态空间表示法



搜索技术-盲目搜索-状态空间图

- 状态空间图：
  - ✓ 把状态空间的问题求解过程用图的形式表示出来
  - ✓ 节点代表状态，弧代表算符
- 四皇后问题：每行、每列和对角线上只允许出现一枚棋子，即棋子之间不许相互攻击





((1,2) (2,4) (3,1)) → ((1,2) (2,4) (3,1) (4,3))

搜索技术-盲目搜索-状态空间搜索的几个概念

- 扩展：

  用合适的算符对某个结点进行操作，生成一组后继结点，扩展过程就是求后继结点的过程。

- 已扩展结点：

  已经求出了其后继结点的结点。

- 未扩展结点：

  尚未求出后继结点的结点。

- OPEN表：

  存放未扩展的结点，记录当前结点及其父结点。

- CLOSED表：

  存放已扩展结点，记录编号、当前结点及其父结点。

搜索技术-盲目搜索-状态空间搜索的几个概念

以八数码问题为例

OPEN表
(记录待扩展的节点)

节点号	父节点号

CLOSED表
(记录扩展过的节点)

编号	节点号	父节点号

搜索技术-盲目搜索-状态空间的一般搜索算法

一般搜索算法的描述：

① 建立一个只含有初始结点S0的搜索图G，把S0放入OPEN表中

② 建立CLOSED表，且置为空表

③ 判断OPEN表是否为空表，若为空，则问题无解，退出

④ 选择OPEN表中的第一个结点，把它从OPEN表移出，并放入CLOSED表中，将此结点记为结点n

⑤ 考察结点n是否为目标结点，若是，则问题有解，成功退出。问题的解就是沿着n到S0的路径得到。若不是转⑥

搜索技术-盲目搜索-状态空间的一般搜索算法

一般搜索算法的描述（续）：

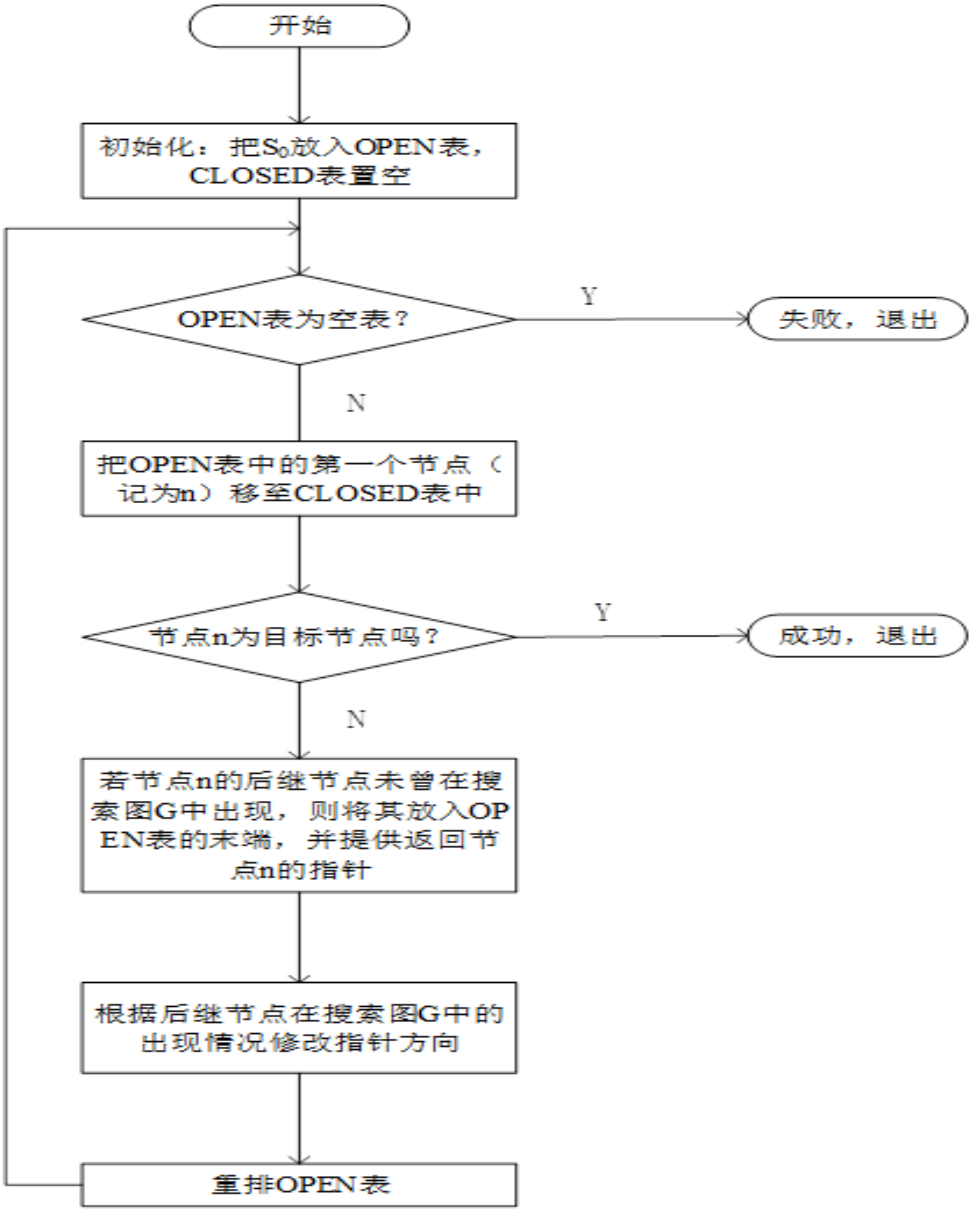
⑥ 扩展结点n生成一组不是n的祖先的后继结点，并将它们记为集合M，将M中的这些结点作为n的后继结点加入图G中

⑦ 对未在G中出现过的（OPEN和CLOSED表中未出现过的）集合M中的结点，设置一个指向父结点n的指针，并把这些结点放入OPEN表中；对于已在G中出现过的M中的结点，确定是否需要修改指向父结点的指针；对于已在G中出现过并已在closed表中的M中的结点，确定是否需要修改通向他们后继结点的指针。

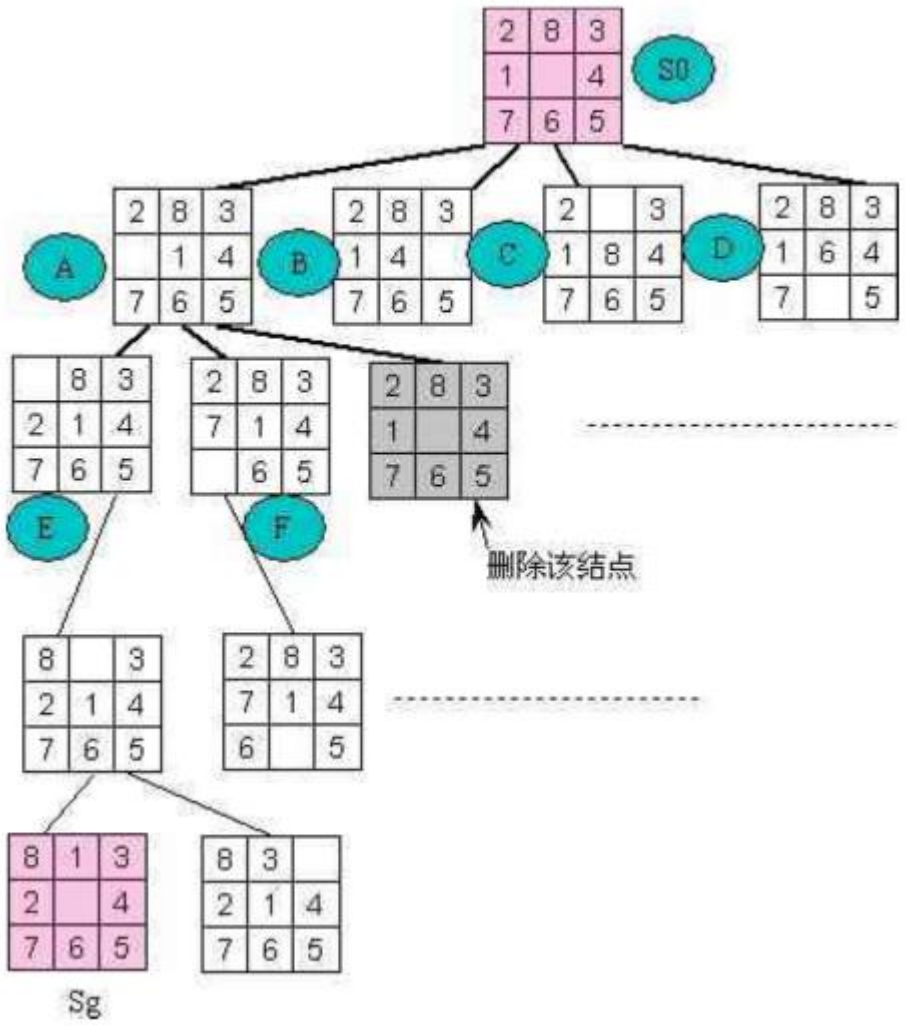
⑧ 按某一任意方式或某种策略重排OPEN表中结点的顺序

⑨ 转③

搜索技术-盲目搜索-状态空间搜索策略流程图

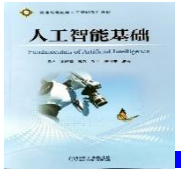


搜索技术-盲目搜索-状态空间搜索实例



节点号	父节点号
S_0	空
A	S_0
B	S_0
C	S_0
D	S_0
E	A
F	A
……	

编号	节点号	父节点号
0	S_0	空
1	A	S_0
2	B	S_0
……		



搜索技术-盲目搜索-宽度优先搜索策略

- 定义

  如果搜索是以接近起始结点的程度依次扩展结点的，那么这种搜索就叫做宽度优先搜索

- 性质

  - ✓ 当问题有解时，一定能找到解
  - ✓ 当问题为单位消耗值，且问题有解时，一定能找到最优解
  - ✓ 算法与问题无关，具有通用性
  - ✓ 时间效率和空间效率都比较低

搜索技术-盲目搜索-宽度优先搜索算法

① 把起始结点放到OPEN表中(如果该起始结点为一目标结点，则求得一个解答)。

② 如果OPEN是个空表，则没有解，失败退出；否则继续。

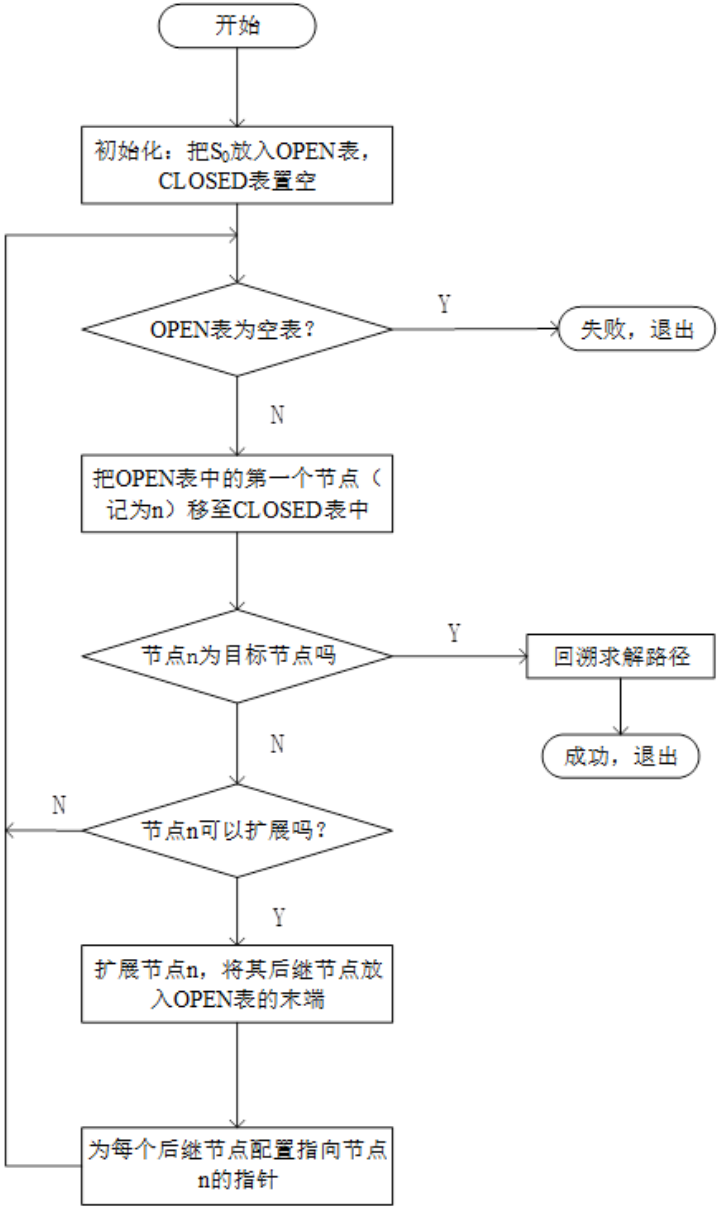
③ 把第一个结点(结点n)从OPEN表移出，并把它放入CLOSED的扩展结点表中。

④ 扩展结点n。如果没有后继结点，则转向上述第(2)步。

⑤ 把n的所有后继结点放到OPEN表末端，并提供从这些后继结点回到n的指针。

⑥ 如果n的任一个后继结点是个目标结点，则找到一个解答，成功退出；否则转向第(2)步。

搜索技术-盲目搜索-宽度优先算法流程图

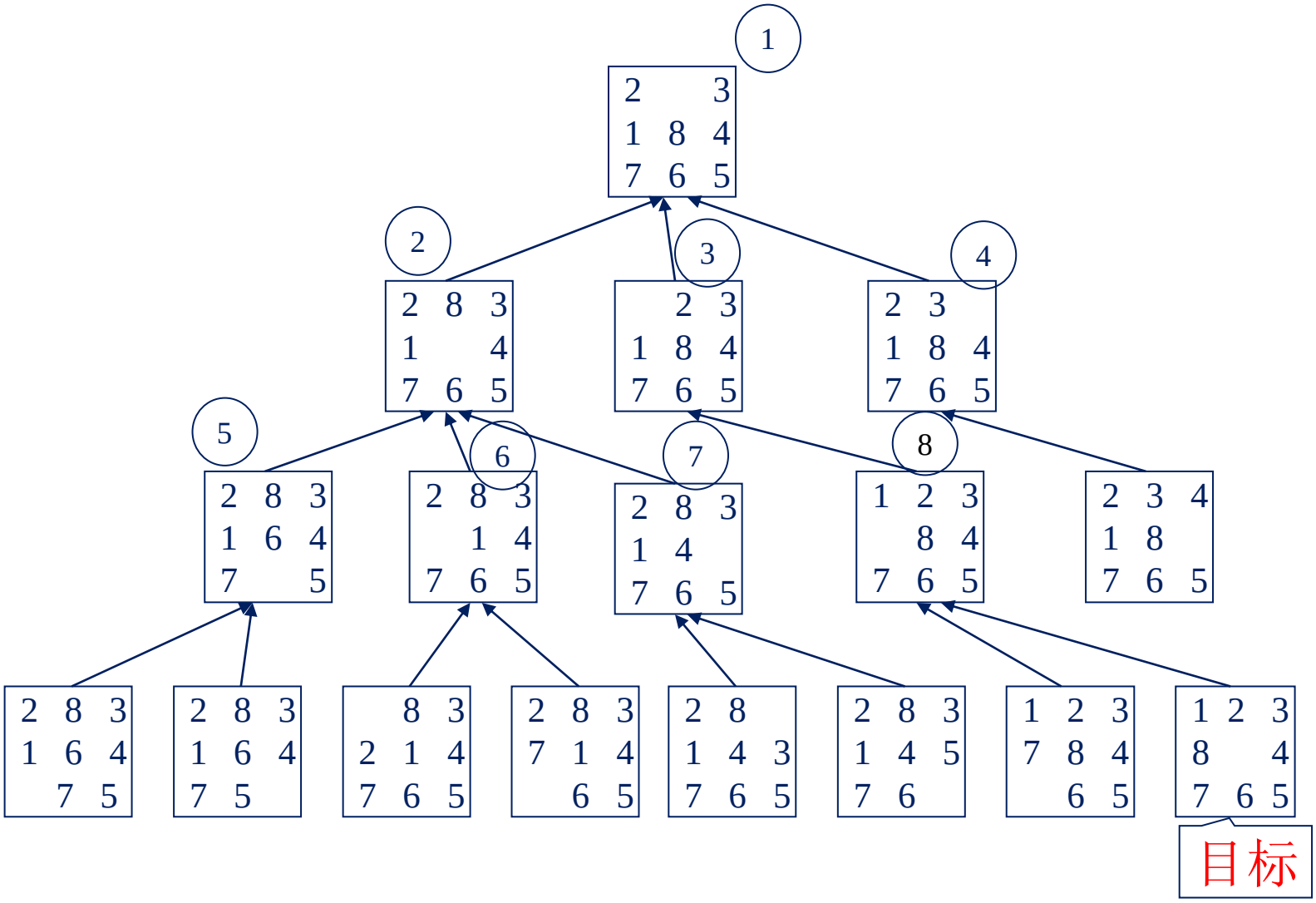


搜索技术-盲目搜索-宽度优先搜索实例

例：把宽度优先搜索应用于八数码难题时所生成的搜索树，这个问题就是要把初始棋局变为如下目标棋局的问题：

1	2	3
8		4
7	6	5

搜索技术-盲目搜索-宽度优先搜索实例



搜索技术-盲目搜索-深度优先搜索策略

- 定义

在此搜索中，首先扩展最新产生的(即最深的)结点。深度相等的结点可以任意排列

- 深度界限

为了避免考虑太长的路径(防止搜索过程沿着无益的路径扩展下去)，往往给出一个结点扩展的最大深度界限。任何结点如果达到了深度界限，那么都将把它们作为没有后继结点处

搜索技术-盲目搜索-深度优先搜索算法

(1) 把起始结点放到OPEN表中(如果该起始结点为一目标结点，则求得一个解答)。

(2) 如果OPEN是个空表，则没有解，失败退出；否则继续。

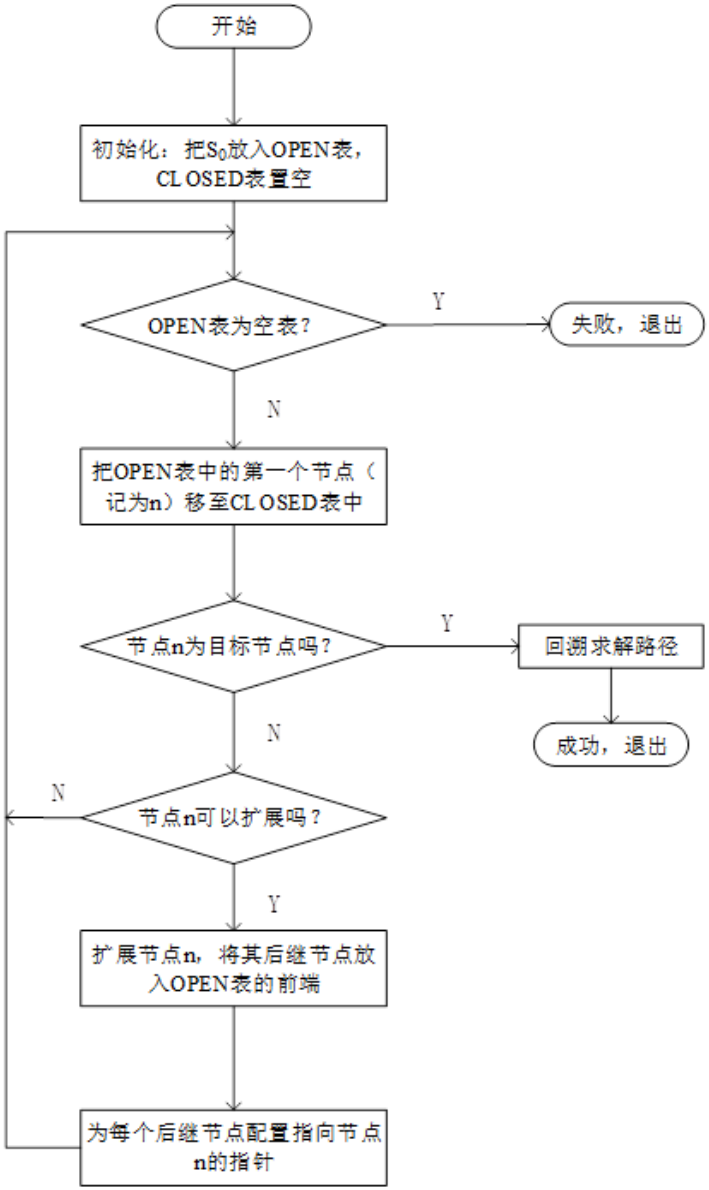
(3) 把第一个结点(结点n)从OPEN表移出，并把它放入CLOSED的扩展结点表中。

(4) 考察结点n是否为目标结点，若是，则找到问题的解，用回溯法求解路径，退出

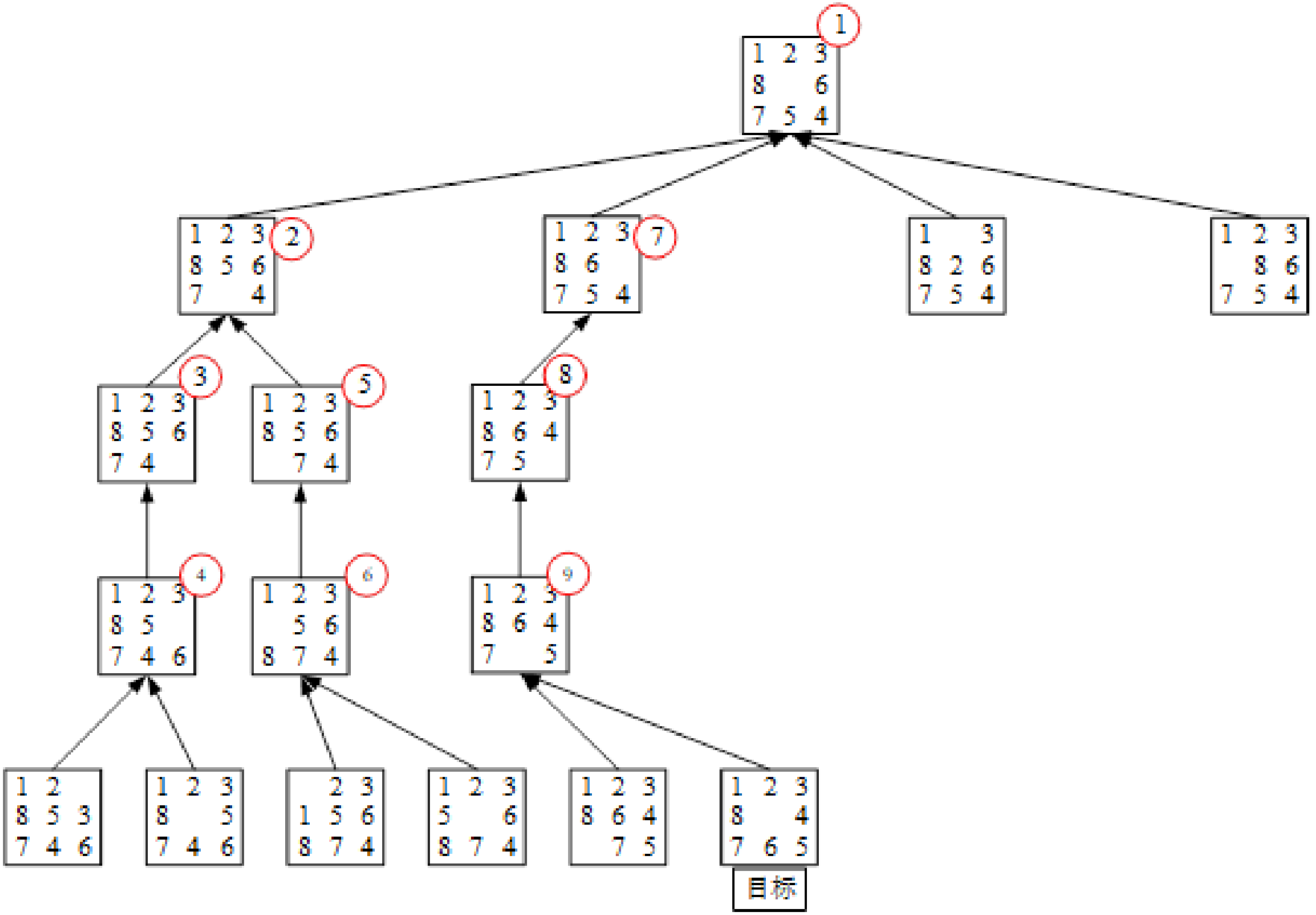
(5) 如果没有后继结点，则转向上述第(2)步。

(6) 扩展结点n，把n的所有后继结点放到OPEN表前端，并提供从这些后继结点回到n的指针。转向第(2)步。

搜索技术-盲目搜索-深度优先算法流程图



搜索技术-盲目搜索-深度优先搜索实例



搜索技术-盲目搜索–代价树的宽度优先搜索

1、定义

状态空间图中各结点之间有向边的代价是不同的，有向边上标有代价的搜索树成为代价搜索树

2、特点

每次从OPEN表中选择一个代价最小的结点，移入CLOSED表

3、连线代价 C(i,j):从结点i到其后继结点j的连线代价

路径代价 g(x):从初始结点到任意结点x的路径代价

$$g(j)=g(i)+C(i,j)$$

搜索技术-盲目搜索–代价树的宽度优先搜索算法

(1) 把起始结点放到OPEN表中，令g(S0)=0

(2) 如果OPEN是个空表，则没有解，失败退出；否则继续

(3) 把OPEN表中代价最小的结点（即排在最前端的结点n），移入CLOSED的扩展结点表中

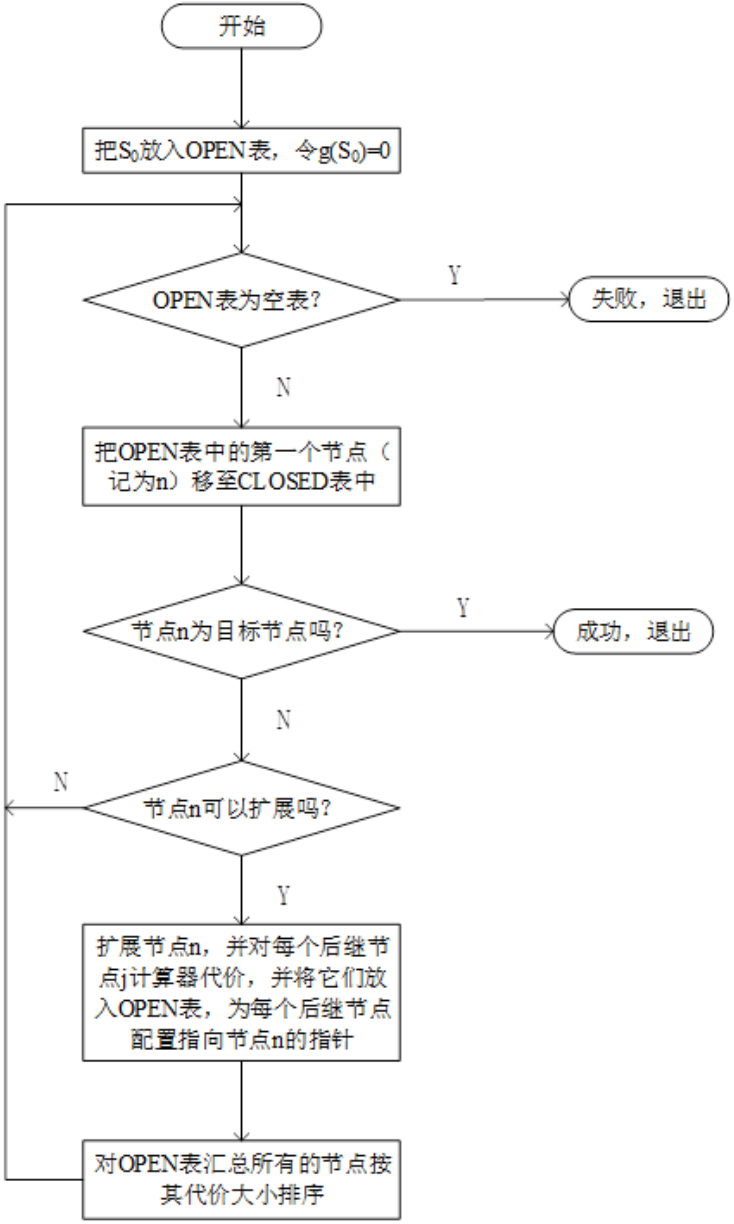
(4) 如果n是目标结点，问题得解，退出；否则继续

(5) 判断结点n是否可扩展。若否则转向第(2)步，若是则转向(6)

(6) 对结点n进行扩展，将他们的所有后继结点放到OPEN表中，并对每个后继结点j计算其总代价g(j)=g(i)+C(i,j),为每个后继结点指向n结点的指针，然后根据结点的代价大小对OPEN表中的所有结点进行从小到大的排序

(7) 转向第(2)步

搜索技术-盲目搜索-代价树的宽度优先搜索流程图



搜索技术-盲目搜索-代价树的深度优先搜索

- 代价树的宽度优先搜索

  每次从OPEN表中的全体结点中选择代价最小的结点移入CLOSED表进行扩展或判断。

- 代价树的深度优先搜索

  从刚刚扩展的结点的后继结点中选择一个代价最小的结点移入CLOSED表中，并进行扩展或判断。

搜索技术-盲目搜索-代价树的深度优先搜索算法

(1) 把起始结点放到OPEN表中，令g(S0)=0

(2) 如果OPEN是个空表，则没有解，失败退出；否则继续

(3) 把OPEN表中的第一个结点（代价最小的结点n），移入CLOSED表

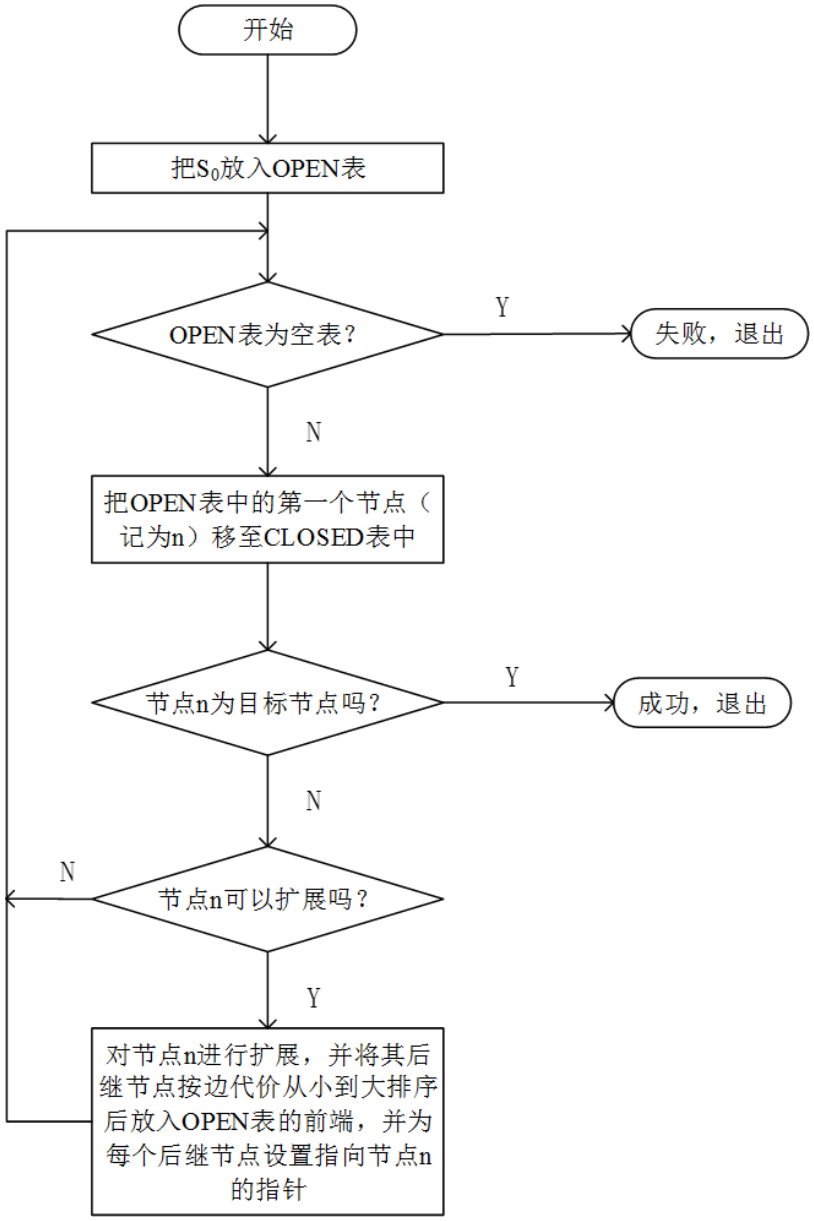
(4) 如果n是目标结点，问题得解，退出。否则继续

(5) 判断结点n是否可扩展。若否则转向第(2)步，若是则转向(6)

(6) 对结点n进行扩展，并将其后继结点按有向边代价（C(i,j)）从小到大排序后放到OPEN表前端，并为每个后继结点设置指向n结点的指针

(7) 转向第(2)步

搜索技术-盲目搜索-代价树深度优先搜索流程图



搜索技术-启发式搜索

- 盲目搜索的一个特点就是它们的搜索路线是事先决定好的，没有利用被求解问题的任何特性信息

- 能否找到一种方法，能够充分利用待求解问题的某些特性，以指导搜索朝着最有利于问题求解的方向发展，即在选择那些最有希望的结点加以扩展，那么搜索的效率就会大大提高

- 这种利用问题的自身特性信息来提高搜索效率的搜索策略，称为启发式搜索

搜索技术-启发式搜索-启发信息

- **启发信息**

  利用与问题有关的知识（即：启发信息）来引导搜索，达到减少搜索范围，降低问题复杂度的搜索过程称为启发式搜索方法

- **核心问题**

  启发信息应用，启发能力度量和如何获得启发信息

- **启发信息的强度**

  ✓强：降低搜索工作量，但可能导致找不到最优解。

  ✓弱：一般导致工作量加大，极限情况下变为盲目搜索，但可能可以找到最优解

搜索技术-启发式搜索-估价函数

- 定义一个估价函数f，对当前的搜索状态进行评估，找出一个“最有希望”的结点来扩展。
- 估价函数的格式：

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

f(n)：估价函数

g(n)：代价函数

初始结点到结点n已实际付出的代价

h(n)：启发函数

从结点n到目标结点的最优路径的估计代价

搜索技术-启发式搜索-局部最佳优先搜索

(1) 把起始结点放到OPEN表中，并计算估价函数f(S0)

(2) 如果OPEN是个空表，则没有解，失败退出；否则继续

(3) 把OPEN表中的第一个结点（股价函数最小的结点n），移入CLOSED表

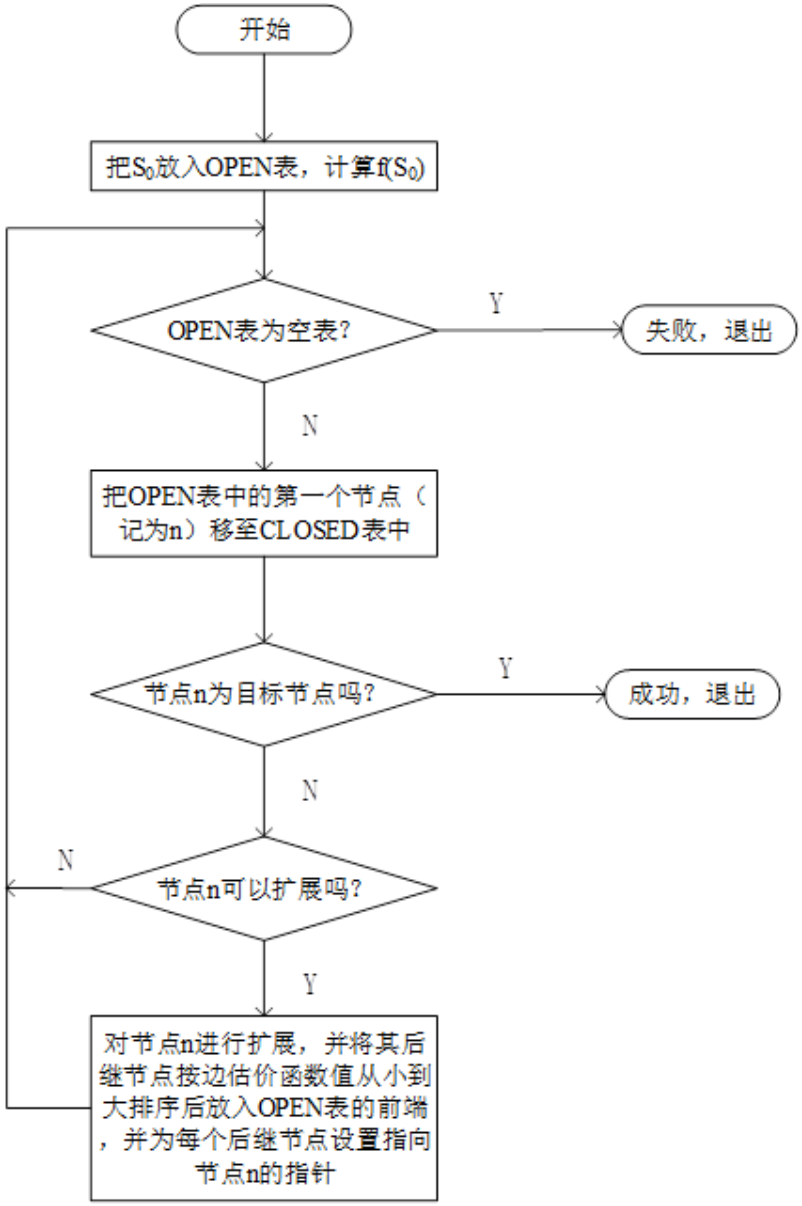
(4) 如果n是目标结点，问题得解，退出。否则继续

(5) 判断结点n是否可扩展。若否则转向第(2)步，若是则转向(6)

(6) 对结点n进行扩展，并对其所有后继结点计算估价函数f(n)的值，并按其值从小到大排序后放到OPEN表前端，并为每个后继结点设置指向n结点的指针

(7) 转向第(2)步

搜索技术-启发式搜索-局部最佳优先搜索流程图



搜索技术-启发式搜索-全局最佳优先搜索

(1) 把起始结点放到OPEN表中，并计算估价函数f(S0)。

(2) 如果OPEN是个空表，则没有解，失败退出；否则继续。

(3) 把OPEN表中的第一个结点（股价函数最小的结点n），移入CLOSED表。

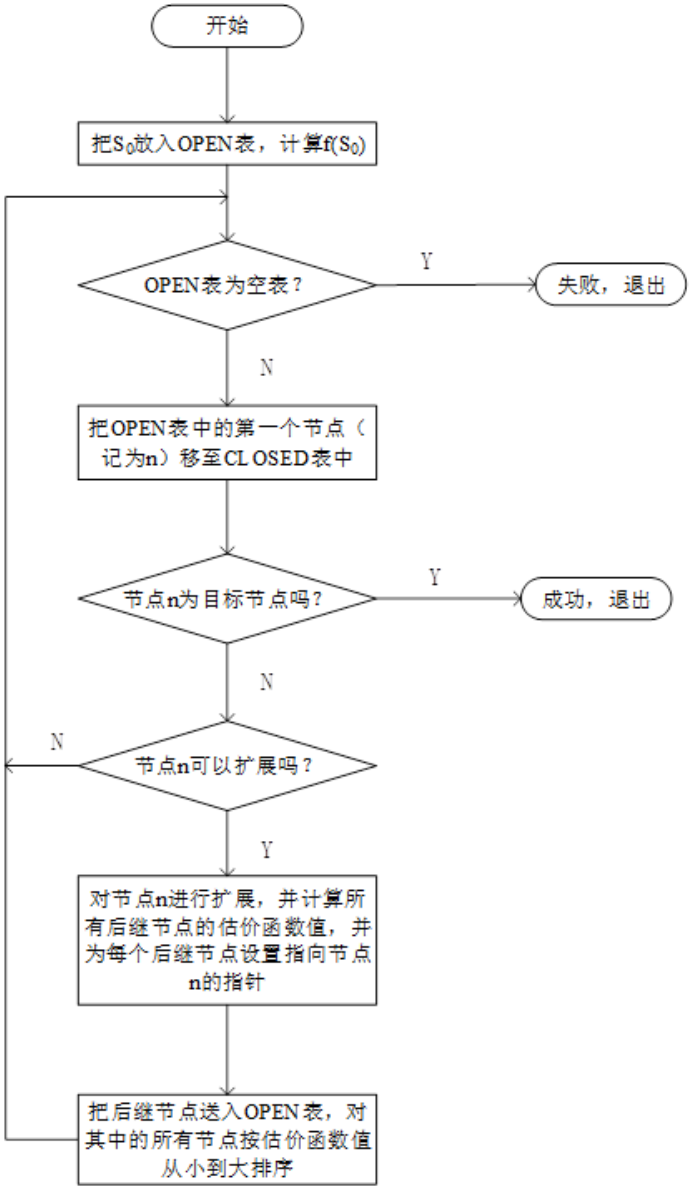
(4) 如果n是目标结点，问题得解，退出。否则继续。

(5) 判断结点n是否可扩展。若否则转向第(2)步，若是则转向(6)。

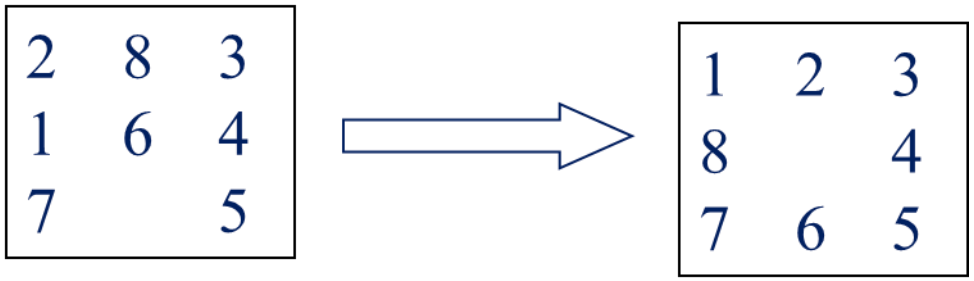
(6) 对结点n进行扩展，并对其所有后继结点计算估价函数f(n)的值，并为每个后继结点设置指向n结点的指针。把这些后继结点都送入OPEN表，然后对OPEN表中的全部结点按照估价函数值从小到大的顺序排序。

(7) 转向第(2)步。

搜索技术-启发式搜索-全局最佳优先搜索流程图



搜索技术-启发式搜索-全局最佳优先搜索实例

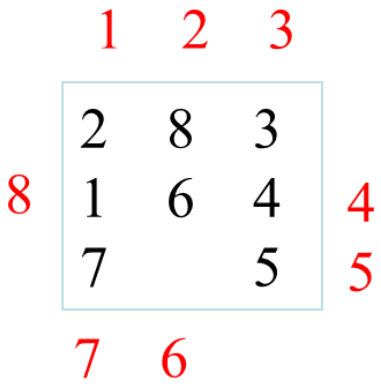


- 定义评价函数：

$f(n) = g(n) + h(n) = d(n) + h(n)$

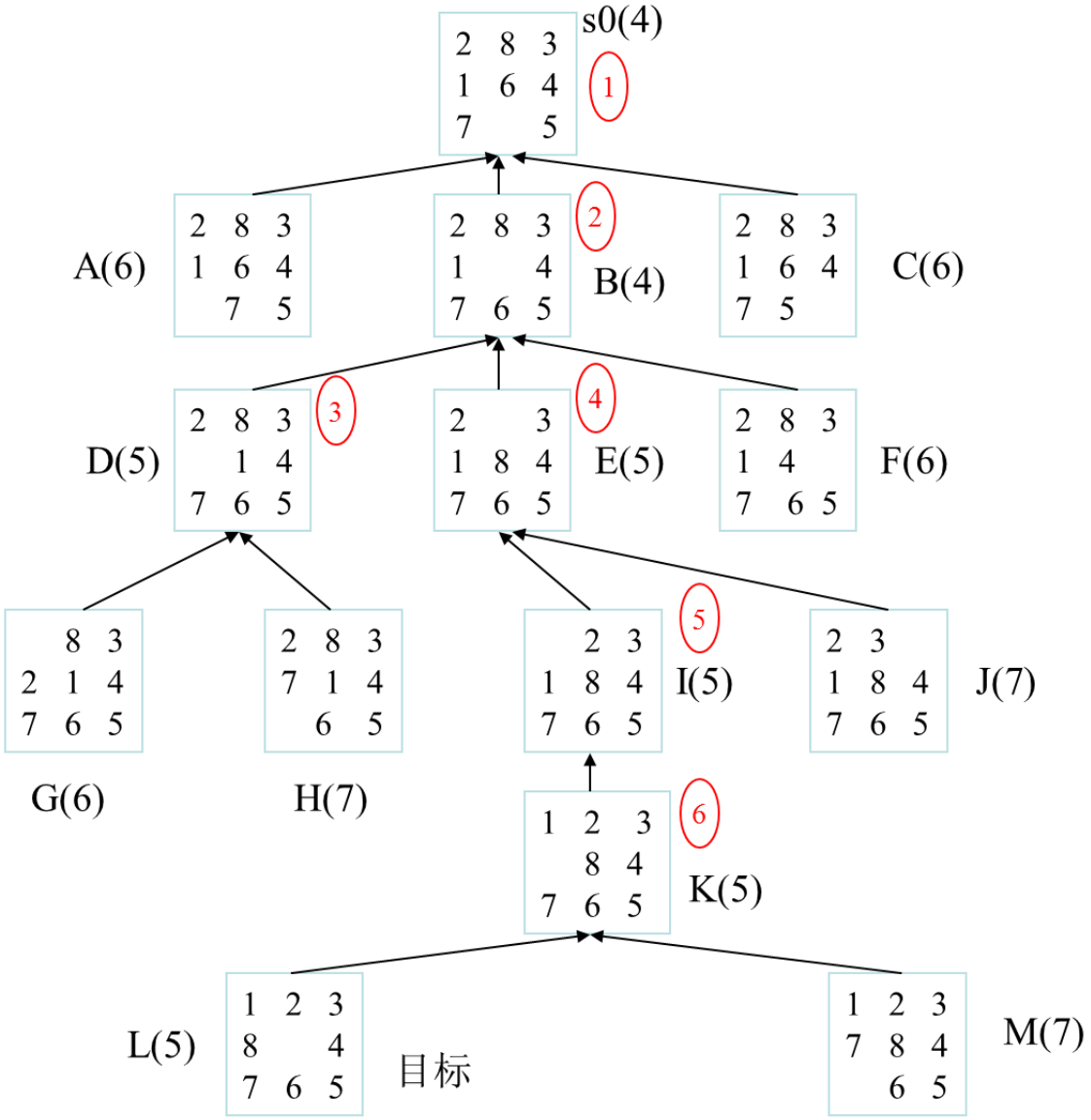
d(n)：代表结点的深度，表示从初始结点到当前结点的消耗值

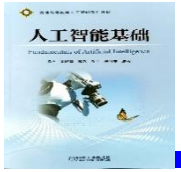
h(n)：为当前结点“不在位”的牌数



h(n)=4

搜索技术-启发式搜索-全局最佳优先搜索实例





搜索技术-启发式搜索- A*算法

- A*算法是一种有序搜索算法，其特点在于对估价函数的定义
  - ✓ 令k(ni，nj)表示任意两个结点ni和nj之间最小代价路径的实际代价
  - ✓ 从结点n到某个具体的目标结点ti，某一条最小代价路径的代价可由k(n,ti)给出
  - ✓ 令h*(n)表示整个目标结点集合｛ti｝上所有k(n,ti)中最小的一个
  - ✓ h*(n)就是从n到目标结点最小代价路径的代价
  - ✓ 从n到目标结点能够获得h*(n)的任一路径就是一条从n到某个目标结点的最佳路径
  - ✓ 对于任何不能到达目标结点的结点n，函数h*没有定义

搜索技术-启发式搜索- A*算法的估价函数

- 估价函数f(n) = g(n) + h(n)是对下列函数的估计或近似：

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

✓ f*(n)：从初始结点到结点n的一条最佳路径的实际代价加上结点n到目标结点的最佳路径的代价之和。

✓ g*(n)：从初始结点到结点n之间最小路径的实际代价

✓ h*(n)：从结点n到目标结点的最小代价路径上代价

✓ 恒有： $g^*(n) \leqslant g(n)$

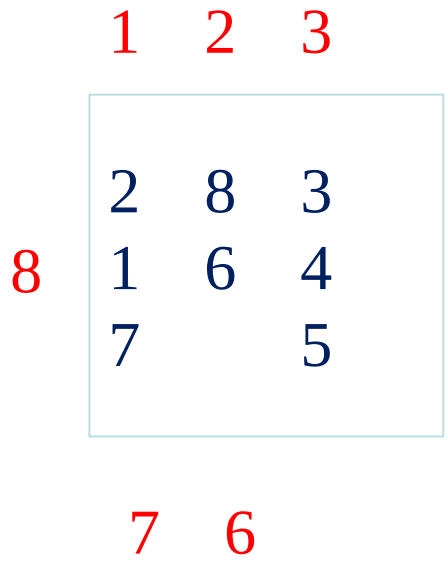
- A*算法中，要求启发函数h(n)是h*(n)的下界

$$h(n) \leqslant h^*(n)$$

- 极端情况下，若h(n)=0，一定能找到最佳解路径

搜索技术-启发式搜索- A*条件举例

- 8数码问题
  - ✓ h(n) = “不在位”的牌数
  - ✓ h*(n) = “不在位”牌的距离和



将牌1：1
将牌2：1
将牌6：1
将牌8：2