第 4 章　搜索策略

教材：

王万良《人工智能应用教程》（2版）

清华大学出版社，2025

第4章 搜索策略

- 在求解一个问题时，涉及到两个方面：一是该问题的表示，二是选择一种相对合适的求解方法。由于绝大多数需要人工智能方法求解的问题缺乏直接求解的方法，因此，搜索不失为一种求解问题的一般方法。搜索求解的应用非常广泛，例如，在下棋等游戏软件中。
- 下面首先讨论搜索的基本概念，然后着重介绍状态空间知识表示和搜索策略，主要有回溯策略、宽度优先搜索、深度优先搜索等盲目的图搜索策略，以及A及A*搜索算法等启发式图搜索策略。

第4章 搜索策略

- 4.1 搜索的概念
- 4.2 如何用状态空间表示搜索对象
- 4.3 回溯策略
- 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

第4章 搜索策略

- ✓ 4.1 搜索的概念
- 4.2 如何用状态空间表示搜索对象
- 4.3 回溯策略
- 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

4.1 搜索的概念

(1) **数据驱动**：从初始状态出发的正向搜索。

正向搜索——从问题给出的条件出发。

(2) **目的驱动**：从目的状态出发的逆向搜索。

逆向搜索：从想达到的目的入手，看哪些操作算子能产生该目的以及应用这些操作算子产生目的时需要哪些条件。

(3) 双向搜索

双向搜索：从开始状态出发作正向搜索，同时又从目的状态出发作逆向搜索，直到两条路径在中间的某处汇合为止。

4.1 搜索的概念

（1）**盲目搜索**：在不具有对特定问题的任何有关信息的条件下，按固定的步骤（依次或随机调用操作算子）进行的搜索。

（2）**启发式搜索**：考虑特定问题领域可应用的知识，动态地确定调用操作算子的步骤，优先选择较适合的操作算子，尽量减少不必要的搜索，以求尽快地到达结束状态。

第4章 搜索策略

- 4.1 搜索的概念
- ✓ 4.2 如何用状态空间表示搜索对象
- 4.3 回溯策略
- 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

4.2 如何用状态空间表示搜索对象

- **4.2.1 状态空间知识表示方法**
- **4.2.2 状态空间的图描述**

4.2.1 状态空间知识表示方法

- **状态**：表示系统状态、事实等叙述型知识的一组变量或数组：

$$Q = [q_1, q_2, \cdots, q_n]^{\mathrm{T}}$$

⑩ **操作**：表示引起状态变化的过程型知识的一组关系或函数：

$$F = \{f_1, f_2, \cdots, f_m\}$$

4.2.1 状态空间知识表示方法

- **状态空间**：利用状态变量和操作符号，表示系统或问题的有关知识的符号体系，状态空间是一个四元组：

$$(S, O, S_0, G)$$

S：状态集合。

O：操作算子的集合。

S_0：包含问题的初始状态，是 S 的非空子集。

G：若干具体状态或满足某些性质的路径信息描述。

4.2.1 状态空间知识表示方法

- **求解路径**：从 S_0 结点到 G 结点的路径。

⑩ **状态空间的一个解**：一个有限的操作算子序列。

$$S_0 \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \cdots \xrightarrow{O_k} G$$

$O_1, \cdots, O_k$：状态空间的一个解。

4.2.1 状态空间知识表示方法

- **例4.1　八数码问题的状态空间。**

2	3	1
5		8
4	6	7

初始状态

1	2	3
8		4
7	6	5

目标状态

状态集 S ：所有摆法

操作算子：

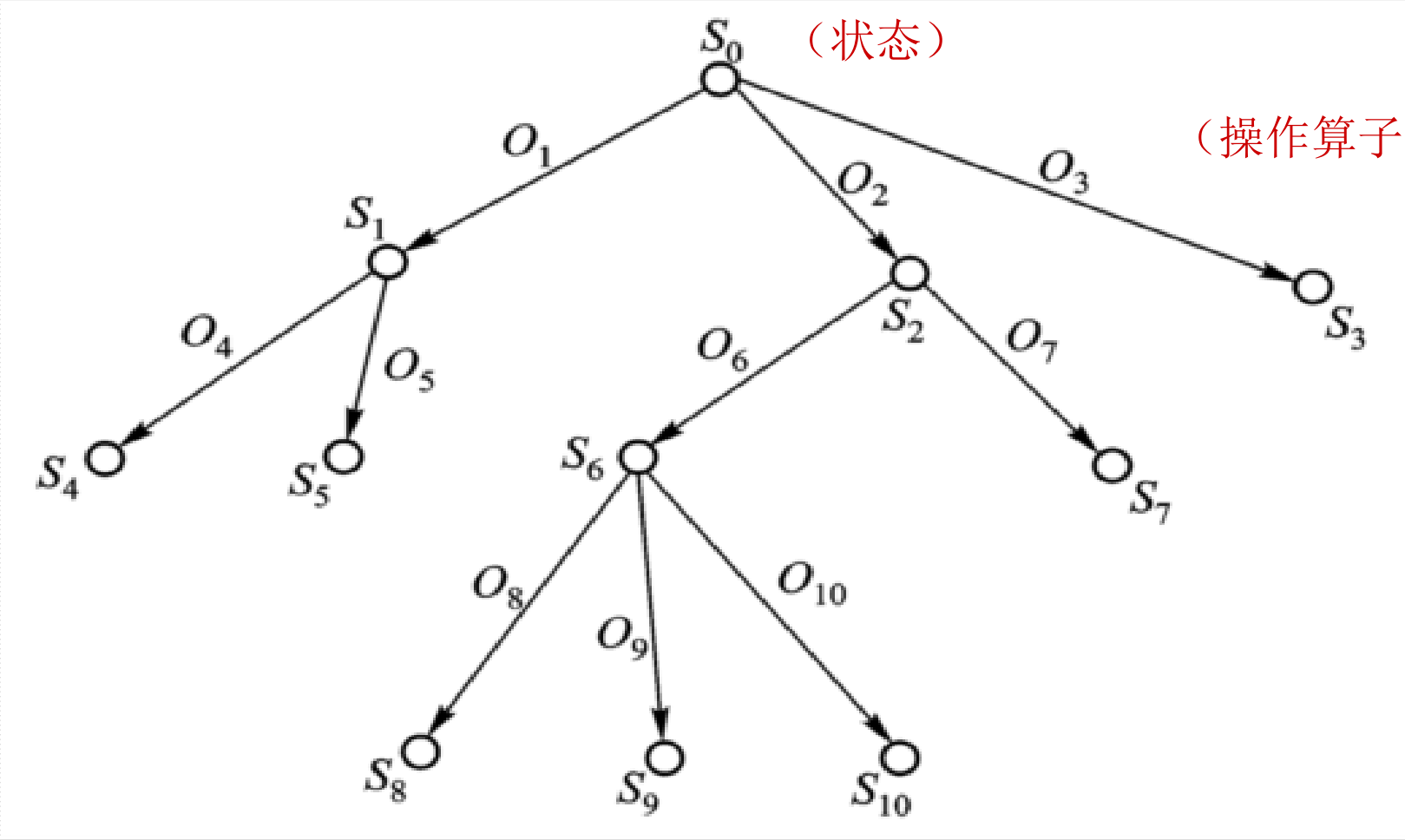
将空格向上移Up

将空格向左移Left

将空格向下移Down

将空格向右移Right

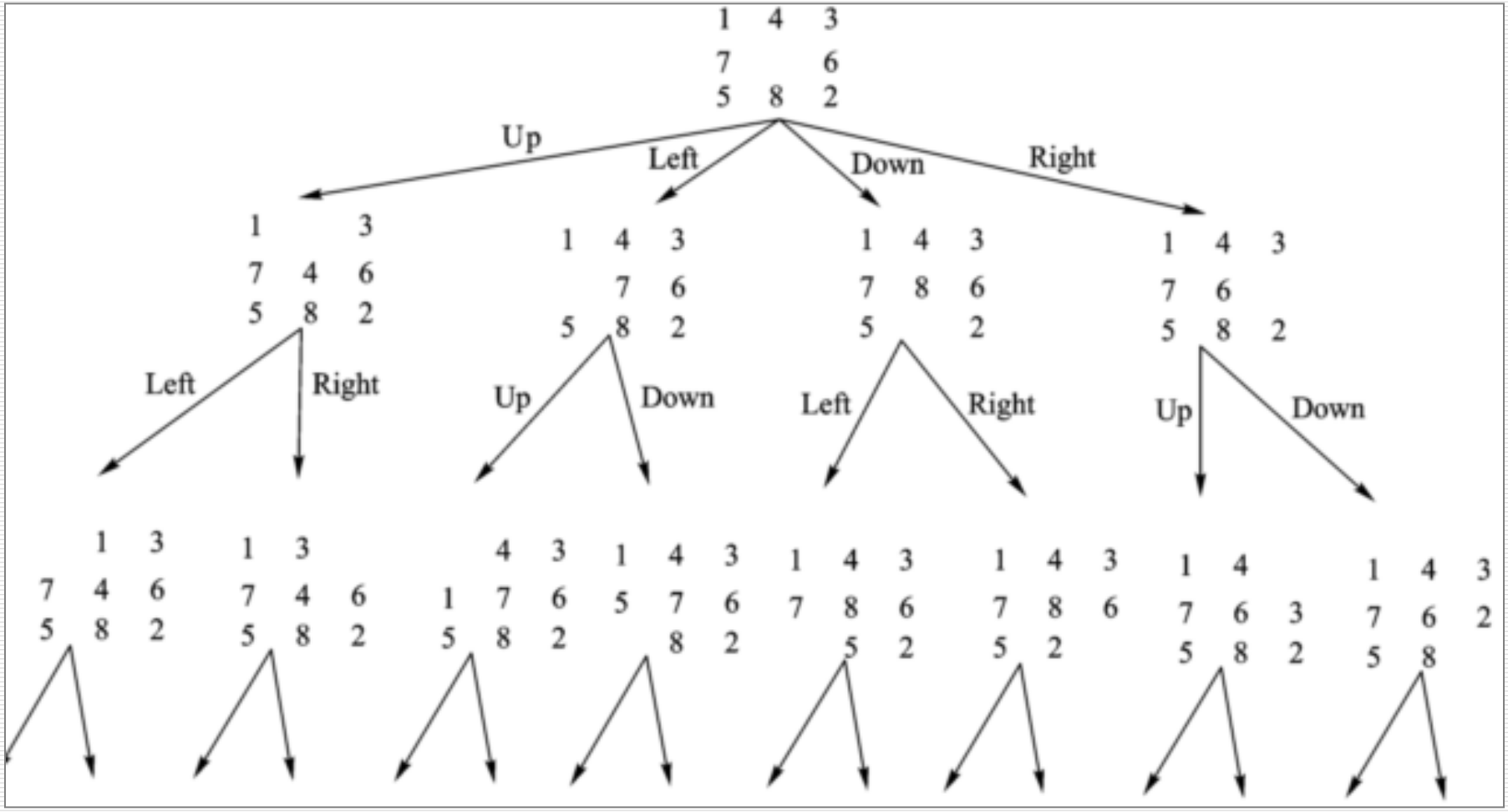
4.2.2 状态空间的图描述



状态空间的有向图描述

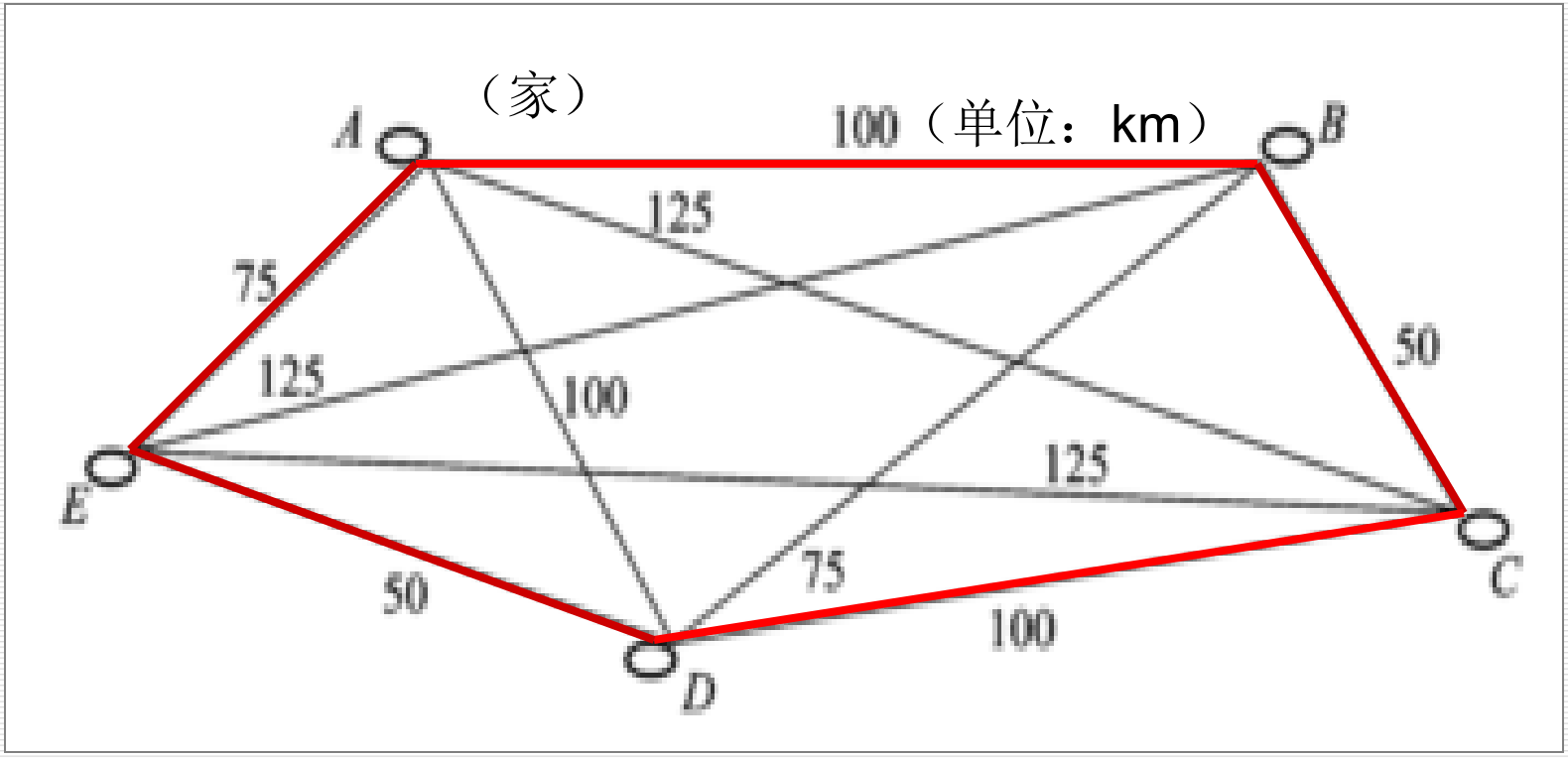
4.2.2 状态空间的图描述

例4.2 八数码状态空间图



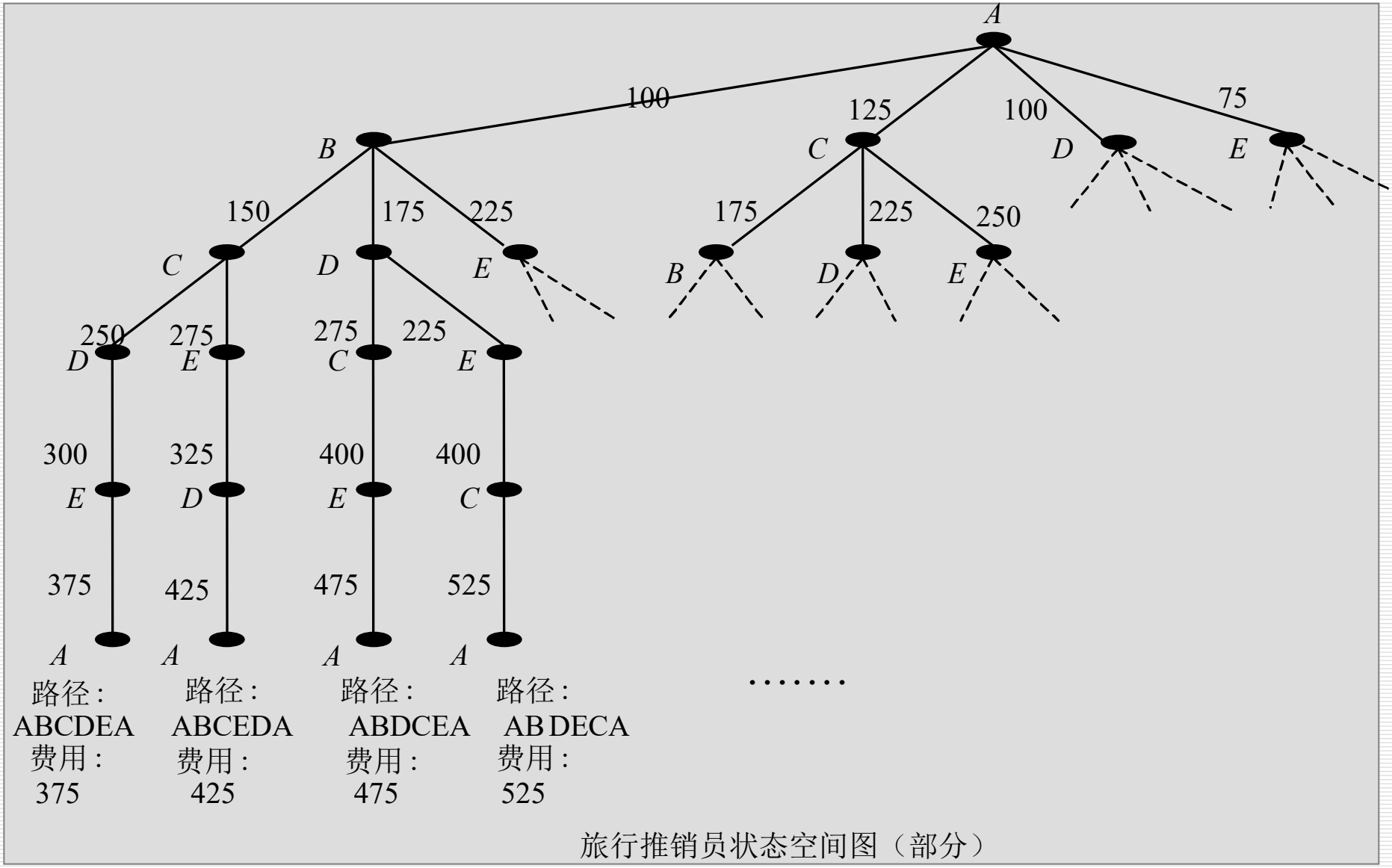
4.2.2 状态空间的图描述

例4.3　旅行商问题（traveling salesman problem, TSP）或邮递员路径问题。



可能路径：费用为375的路径（*A*，*B*，*C*，*D*，*E*，*A*）

4.2.2 状态空间的图描述



旅行推销员状态空间图（部分）

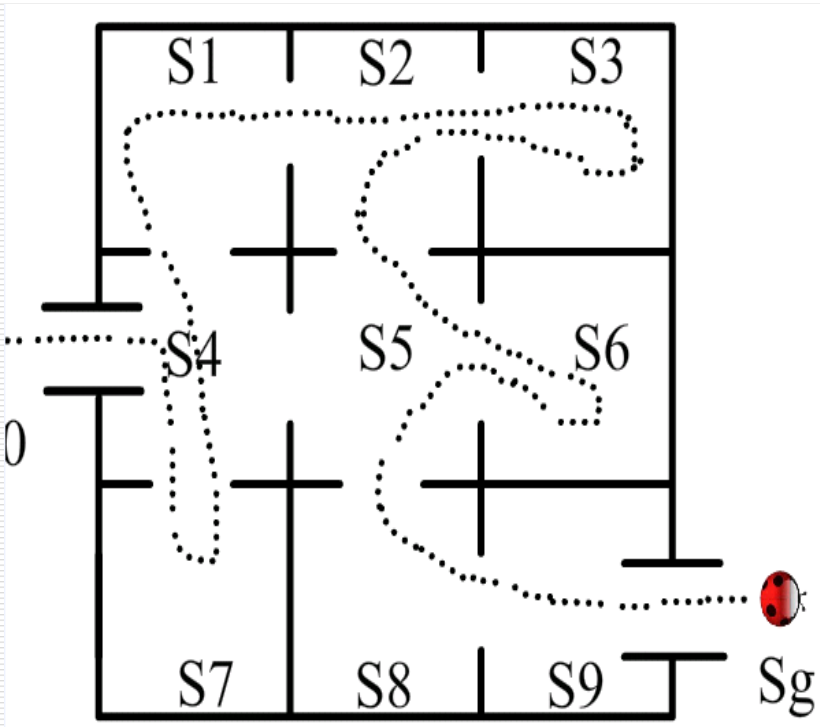
第4章 搜索求解策略

- 4.1 搜索的概念
- 4.2 如何用状态空间表示搜索对象
- ✓ 4.3 回溯策略
- 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

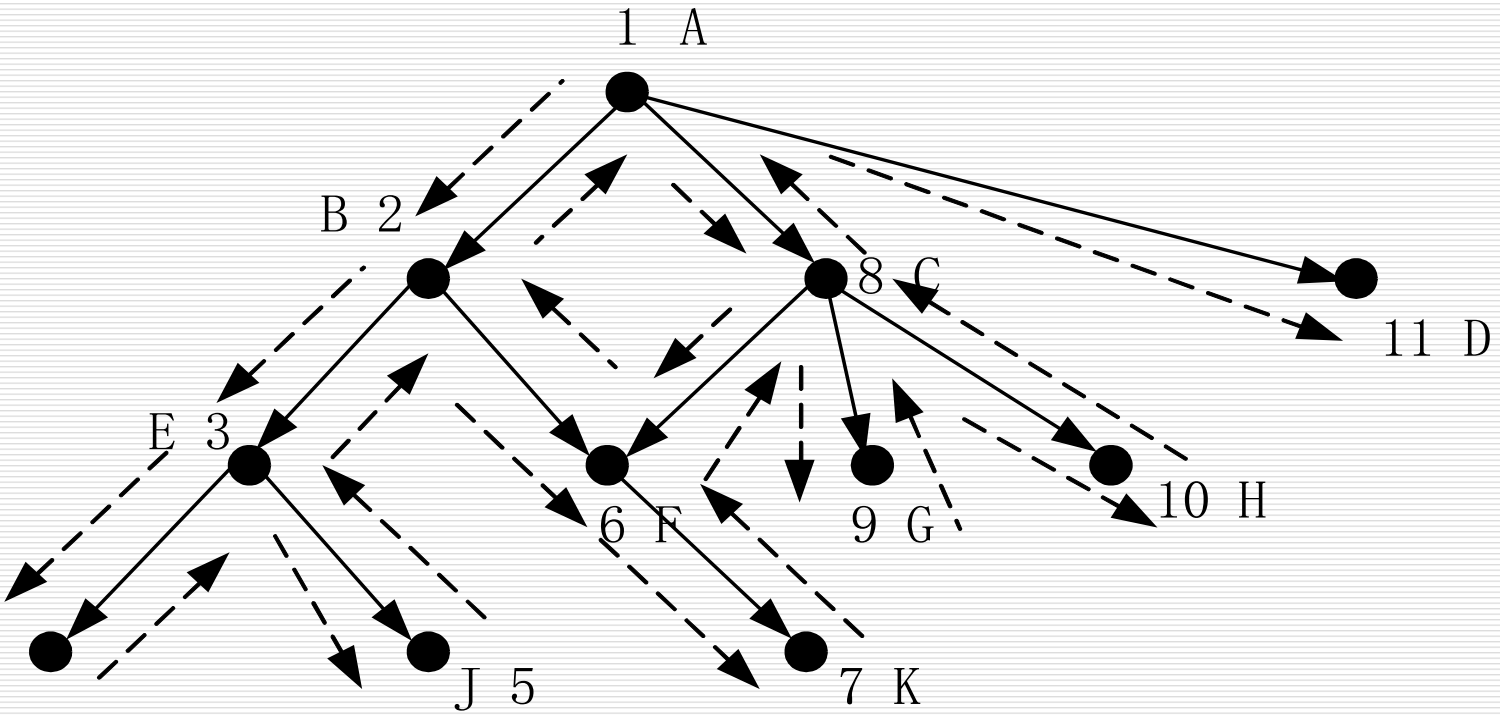
4.3 回溯策略

带回溯策略的搜索：

- 从初始状态出发，不停地、试探性地寻找路径，直到它到达目的或“不可解结点”，即“死胡同”为止。
- 若它遇到不可解结点就回溯到路径中最近的父结点上，查看该结点是否还有其他的子结点未被扩展。若有，则沿这些子结点继续搜索；如果找到目标，就成功退出搜索，返回解题路径。



4.3 回溯策略



回溯搜索示意图

第4章 搜索求解策略

- 4.1 搜索的概念
- 4.2 如何用状态空间表示搜索对象
- 4.3 回溯策略
- ✓ 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

4.4.1 宽度优先搜索策略

- **宽度优先搜索(breadth-first search，广度优先搜索): 以接近起始节点的程度（深度）为依据，进行逐层扩展的节点搜索方法。**

⑩ 特点:

1) 每次选择深度最浅的节点首先扩展，搜索是逐层进行的;

2) 一种高价搜索，但若有解存在，则必能找到它。

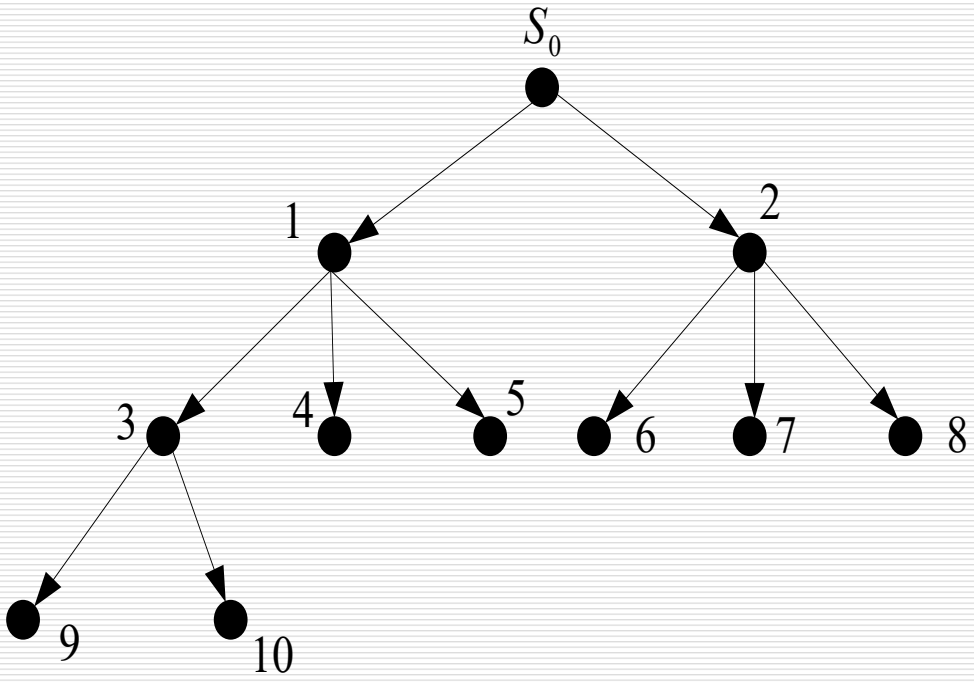
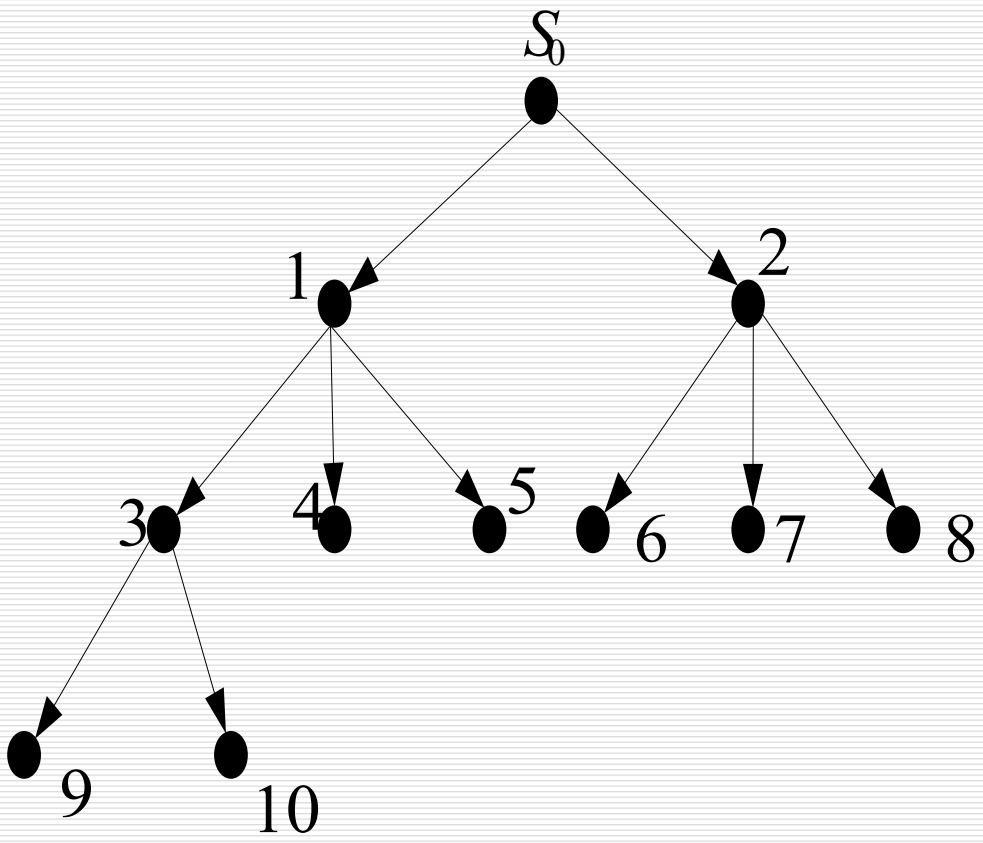


图5.6 宽度优先搜索法中状态的搜索次序

4.4.1 宽度优先搜索策略

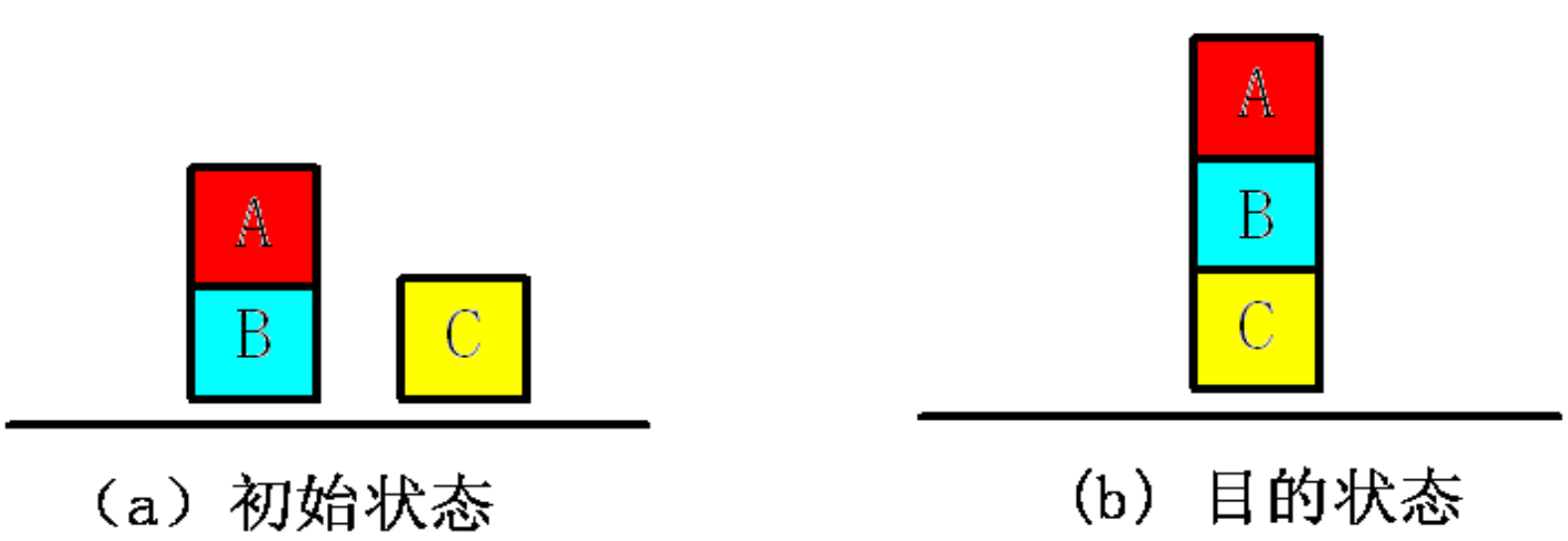


宽度优先搜索法中状态的搜索次序

- open表（NPS表）：已经生成出来但其子状态未被搜索的状态。
- closed表（ PS表和NSS表的合并）：记录了已被生成扩展过的状态。

4.4.1 宽度优先搜索策略

- 例4.4　通过搬动积木块，希望从初始状态达到一个目的状态，即三块积木堆叠在一起。



（a）初始状态　　（b）目的状态

图5.7 积木问题

4.4.1 宽度优先搜索策略

- **操作算子为MOVE（X，Y）：把积木X搬到Y（积木或桌面）上面。**

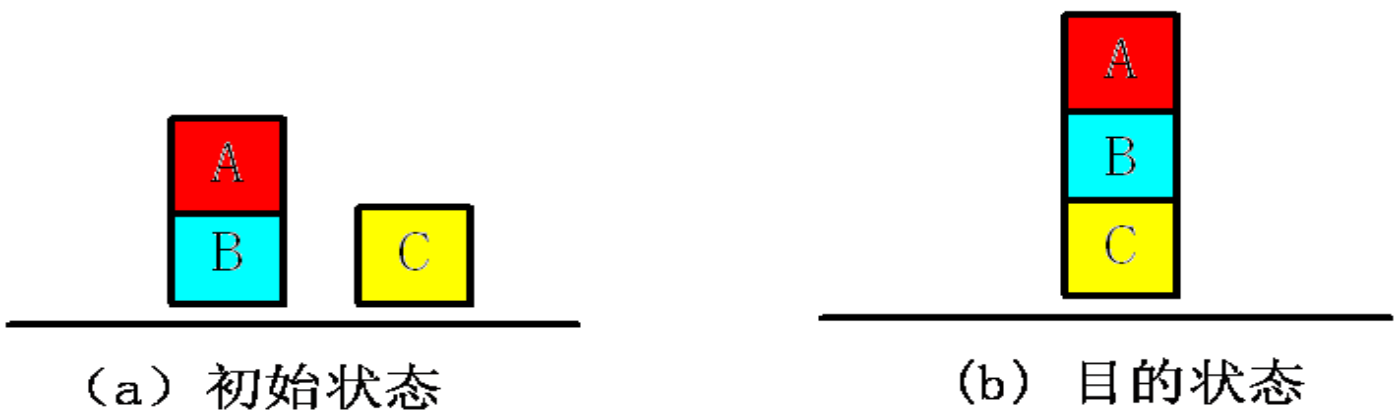
MOVE（A，Table）：“搬动积木A到桌面上”。

- 操作算子可运用的先决条件：

1）被搬动积木X的顶部必须为空；

2）如果 Y 是积木，则积木 Y 的顶部也必须为空；

3）同一状态下，运用操作算子的次数不得多于一次。



（a）初始状态　　（b）目的状态

图5.7 积木问题

4.4.1 宽度优先搜索策略

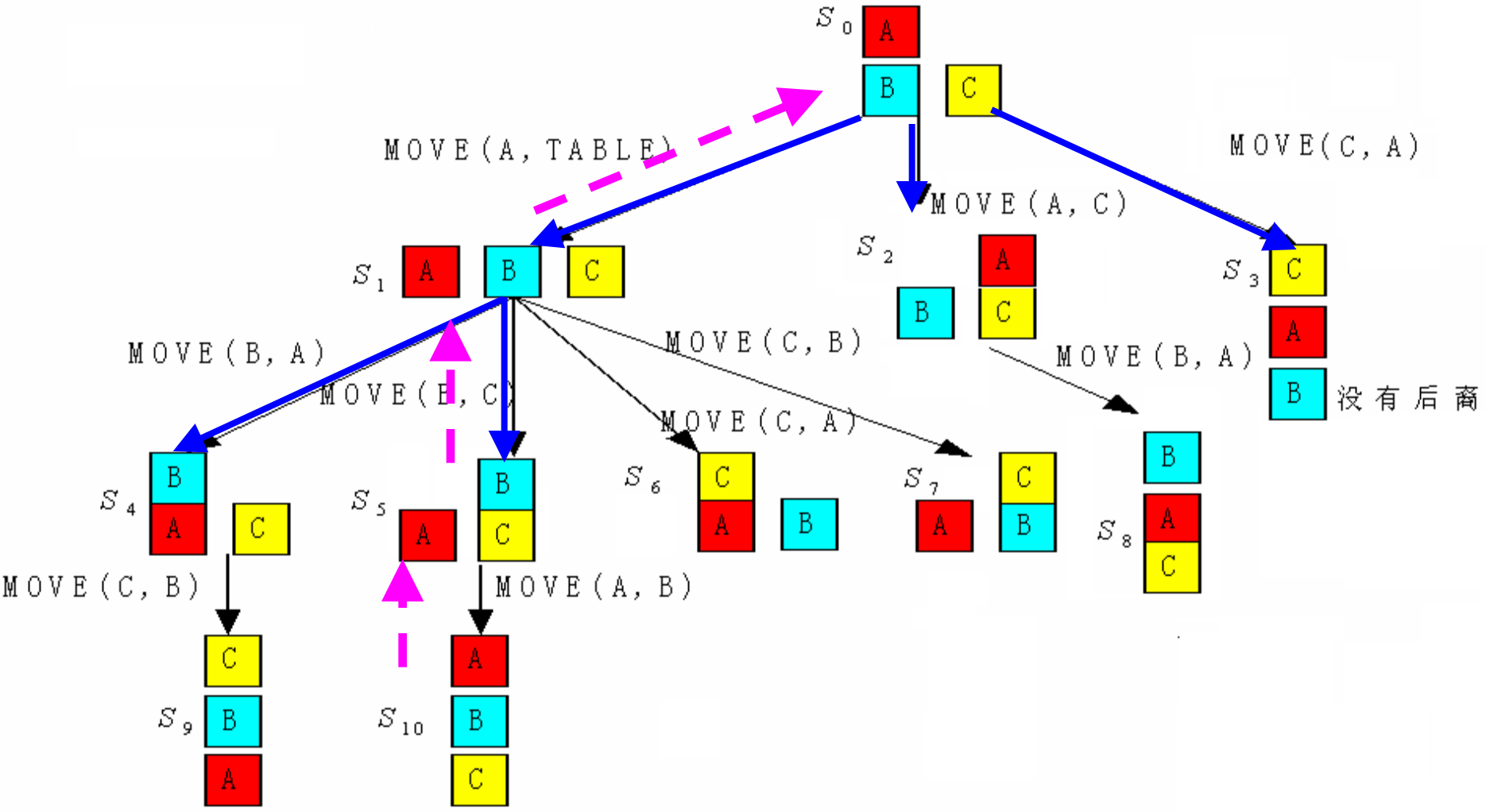
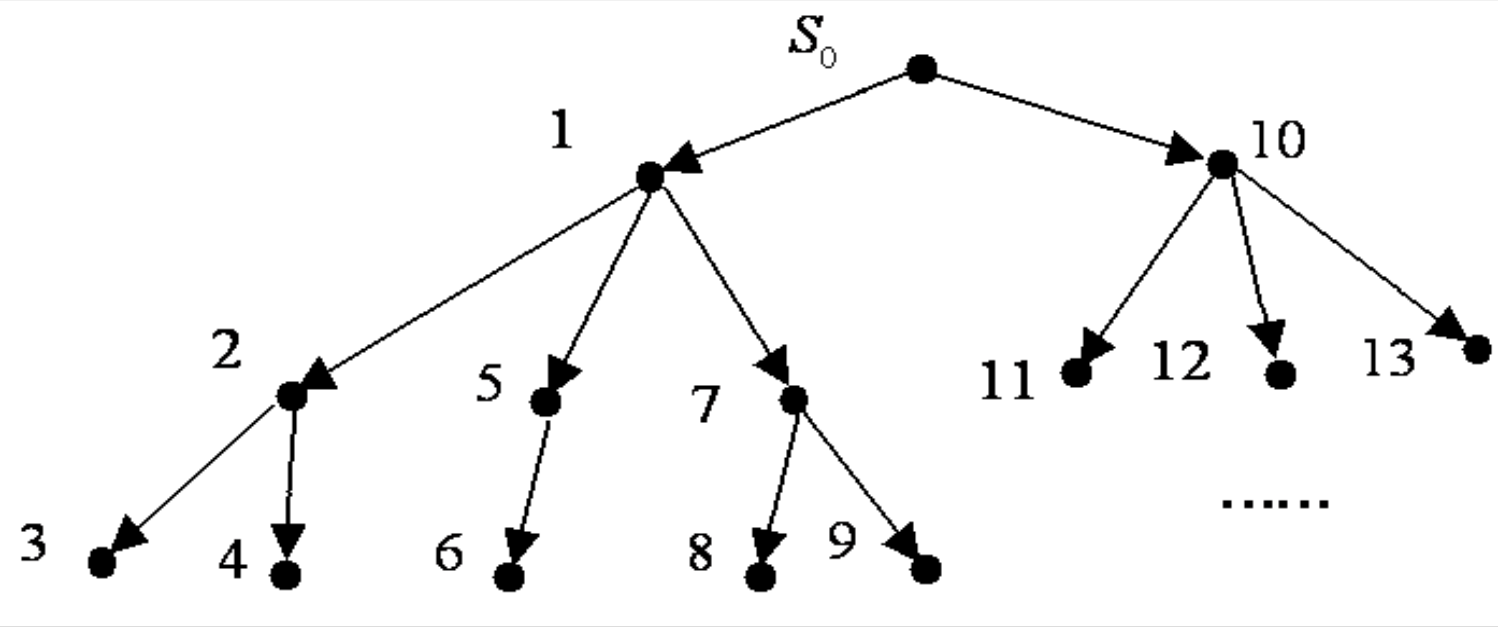


图 5.8 积木问题的宽度优先搜索树

4.4.2 深度优先搜索策略

- **深度优先搜索（Depth-first Search):** 首先扩展最新产生的节点, 深度相等的节点按生成次序的盲目搜索。



- 特点：扩展最深的节点的结果使得搜索沿着状态空间某条单一的路径从起始节点向下进行下去；仅当搜索到达一个没有后裔的状态时，才考虑另一条替代的路径。

4.4.2 深度优先搜索策略

- 在深度优先搜索中，当搜索到某一个状态时，它所有的子状态以及子状态的后裔状态都必须先于该状态的兄弟状态被搜索。
- 为了保证找到解，应选择合适的深度限制值，或采取不断加大深度限制值的办法，反复搜索，直到找到解。
- 深度优先搜索并不能保证第一次搜索到的某个状态时的路径是到这个状态的最短路径。
- 对任何状态而言，以后的搜索有可能找到另一条通向它的路径。如果路径的长度对解题很关键的话，当算法多次搜索到同一个状态时，它应该保留最短路径。

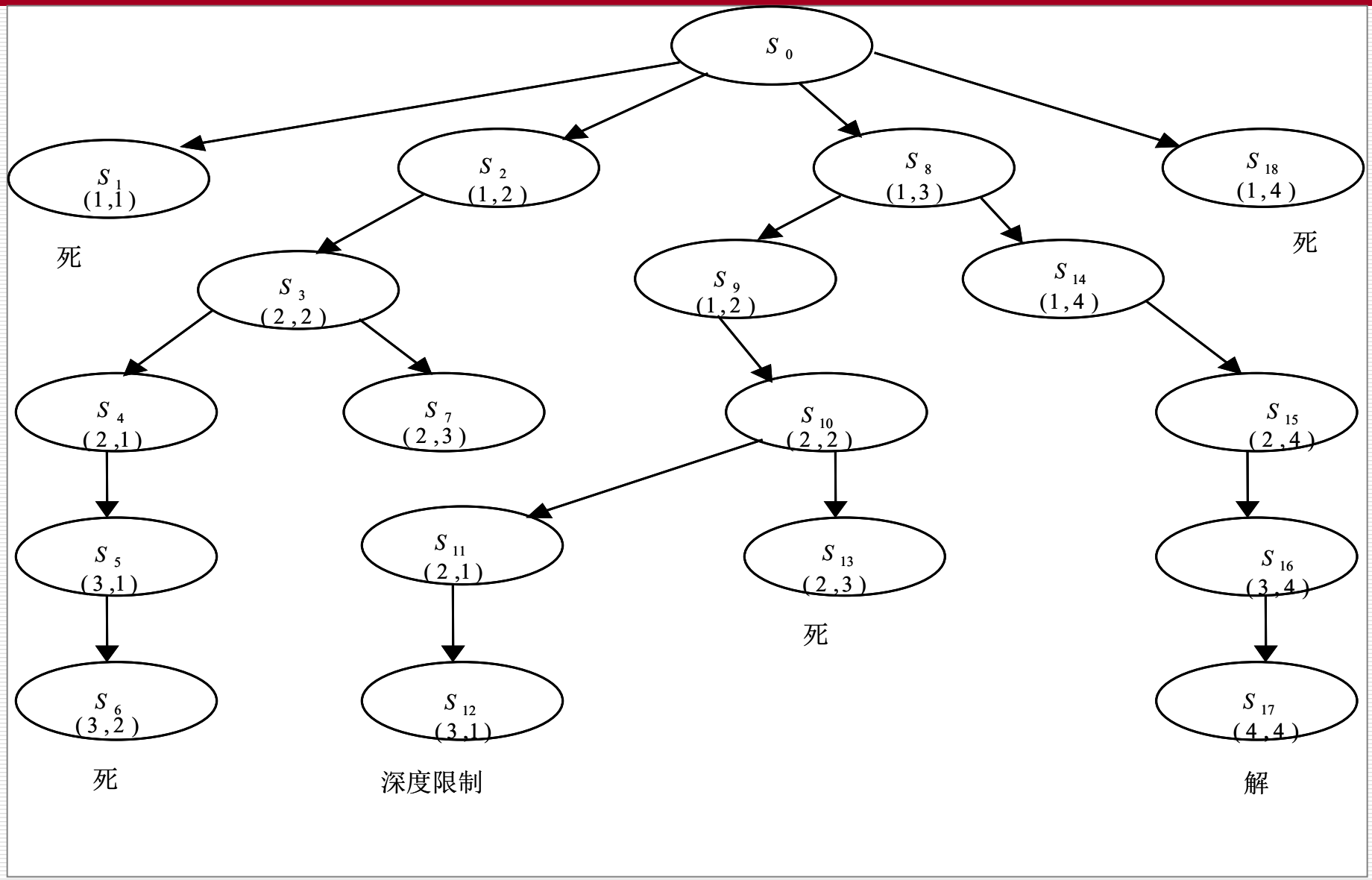
4.4.2 深度优先搜索策略

- **例4.5　卒子穿阵问题**，要求一卒子从顶部通过下图所示的阵列到达底部。卒子行进中不可进入到代表敌兵驻守的区域（标注1），并不准后退。假定深度限制值为5。

行	1	2	3	4	列
1	1	0	0	0	
2	0	0	1	0	
3	0	1	0	0	
4	1	0	0	0	

阵列图

4.4.2 深度优先搜索策略



卒子穿阵的深度优先搜索树

第4章 搜索求解策略

- 4.1 搜索的概念
- 4.2 如何用状态空间表示搜索对象
- 4.3 回溯策略
- 4.4 盲目的图搜索策略
- 4.5 启发式图搜索策略

4.5 启发式图搜索策略

- **4.5.1 启发式策略**
- **4.5.2 启发信息和估价函数**
- **4.5.3 *A*搜索算法**
- **4.5.4 *A**搜索算法**
- **4.5.5 蒙特卡洛树搜索算法**

4.5.1 启发式策略

- 什么是启发式信息？
  - 用来简化搜索过程有关具体问题领域的特性的信息叫做启发信息。
- 启发式图搜索策略（利用启发信息的搜索方法）的特点：重排OPEN表，选择最有希望的节点加以扩展。
- 种类：*A*、*A**算法等。
- 运用启发式策略的两种基本情况：

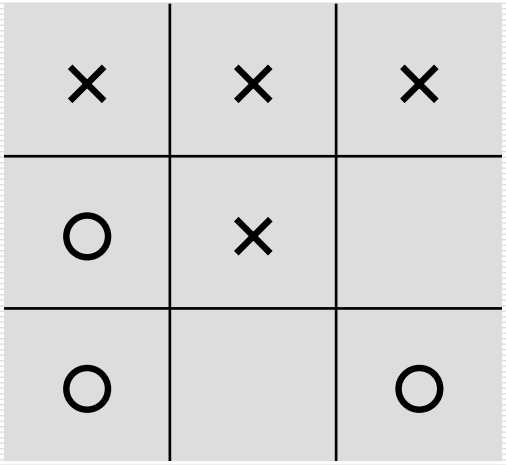
（1）一个问题由于存在问题陈述和数据获取的模糊性，可能会使它没有一个确定的解。

（2）虽然一个问题可能有确定解，但是其状态空间特别大，搜索中生成扩展的状态数会随着搜索的深度呈指数级增长。

4.5.1 启发式策略

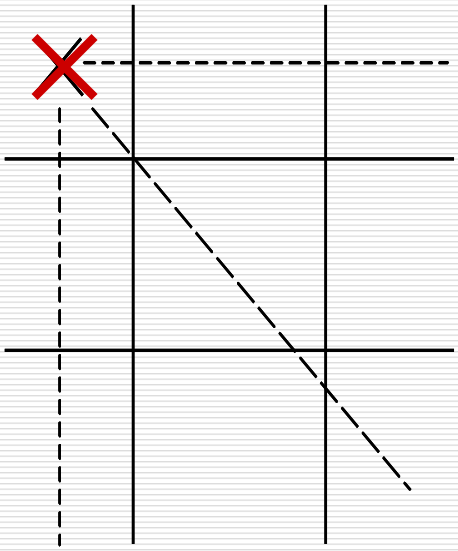
例4.6 一字棋。

在九宫棋盘上，从空棋盘开始，双方轮流在棋盘上摆各自的棋子 × 或 ○（每次一枚），谁先取得三子一线（一行、一列或一条对角线）的结果就取胜。

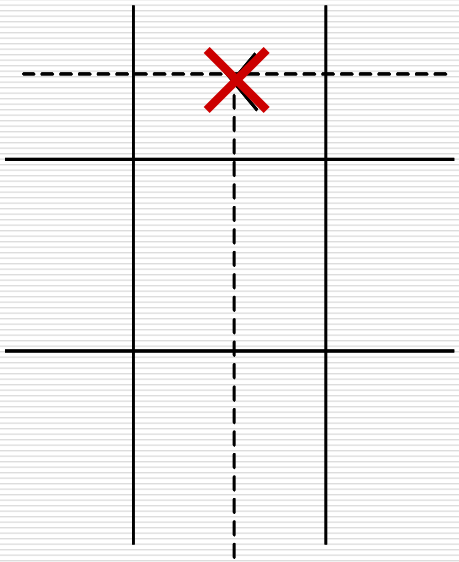


- × 和 ○ 能够在棋盘中摆成的各种不同的棋局就是问题空间中的不同状态。
- 9个位置上摆放{空， × ， ○}有 3^9 种棋局。
- 可能的走法： $9\times8\times7\times\cdots\times1$ 。

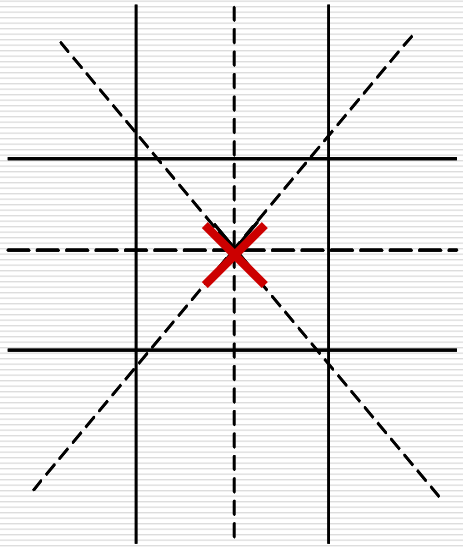
4.5.1 启发式策略



赢的几率③



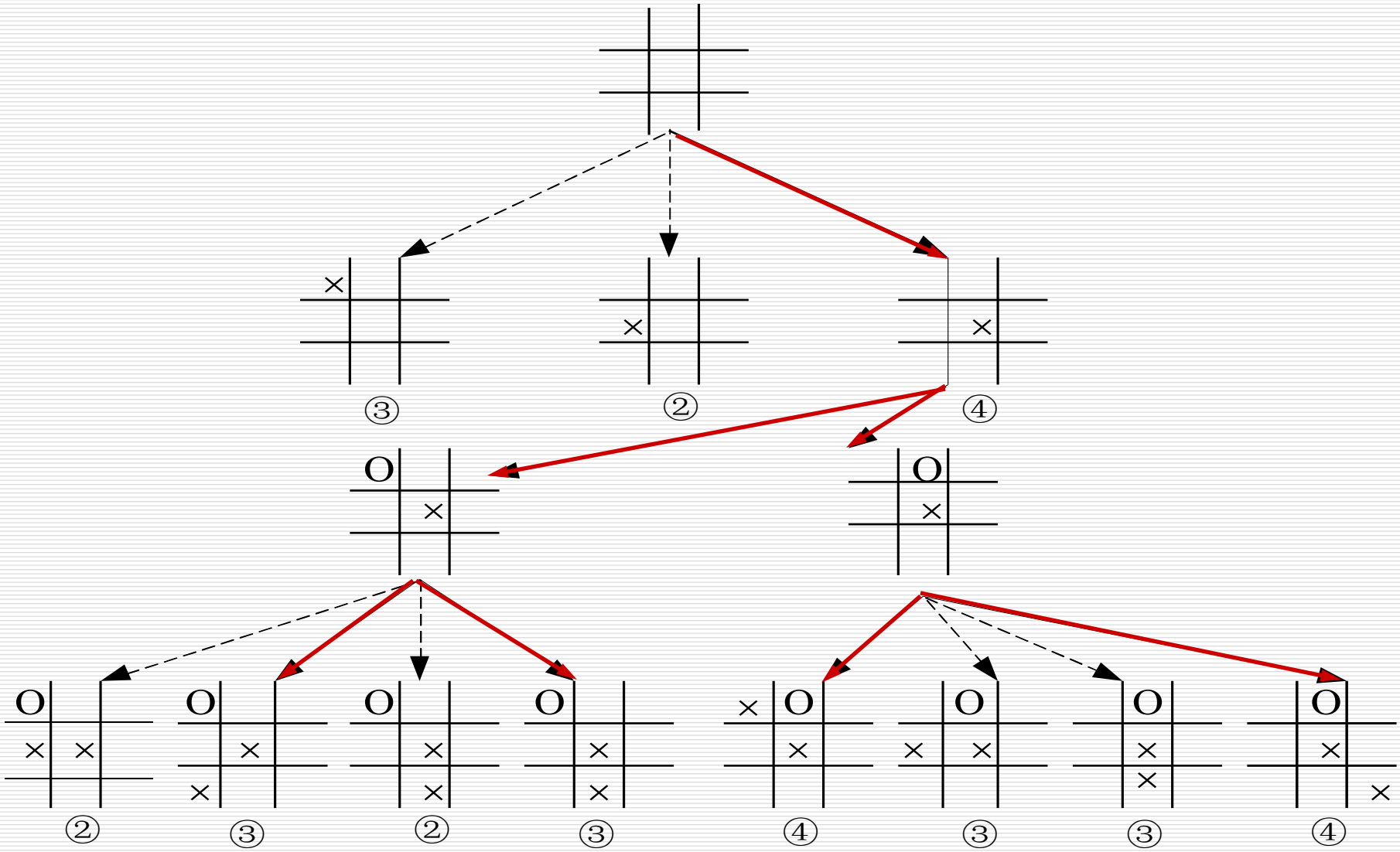
赢的几率②



赢的几率④

启发式策略的运用

4.5.1 启发式策略



启发式搜索下缩减的状态空间

4.5.2 启发信息和估价函数

- 在具体求解中，能够利用与该问题有关的信息来简化搜索过程，称此类信息为**启发信息**。
- **启发式搜索：**利用启发信息的搜索过程。
- 求解问题中能利用的大多是非完备的启发信息：

（1）求解问题系统不可能知道与实际问题有关的全部信息，因而无法知道该问题的全部状态空间。

（2）有些问题在理论上虽然存在着求解算法，但是在工程实践中，这些算法不是效率太低，就是根本无法实现。

一字棋：9！，西洋跳棋：10^{78}，国际象棋：10^{120}，围棋：10^{761}。

假设每步可以搜索一个棋局，用极限并行速度（10^{-104}年/步）来处理，搜索一遍国际象棋的全部棋局也得10^{16}年即1亿亿年才可以算完！

4.5.2 启发信息和估价函数

- 估价函数（evaluation function）：估算节点“希望”程度的量度。

- 估价函数值 $f(n)$：从初始节点经过 n 节点到达目标节点的路径的最小代价估计值，其一般形式是

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

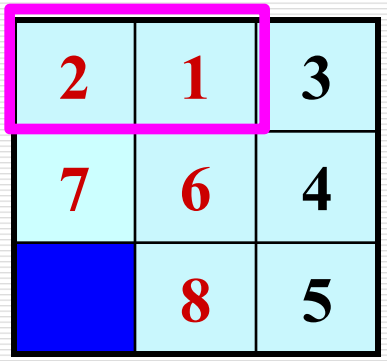
- $g(n)$：从初始节点 S_0 到节点 n 的实际代价；
- $h(n)$：从节点 n 到目标节点 S_g 的最优路径的估计代价，称为启发函数。

$h(n)$ 比重大：降低搜索工作量，但可能导致找不到最优解；

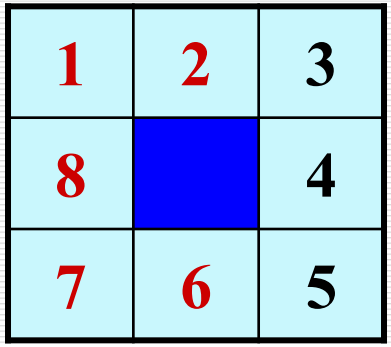
$h(n)$ 比重小：一般导致工作量加大，极限情况下变为盲目搜索，但可能可以找到最优解。

4.5.2 启发信息和估价函数

- 例4.7　八数码问题的启发函数：
- 启发函数1：取一棋局与目标棋局相比，其位置不符的数码数目，例如 $h(S_0) = 5$;
- 启发函数2：各数码移到目标位置所需移动的距离的总和,例如 $h(S_0) = 6$;
- 启发函数3：对每一对逆转数码乘以一个倍数, 例如3倍，则 $h(S_0) = 3$;
- 启发函数4：将位置不符数码数目的总和与3倍数码逆转数目相加，例如 $h(S_0) = 8$。



初始棋局



目标棋局

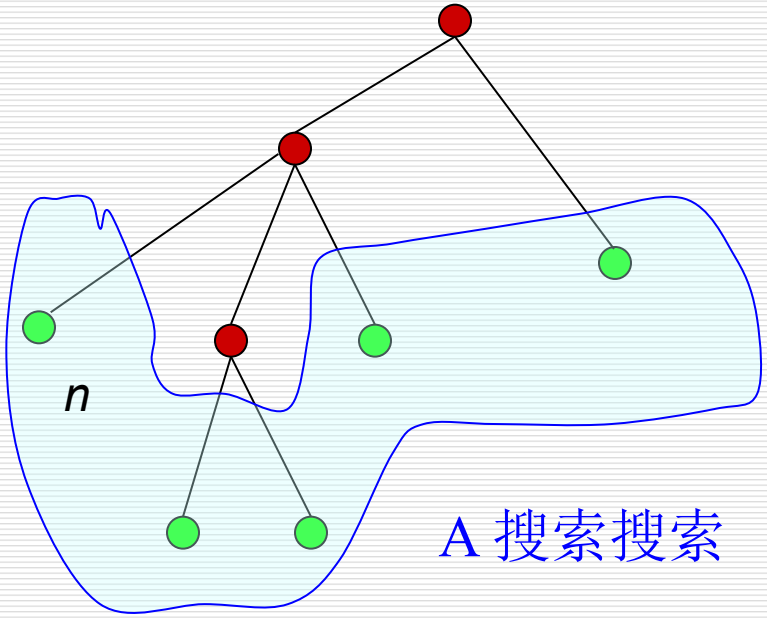
4.5.3 A搜索算法

A 搜索算法：使用了估价函数 f 的最佳优先搜索。

- 估价函数 $f(n) = g(n) + h(n)$
- 如何寻找并设计启发函数 $h(n)$，然后以 $f(n)$ 的大小来排列OPEN表中待扩展状态的次序，每次选择 $f(n)$ 值最小者进行扩展。

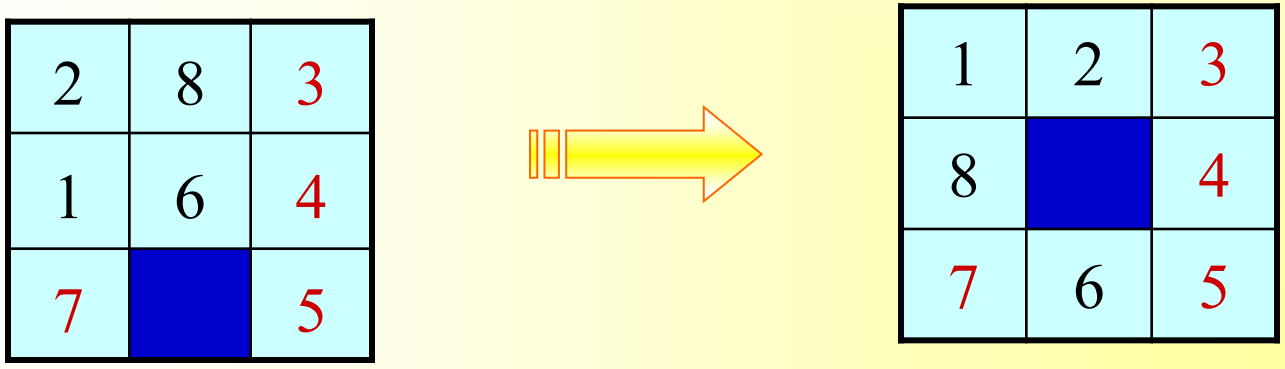
$g(n)$：状态 n 的实际代价，例如搜索的深度；

$h(n)$：对状态 n 与目标“接近程度”的某种启发式估计。



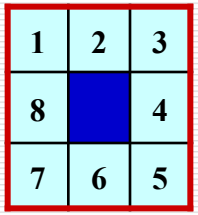
4.5.3 *A*搜索算法

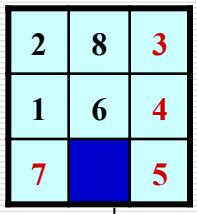
- 例4.8 利用 ***A* 搜索算法**求解八数码问题，问最少移动多少次就可达到目标状态？
- 估价函数定义为 $f(n) = g(n) + h(n)$
- $g(n)$：节点 n 的深度，如 $g(S_0)=0$。
- $h(n)$：节点 n 与目标棋局不相同的位数（包括空格），简称“不在位数”，如 $h(S_0)=5$。



初始状态 S_0　　目标状态

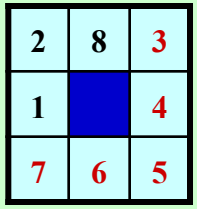
操作算子集：↑, ↓, →, ←

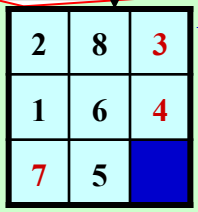




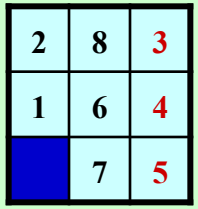
S（0+5=5）

A（1+3=4）



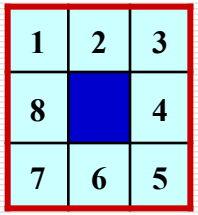


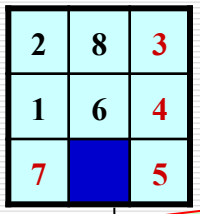
B（1+6=7）



C（1+6=7）

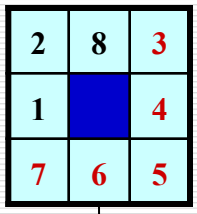
操作算子集：↑, ↓, →, ←



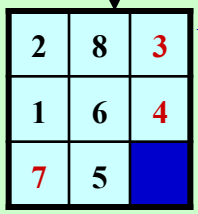


S（0+5=5）

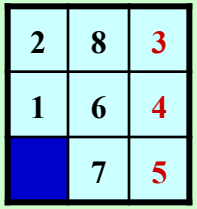
A（1+3=4）



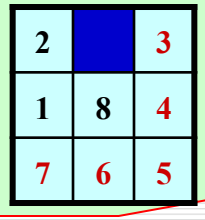
B（1+6=7）

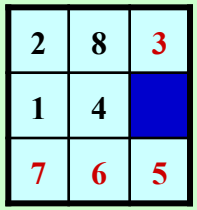


C（1+6=7）

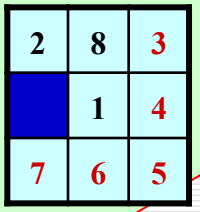


D（2+4=6）



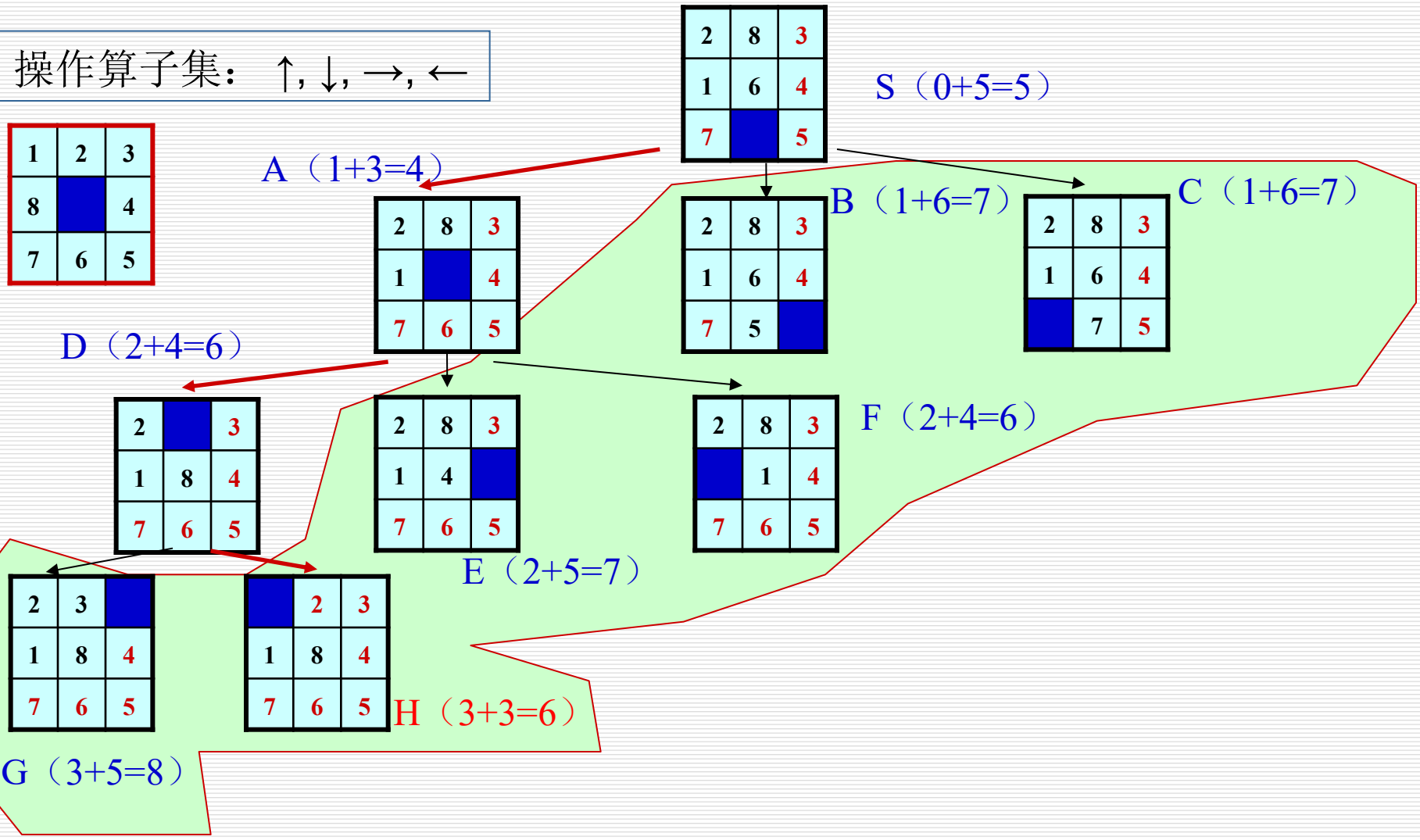


E（2+5=7）

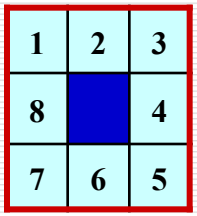


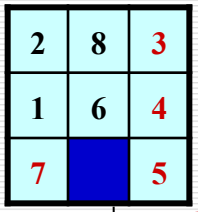
F（2+4=6）

操作算子集：↑, ↓, →, ←



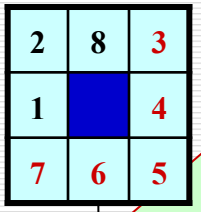
操作算子集：↑, ↓, →, ←

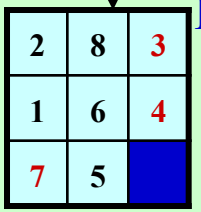




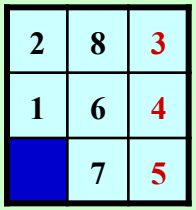
S（0+5=5）

A（1+3=4）



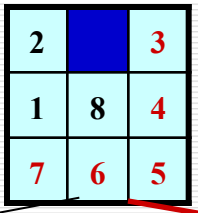


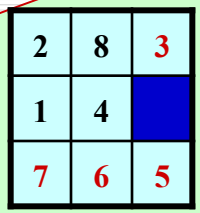
B（1+6=7）



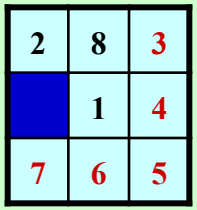
C（1+6=7）

D（2+4=6）

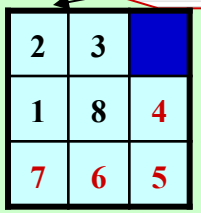


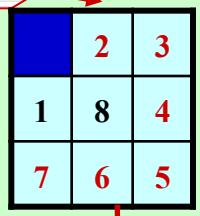


E（2+5=7）



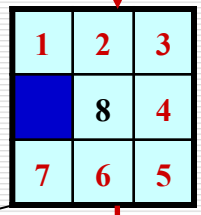
F（2+4=6）





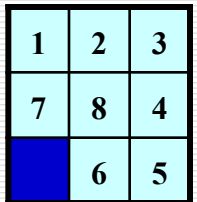
H（3+3=6）

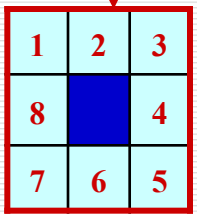
G（3+5=8）



I（4+2=6）

J（5+3=8）





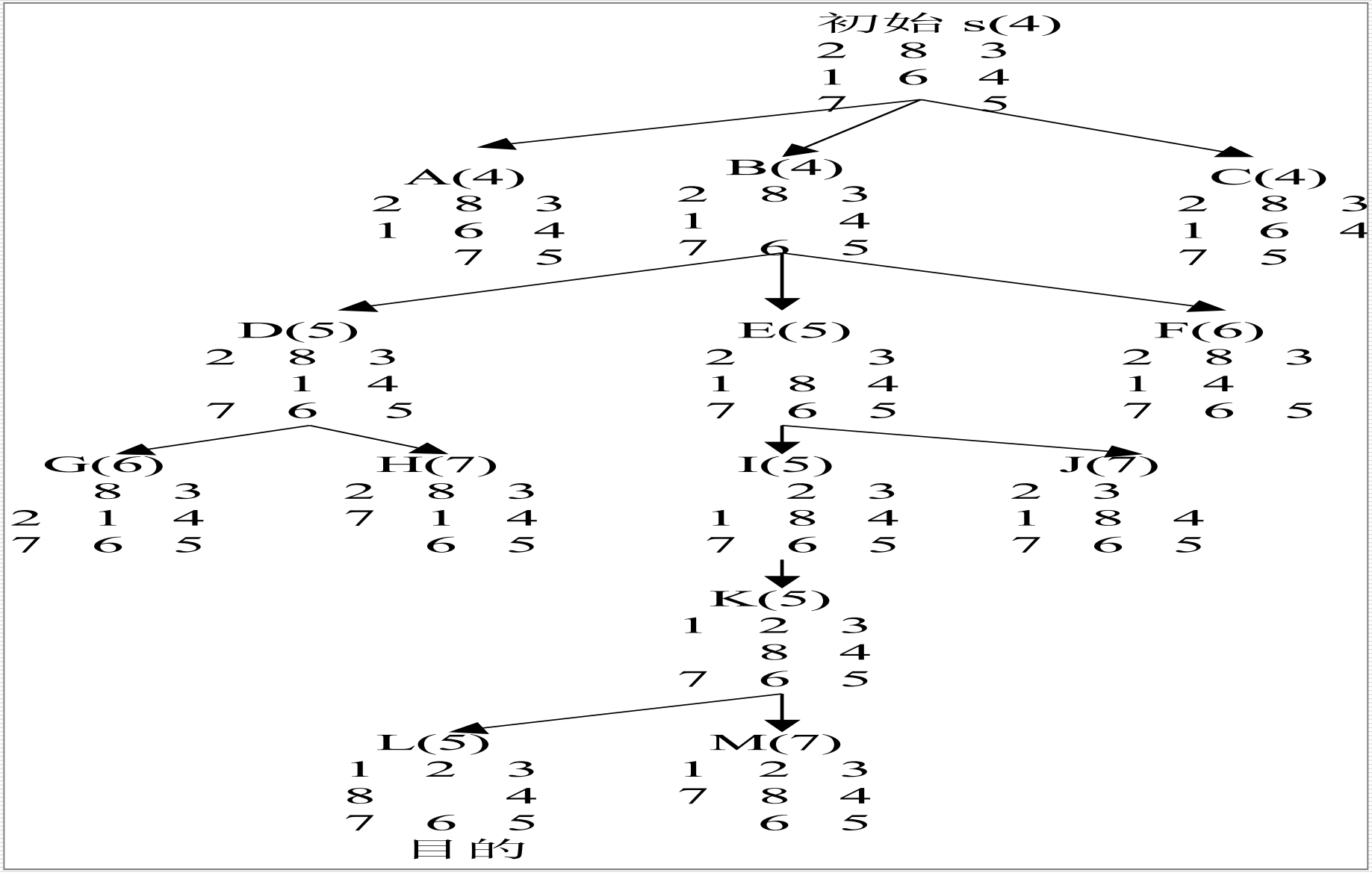
K（5+0=5）

- 问题：A搜索算法能不能保证找到最优解（路径最短的解）？

  $g(n)$：从初始节点 S 到节点 n 的实际代价；

  $h(n)$：对状态n与目标节点接近程度的某种启发式估计

4.5.3 *A*搜索算法


初始 s(4)
2 8 3
1 6 4
7 5
A(4)
2 8 3
1 6 4
7 5
B(4)
2 8 3
1 4
7 6 5
C(4)
2 8 3
1 6 4
7 5
D(5)
2 8 3
1 4
7 6 5
E(5)
2 3
1 8 4
7 6 5
F(6)
2 8 3
1 4
7 6 5
G(6)
8 3
2 1 4
7 6 5
H(7)
2 8 3
7 1 4
6 5
I(5)
2 3
1 8 4
7 6 5
J(7)
2 3
1 8 4
7 6 5
K(5)
1 2 3
8 4
7 6 5
L(5)
1 2 3
8 4
7 6 5
目的
M(7)
1 2 3
7 8 4
6 5

4.5.4 *A**搜索算法

- A*搜索算法是由著名的人工智能学者Nilsson提出的，它是目前最有影响的启发式图搜索算法，也称为最佳图搜索算法。
- 如果某一问题有解，那么利用*A**搜索算法对该问题进行搜索则一定能搜索到解，并且一定能搜索到最优的解而结束。
- 定义 $h^*(n)$为状态n到目的状态的最优路径的代价，则当A搜索算法的启发函数$h(n)$小于等于$h^*(n)$，即满足 $h(n) \le h^*(n)$ ，对所有节点n时，被称为A*搜索算法。

4.5.5 蒙特卡洛树搜索算法

- 下棋思维过程：我们下棋时，每走一步前不可能在脑子里把所有可能的棋步都列出来，而是根据“棋感”大致筛选出了几种“最可能”的走法，然后再比较出自己感觉“最好”的走法。这就是蒙特卡洛树搜索算法的设计思路。
- 蒙特卡洛树搜索（Monte Carlo tree search，MCTS）算法是Remi Coulom于2006年在围棋人机对战引擎Crazy Stone中首次发明并使用的，其中使用蒙特卡洛法评价每个选择的优劣。
- 蒙特卡罗树搜索大概可以被分成4步：选择（Selection），拓展（Expansion），模拟（Simulation），反向传播（Backpropagation）。

4.5.5 蒙特卡洛树搜索算法

- 1. 选择：是根据当前获得所有子步骤的统计结果，选择一个最优的子步骤。
- 对于被检查到的结点可能有以下3种可能：
- （1）完全展开结点：该结点所有可行动作都已经被拓展过。那么计算该结点所有子结点的UCT值，并找到一个值最大的子结点继续检查。反复向下迭代。
- （2）未完全展开结点：该结点有可行动作还未被拓展过。例如有10个可行动作，但是在搜索树中才创建了9个子结点，那么就认为这个结点是本次迭代的目标结点N，并找出N还未被拓展的结点L执行“拓展”步骤。

4.5.5 蒙特卡洛树搜索算法

- （3）未访问结点：该结点所有可行动作都未被拓展过。那么从该结点直接执行“回溯”步骤。
- 在反复迭代之后，将在搜索树的底端找到一个结点，来继续后面的步骤。
- 改进算法：信任度上限树算法(Upper Confidence bound applied to Trees，UCT)，比蒙特卡罗规划更容易获得最优解。

$$UCT(v_i,v)=\frac{Q(v_i)}{N(v_i)}+c\sqrt{\frac{\log(N(v))}{N(v_i)}}$$

- 从根结点出发，每次选择己方UCT值最优的结点，向下搜索一个“未完全展开结点”，然后随机选一个结点进行扩展。上式第一部分是模拟的胜率；随着访问次数的增加，第二部分的值会越来越小。因此，会倾向于选择那些还没有怎么被统计过的结点。

4.5.5 蒙特卡洛树搜索算法

- 2. 拓展：在当前获得的统计结果不足以计算出下一个步骤时，随机选择一个子步骤。
- 3. 模拟：从新结点M出发，模拟拓展搜索树，直到找到一个终止结点。为了让新的结点M得到一个初始的评分，从M开始让游戏随机进行，直到得到一个游戏结局，这个结局将作为M的初始评分。
- 4. 反向传播：在M的模拟结束后，它的父结点N以及从根结点到N的路径上的所有结点都会根据本次模拟的结果来添加自己的累计评分。如果在“选择”步骤中直接发现了一个游戏结局的话，根据该结局来更新评分。

THE END