

---

## 第5章 搜索求解策略



# 传教士与野人问题

■ [http://www.4399.com/flash/77287\\_2.htm](http://www.4399.com/flash/77287_2.htm)

- ▶ [Saul Amarel (1968), "On representations of problems of reasoning about actions", in "Machine Intelligence 3", pp131-171, Elsevier.]

Press on Play

Please help the 3 cannibals  
and the 3 missionaries to move  
to the other side of the lake.

notice that: when there is on one side  
more cannibals than  
missionaries, they eat them.

play

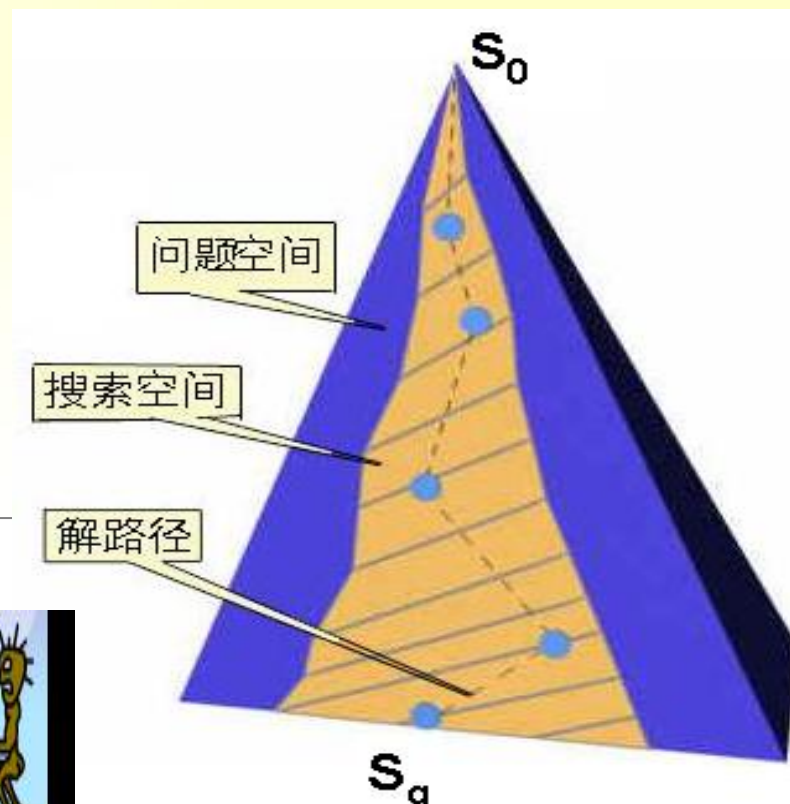
右岸



左岸

# 5.1 搜索的概念

- 搜索法：如何在一个比较大的问题空间中，只通过搜索比较小的范围，就能找到问题的解。
- ◆ 搜索什么（目标）
- ◆ 在哪里搜索（搜索空间）
- 使用不同的搜索策略，找到解的搜索空间范围是有区别的。

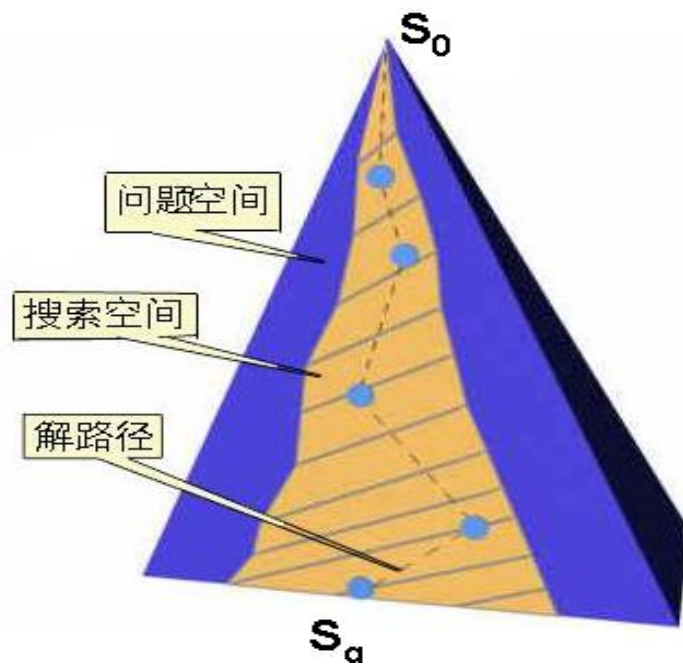
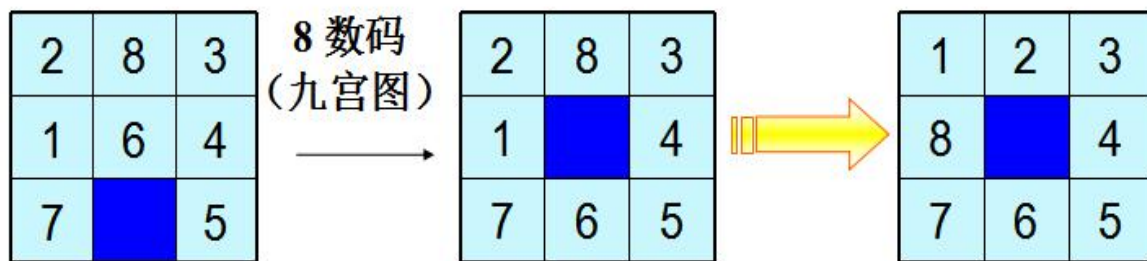


搜索空间示意图



# 5.1 搜索的概念

- 通常搜索策略的主要任务是确定如何选取规则的方式
  - 盲目搜索: 不具有特定问题有关信息的条件下, 按固定的步骤 (依次或随机调用操作算子) 进行的搜索。
  - 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, 以求尽快地到达目标状态的搜索。



搜索空间示意图

# 5.1 搜索的概念

- 根据问题的表示方式，可分为状态空间搜索和与/或树搜索。
- 根据搜索方向，可分为正向搜索、逆向搜索和双向搜索。
- 一般搜索策略的评价准则（**Stuart Russell and Peter Norving, 1995**):
  - **完备性**：如果存在一个解，该策略是否保证能够找到？
  - **时间复杂性**：需要多长时间可以找到解？
  - **空间复杂性**：执行搜索需要多少存储空间？
  - **最优性**：如果存在不同的几个解，该策略是否可以发现最高质量的解？

# 第5章 搜索求解策略

## ■ 5.1 搜索的概念

## ■ 5.2 状态空间表示及状态空间图

## ■ 5.3 盲目的图搜索策略

## ■ 5.4 启发式图搜索策略

➤ 了解搜索的基本概念，掌握状态空间表示及启发式图搜索策略（A\*搜索等），了解A\*搜索、 $\alpha$ - $\beta$ 剪枝搜索等应用。

➤ 重点：A\*搜索、Min-Max搜索、 $\alpha$ - $\beta$ 剪枝搜索

➤ 难点：启发函数的确定、 $\alpha$ - $\beta$ 剪枝搜索



## 5.2 状态空间表示及状态空间图

- 5.2.1 状态空间表示法
- 5.2.2 状态空间的图描述

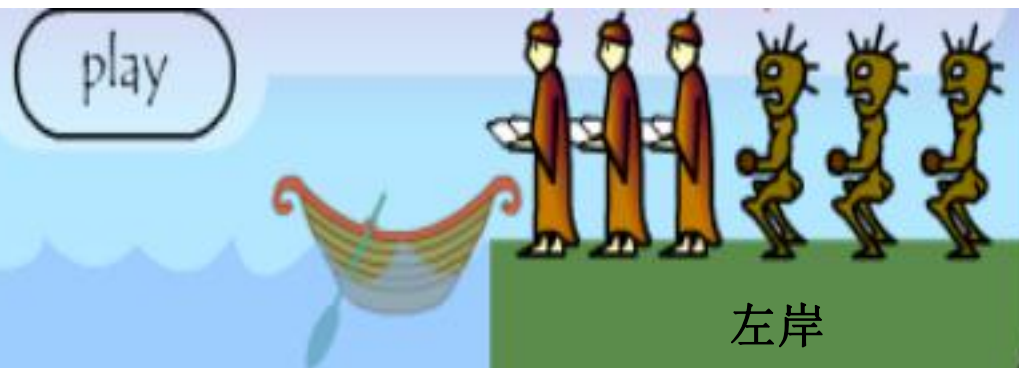
■ 状态空间表示法：表示问题及其搜索过程的一种方法

- 问题：用状态和操作表示
- 问题的求解过程：用状态空间的搜索表示

## 5.2.1 状态空间表示法

■ **状态(State)**: 表示系统状态、事实等叙述型知识的一组变量或数组, 也是描述问题求解过程中任意时刻的数据结构。

状态  $(x, y, z)$ :  $x$ 表示左岸传教士数,  $y$ 表示左岸野人数,  $z$ 表示船的情况 (1: 左岸, 0: 右岸)



目标状态:  $(0, 0, 0)$

**LR(1,1)**

初始状态:  $(3, 3, 1)$

操作 **LR(m, c)**: 从左岸到右岸、操作 **RL(m, c)**: 从右岸到左岸  
( $m$ : 传教士数,  $c$ : 野人数)

■ **操作**: 表示引起状态变化的过程型知识的一组关系或函数。

- **操作符 (算符、操作算子)**: 走步、过程、规则、数学算子、运算符号或逻辑符号等。



## 5.2.1 状态空间表示法

■ **状态(State)**: 表示系统状态、事实等叙述型知识的一组变量或数组, 也是描述问题求解过程中任意时刻的数据结构。

**状态  $(x,y,z)$** :  $x$ 表示左岸传教士数,  $y$ 表示左岸野人数,  $z$ 表示船的情况 (1: 左岸, 0: 右岸)

右岸



左岸

状态1:  $(2, 2, 0)$

**LR(1,1)**

初始状态:  $(3,3, 1)$

**操作 LR(m, c)**: 从左岸到右岸、**操作 RL(m, c)**: 从右岸到左岸  
( $m$ : 传教士数,  $c$ : 野人数)

■ **操作**: 表示引起状态变化的过程型知识的一组关系或函数。

- **操作符 (算符、操作算子)**: 走步、过程、规则、数学算子、运算符号或逻辑符号等。

## 5.2.1 状态空间表示法

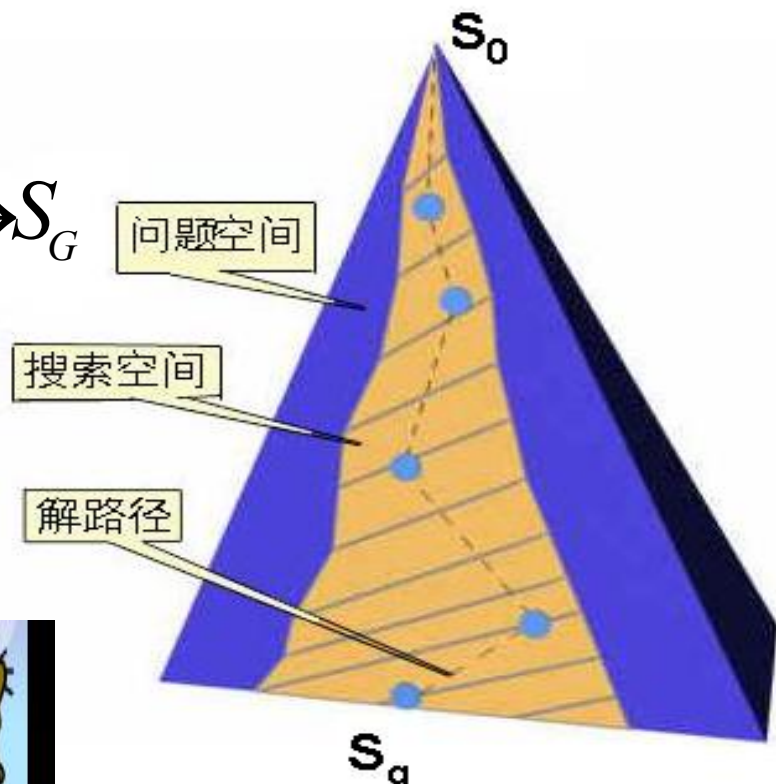
■ **状态空间(State Space)**: 利用状态和操作表示问题的有关知识的符号体系。

■ **解路径**:

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

■ 状态空间的一个解就是一个有限的操作算子序列:

$$O_1, \cdots, O_k$$



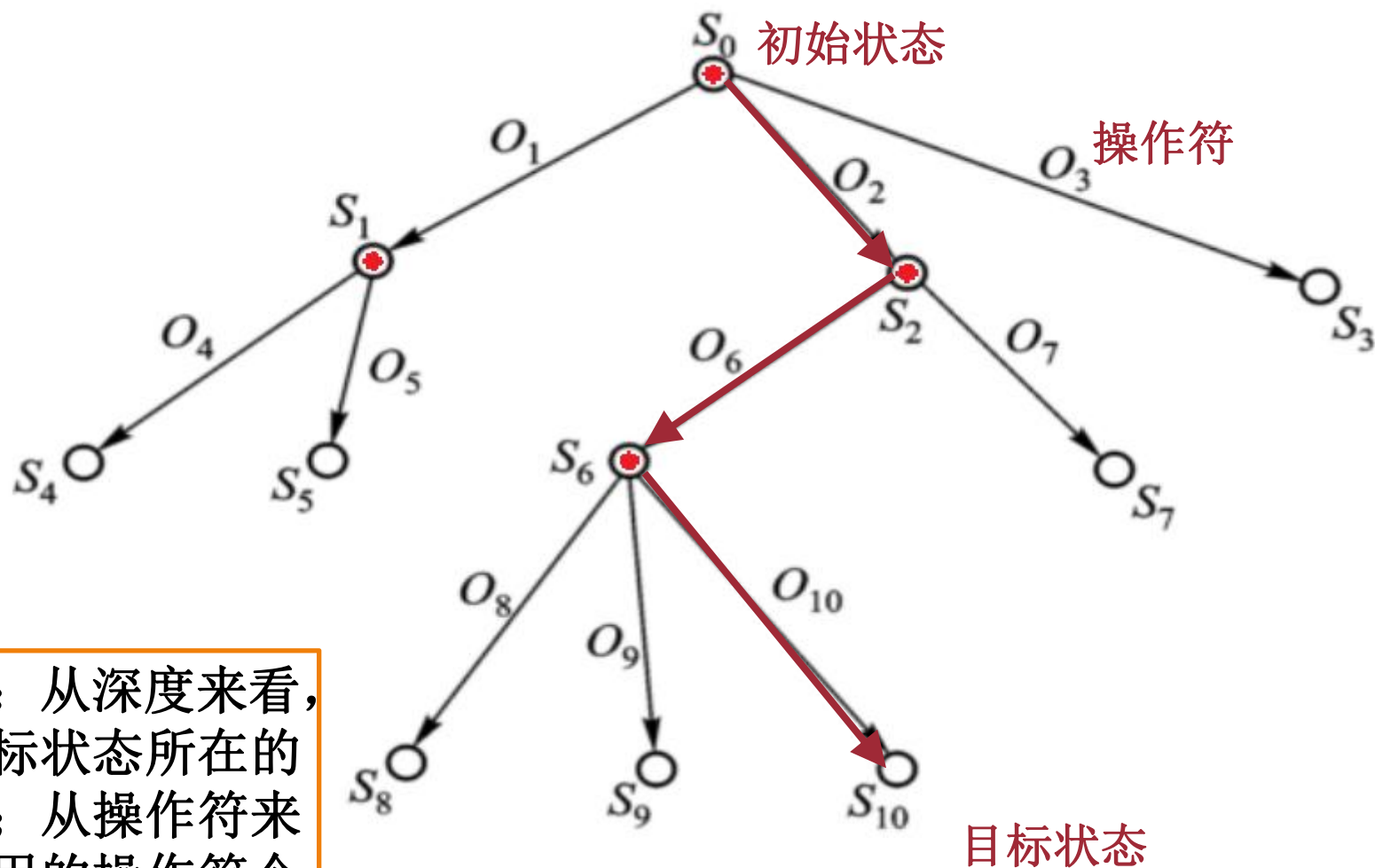
搜索空间示意图

## 5.2 状态空间表示及状态空间图

- 5.2.1 状态空间表示法
- 5.2.2 状态空间的图描述

图：由一系列节点（**node**）和弧（**arc**）或连接（**link**）构成的。

## 5.2.2 状态空间的图描述



最短路径：从深度来看，找到的目标状态所在的深度最小；从操作符来看，所采用的操作符个数最少。

## 5.2.2 状态空间的图描述

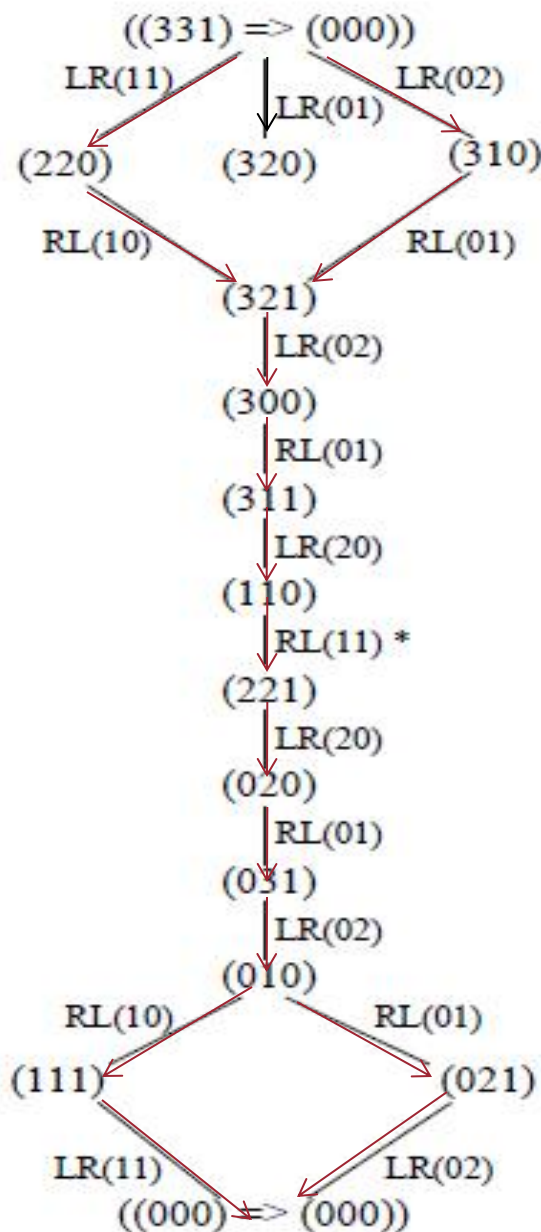
传教士与野人问题  
( $N=3$ ,  $K \leq 2$ )  
状态空间图

状态  $(x, y, z)$ :

$x$ 表示左岸传教士数,

$y$ 表示左岸野人数,

$z$ 表示船的情况 (1:  
左岸, 0: 右岸)

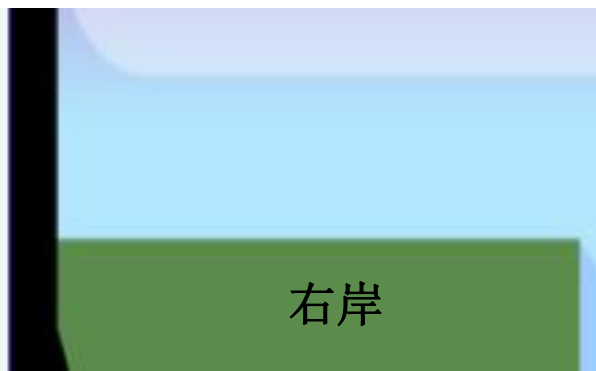


操作  $LR(m, c)$ : 从左岸到  
右岸

操作  $RL(m, c)$ : 从右岸到  
左岸

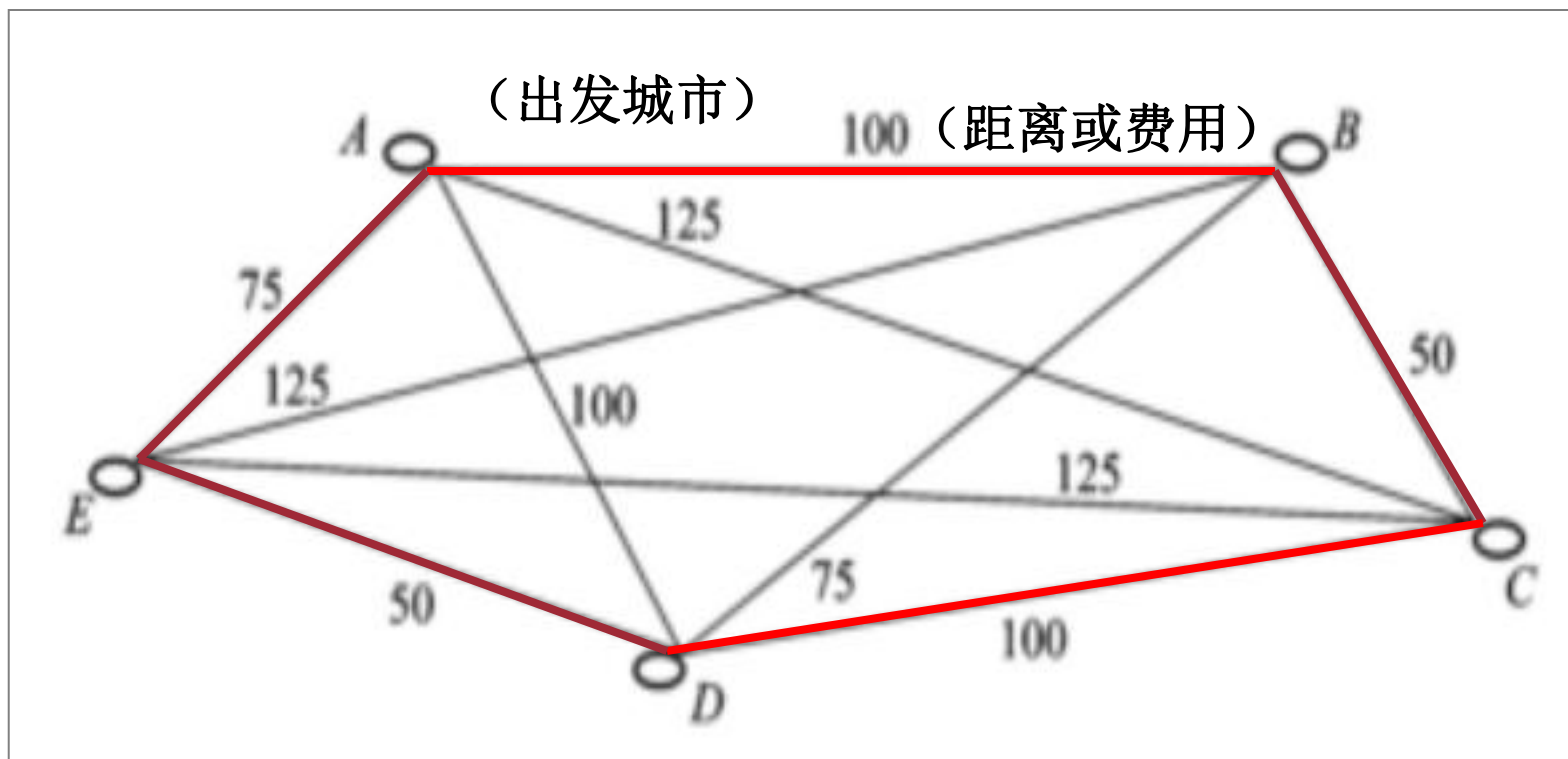
$m$ : 传教士数

$c$ : 野人数



## 5.2.2 状态空间的图描述

例5.3 旅行商问题（traveling salesman problem, TSP）或邮递员路径问题。

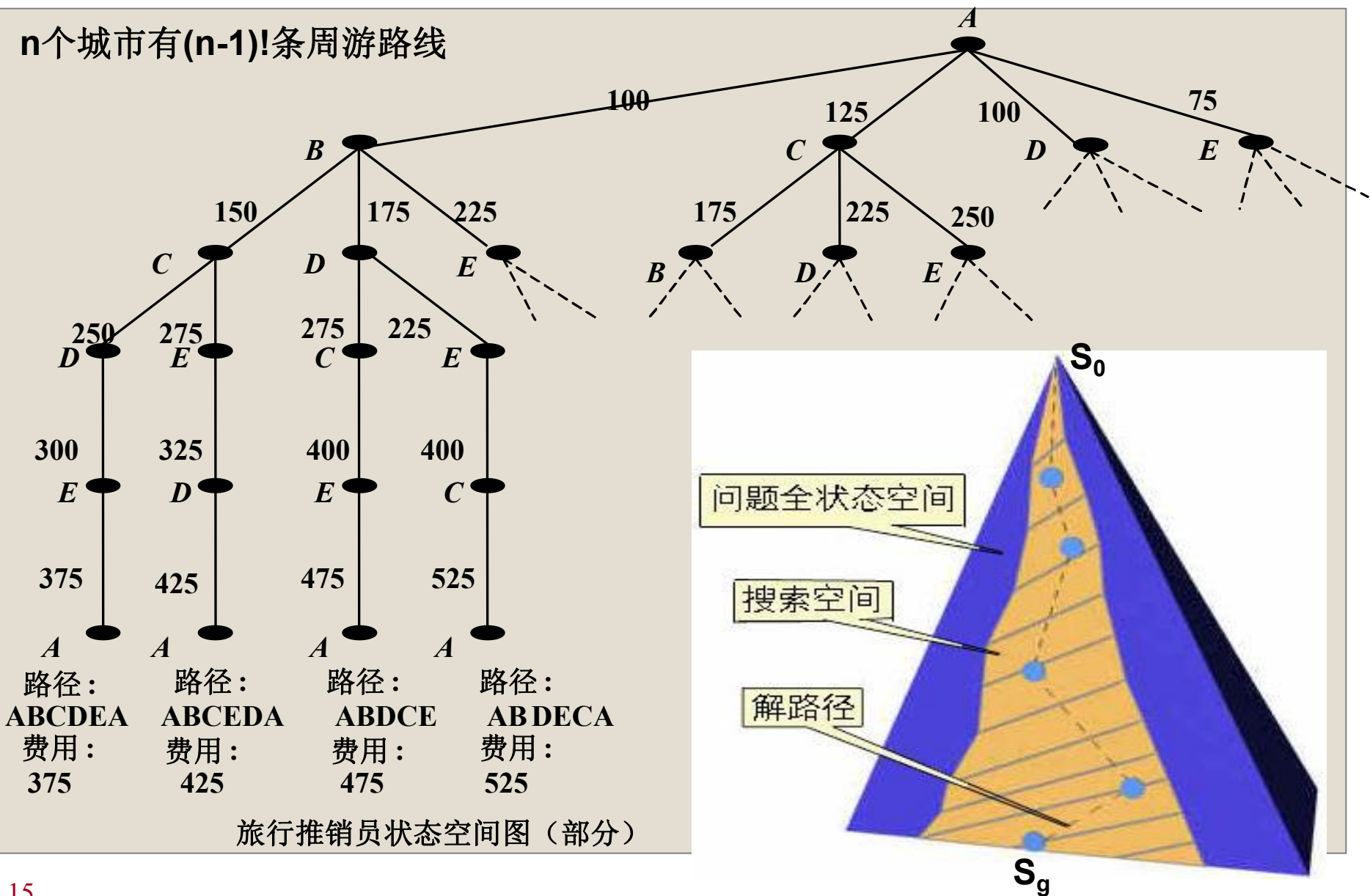


可能路径：费用为375的路径（ $A, B, C, D, E, A$ ）



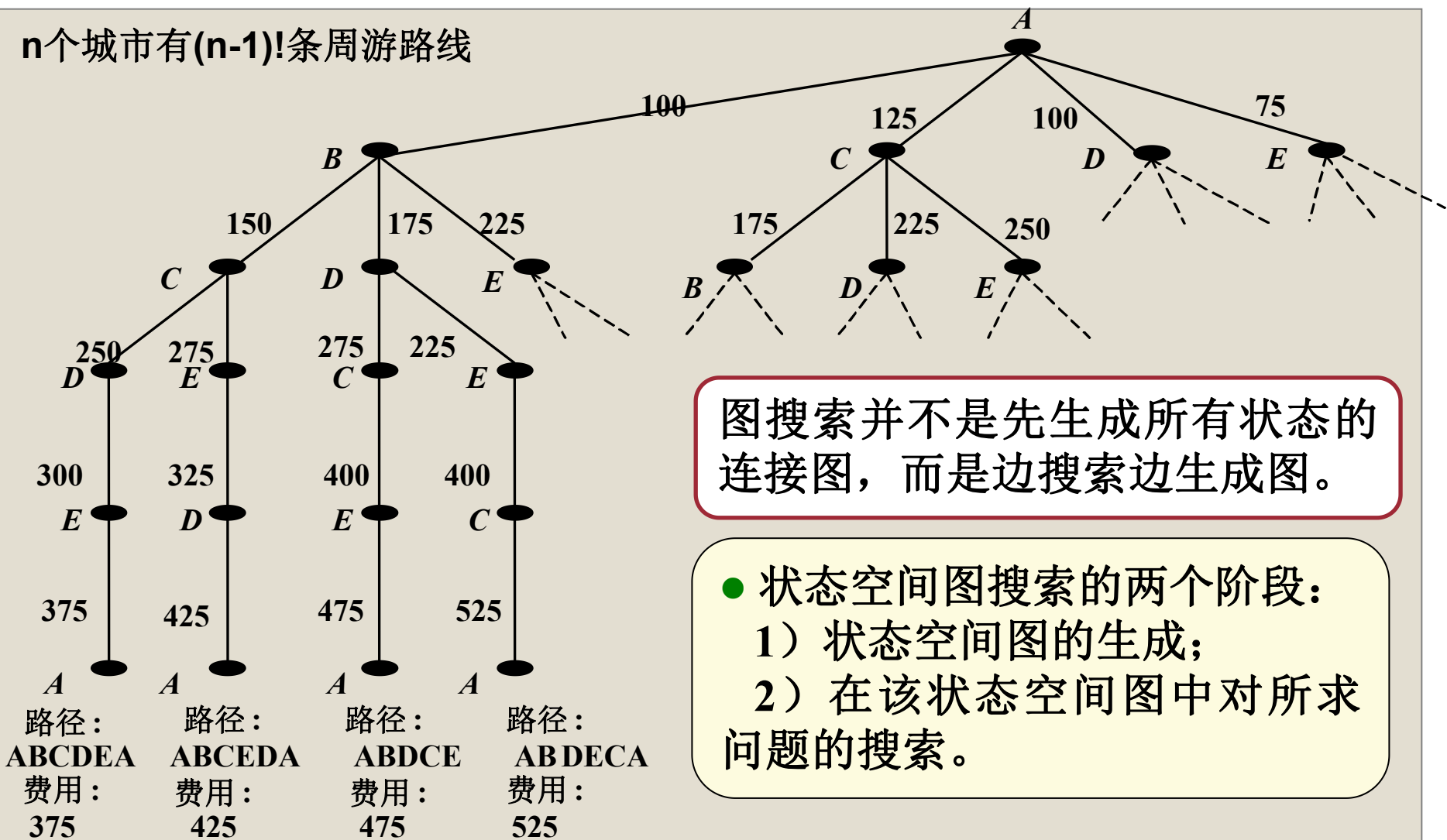
## 5.2.2 状态空间的图描述

$n$ 个城市有 $(n-1)!$ 条周游路线



## 5.2.2 状态空间的图描述

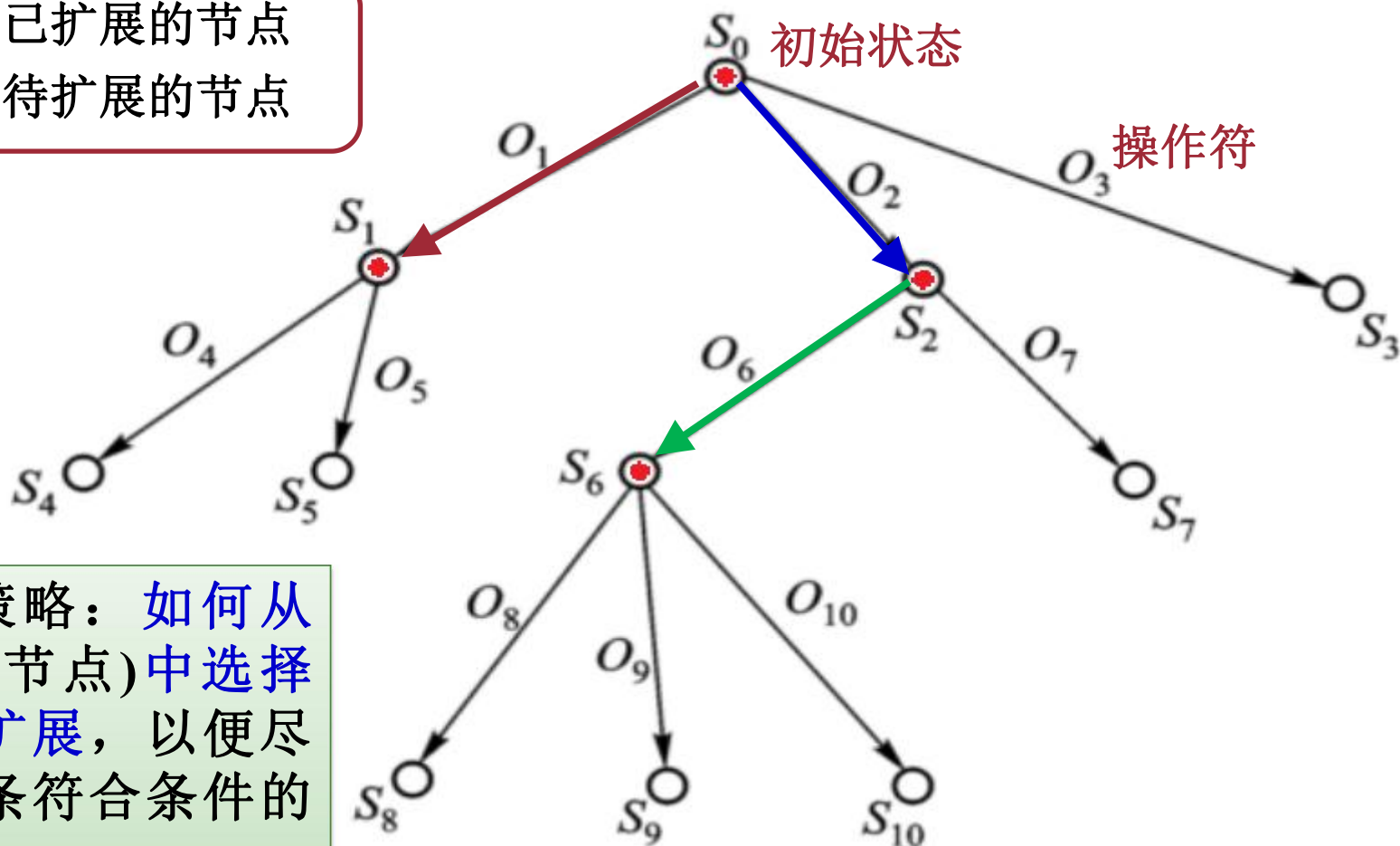
$n$ 个城市有 $(n-1)!$ 条周游路线



旅行推销员状态空间图（部分）

## 5.2.2 状态空间的图描述

- 红心圆：已扩展的节点
- 空心圆：待扩展的节点



➤图搜索策略：如何从空心圆(叶节点)中选择一个节点扩展，以便尽快找到一条符合条件的路径。

➤不同的选择方法构成了不同的图搜索策略。

状态空间的有向图描述

# 第5章 搜索求解策略

## ■ 5.1 搜索的概念

## ■ 5.2 状态空间表示及状态空间图

## ■ 5.3 盲目的图搜索策略

## ■ 5.4 启发式图搜索策略

如果在搜索过程中没有利用与问题有关的知识或者启发信息，则称之为盲目搜索。

## 5.3 盲目的图搜索策略

- 5.3.1 宽度优先搜索策略
- 5.3.2 深度优先搜索策略

## 5.3.1 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。

- 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**。

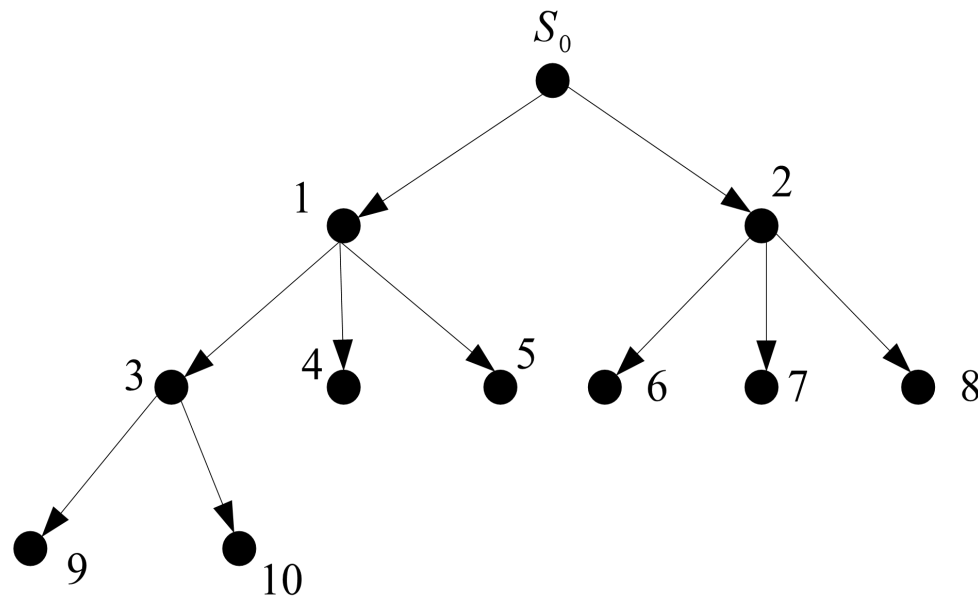
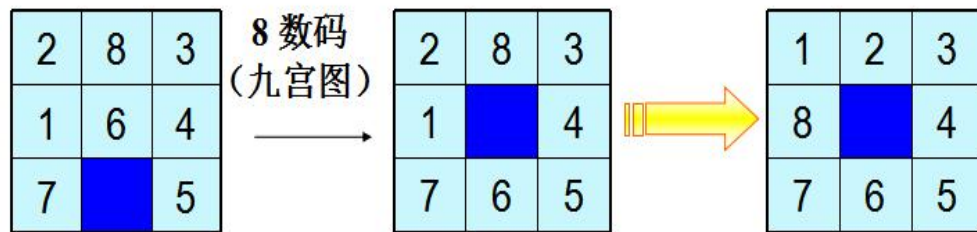


图5.6 宽度优先搜索法中状态的搜索次序

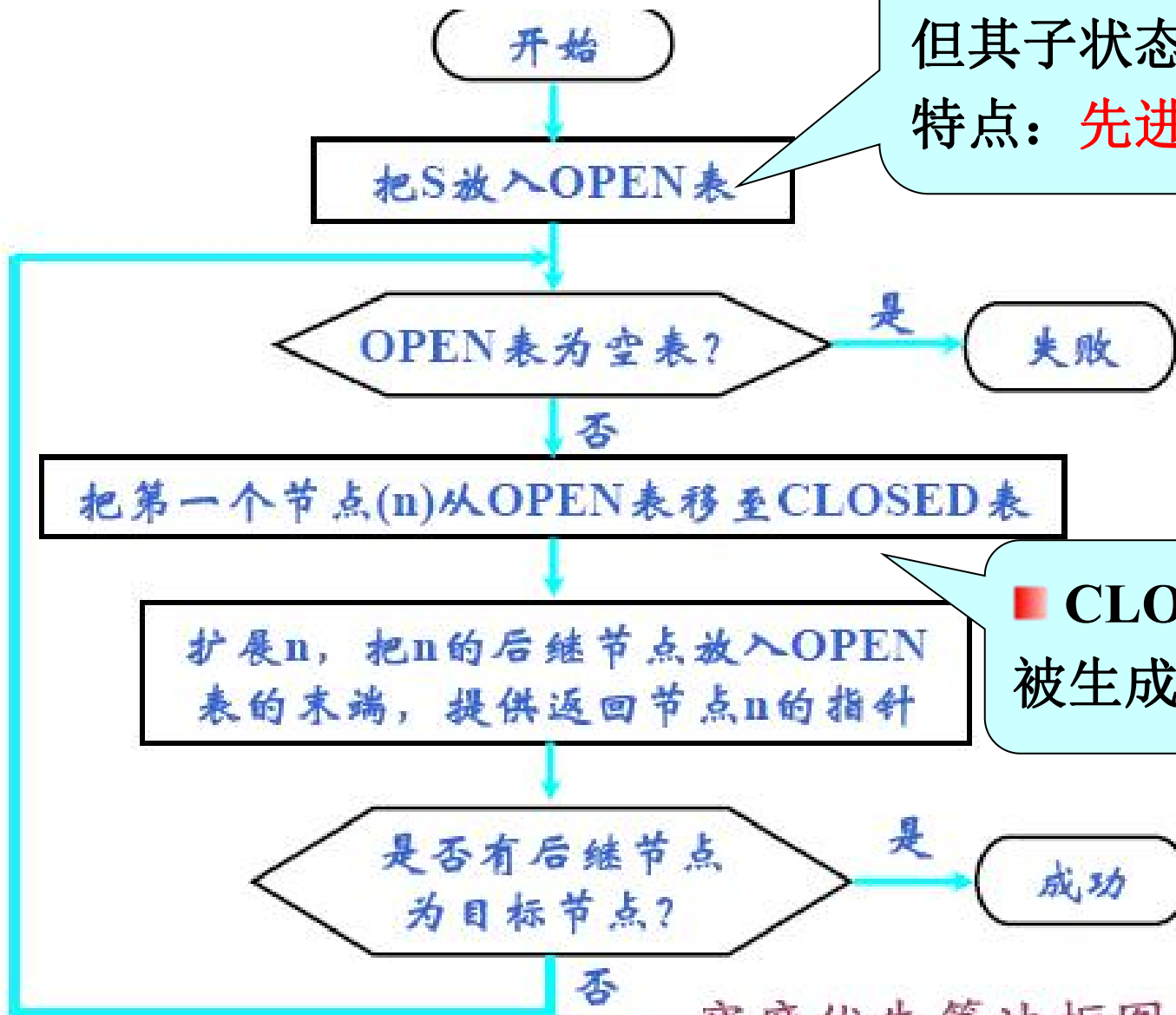
同层节点的深度一样，会按照其出现的先后次序搜索，先生成的先搜索，**生成的次序就取决于使用操作算子的次序**，（即规则的使用次序）可以是固定的，也可以是随机的。





## 5.3.1 宽度优先搜索策略

■ OPEN表：已经生成出来但其子状态未被扩展的状态，特点：**先进先出**。



■ CLOSED表：记录已被生成扩展过的状态。

宽度优先算法框图

## 5.3.1 宽度优先搜索策略

■ 例5.4 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。

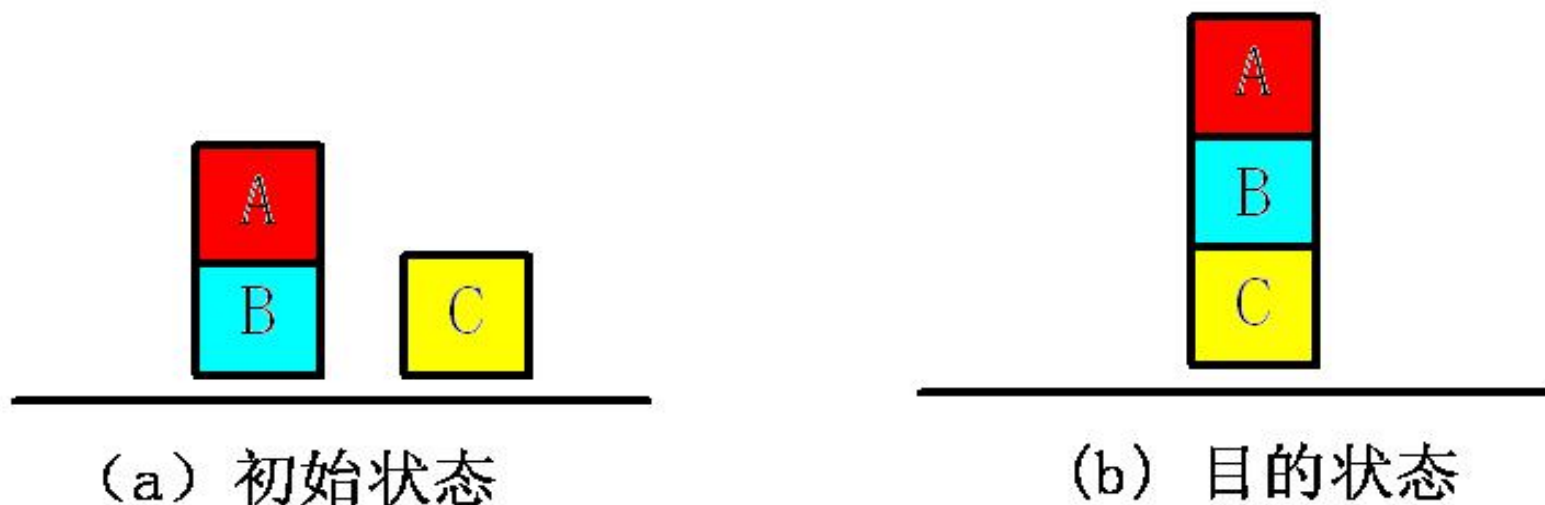


图5.7 积木问题

状态：三元组  $(X, Y, Z)$  表示积木A, B, C在哪个积木或桌子上。  
初始状态：  $(B, \text{table}, \text{table})$ ；目标状态：  $(B, C, \text{table})$ 。

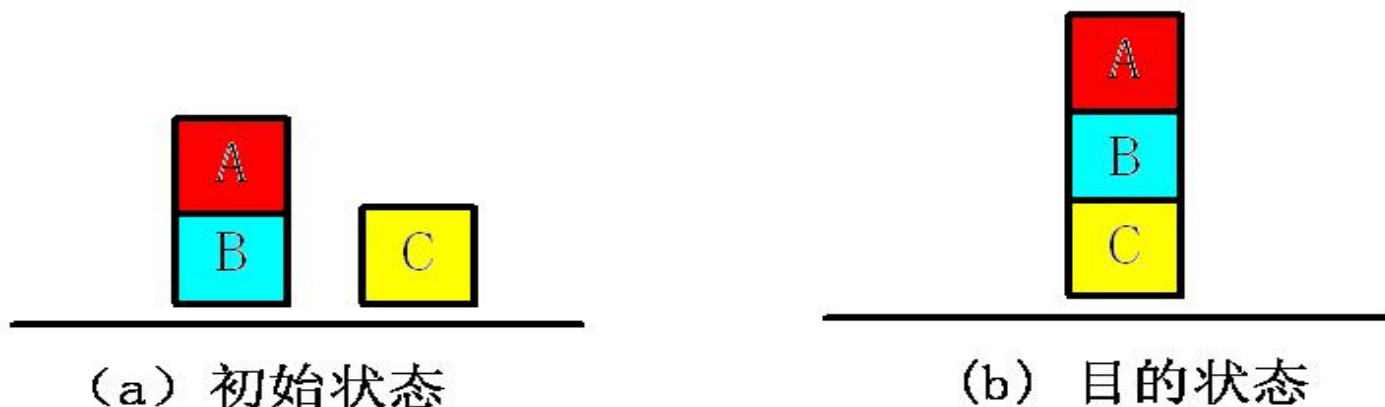
## 5.3.1 宽度优先搜索策略

- 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。



## 5.3.1 宽度优先搜索

问题：搜索结束时，**Open**表和**Closed**表包含哪些状态？扩展节点数和生成节点数分别是多少？

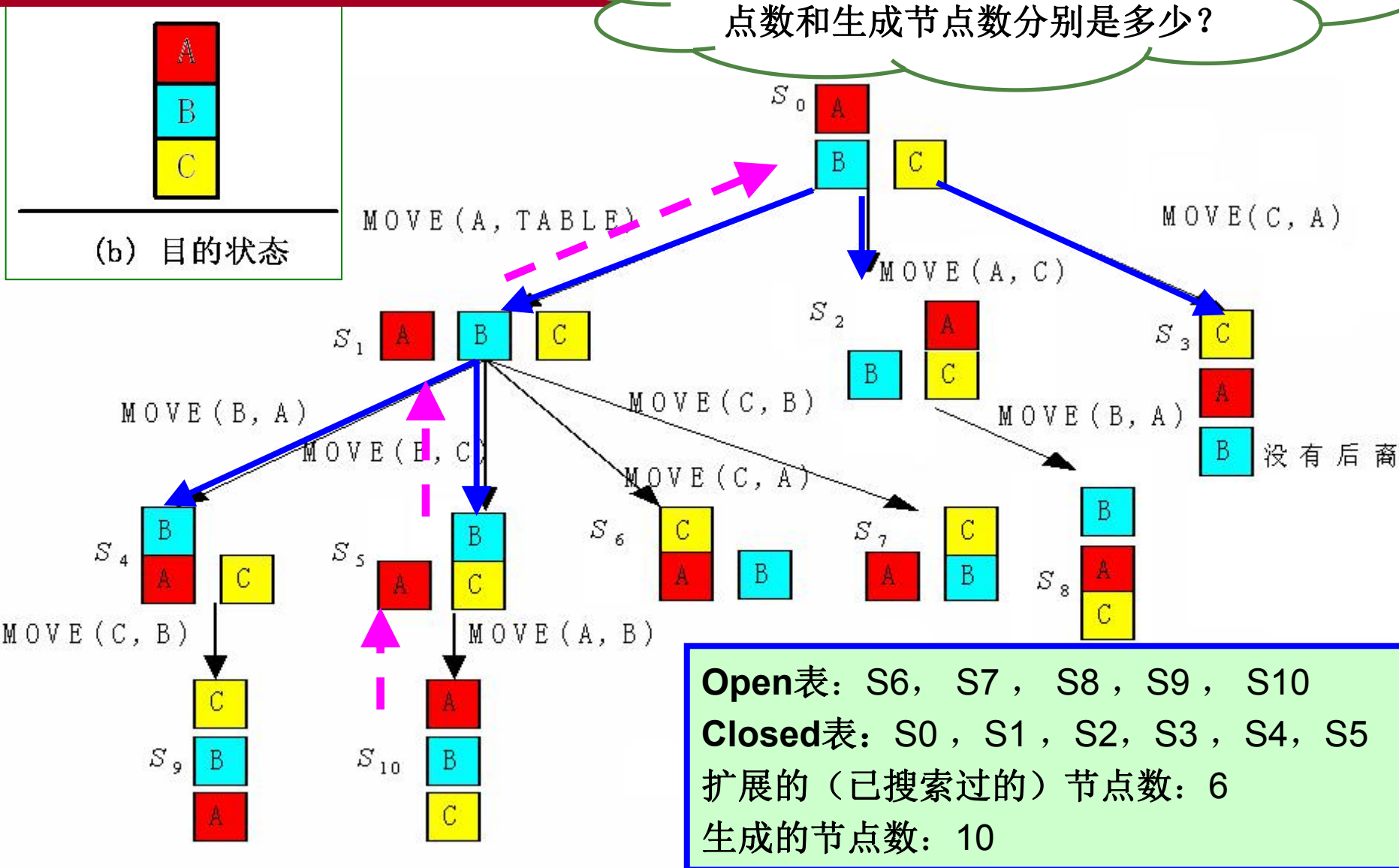


图 5.8 积木问题的宽度优先搜索树

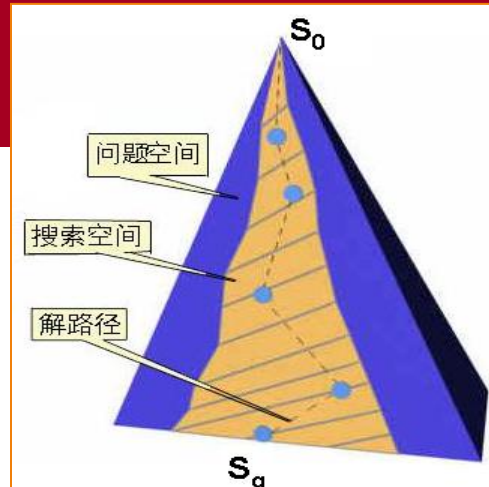
# 第5章 搜索求解策略

## 5.1 搜索的概念

## 5.2 状态空间表示及状态空间图

## 5.3 盲目的图搜索策略

## 5.4 启发式图搜索策略



➤ 了解搜索的基本概念，掌握状态空间表示及启发式图搜索策略（A\*搜索等），了解A\*搜索、 $\alpha$ - $\beta$ 剪枝搜索等应用。

➤ 重点：A\*搜索、Min-Max搜索、 $\alpha$ - $\beta$ 剪枝搜索

➤ 难点：启发函数的确定、 $\alpha$ - $\beta$ 剪枝搜索

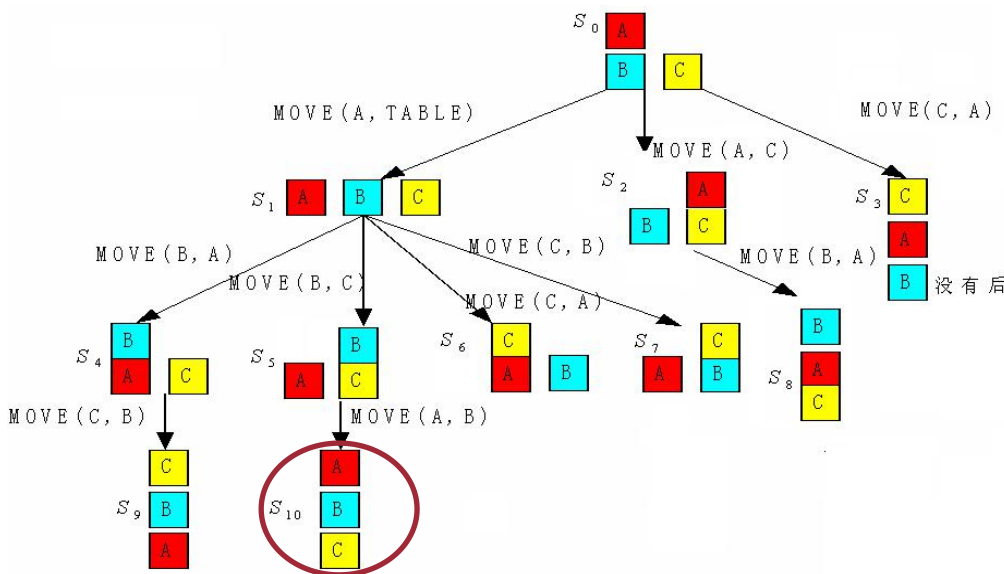


图 5.8 积木问题的宽度优先搜索树

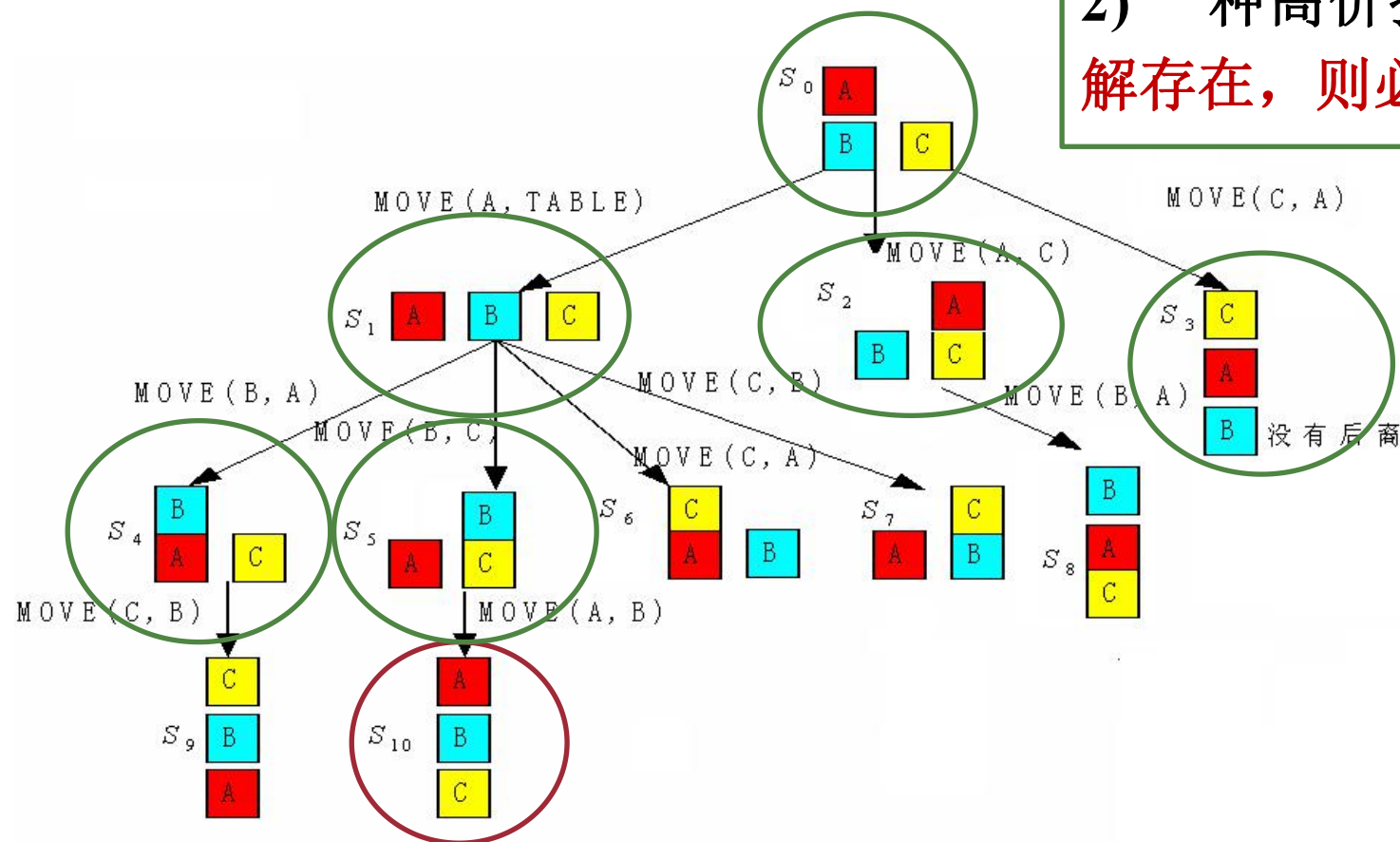
# 5.3 盲目的图搜索策略

## ■ 5.3.1 宽度优先搜索策略

## ■ 5.3.2 深度优先搜索策略

1) 每次选择深度最浅的节点首先扩展，搜索是逐层进行的；

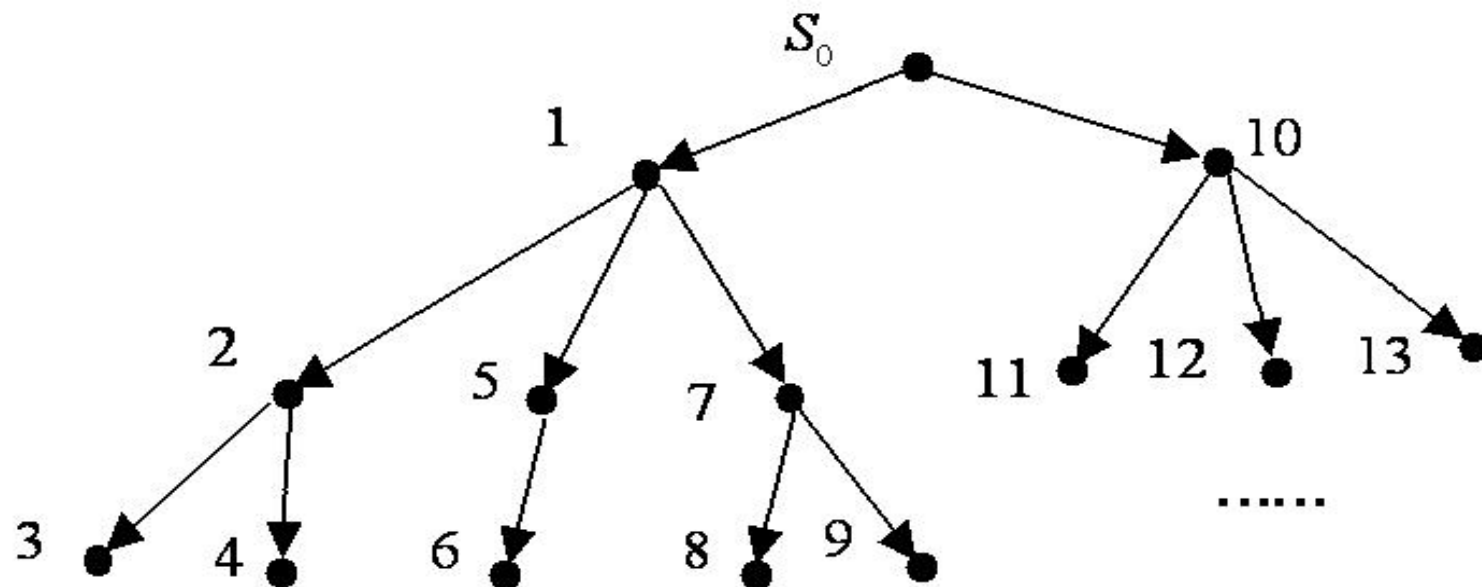
2) 一种高价搜索，但若解存在，则必能找到它。





## 5.3.2 深度优先搜索策略

■ **深度优先搜索 (Depth-first Search):** 先扩展最新生成的节点, 深度相等的节点按生成次序 (后生成排前面) 的盲目搜索。

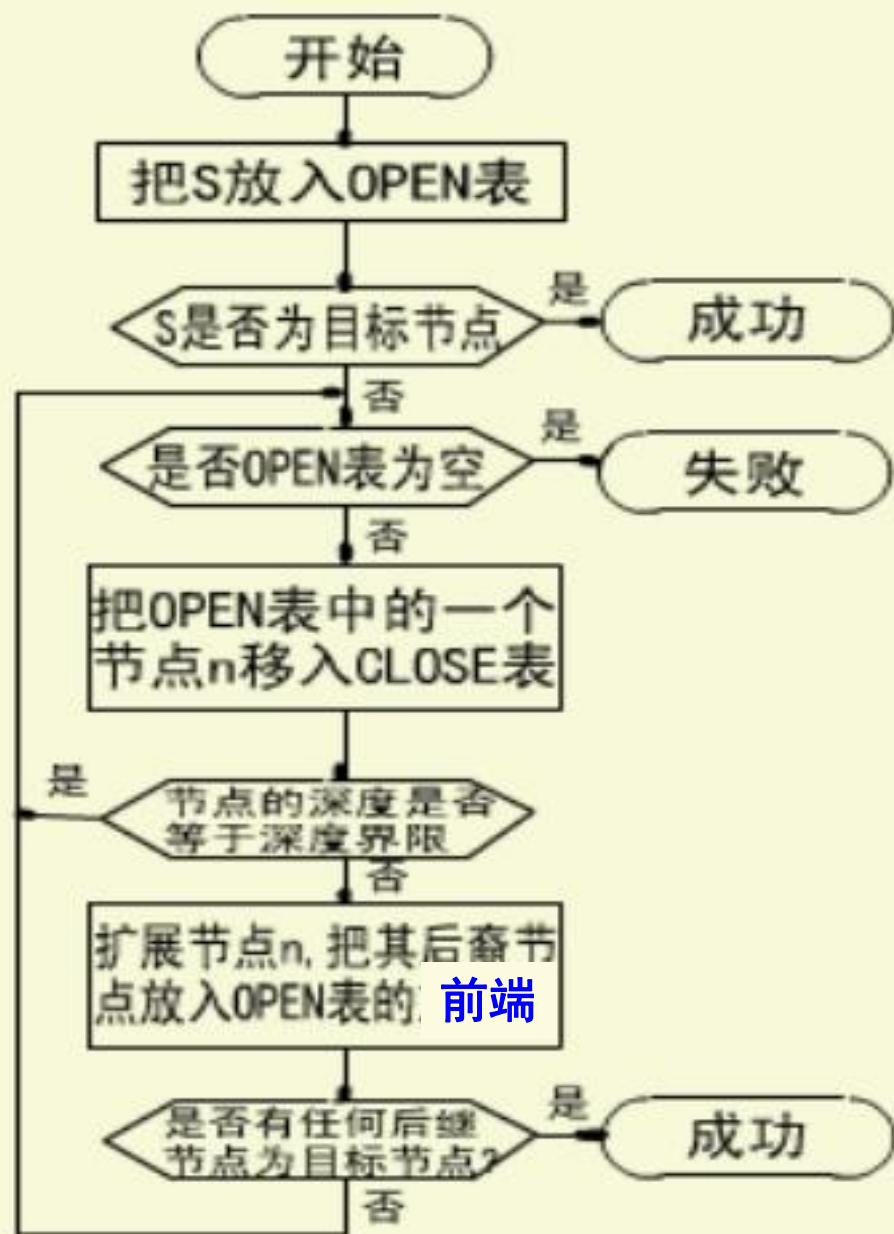


■ **特点:** 扩展最深的节点的结果使得搜索沿着状态空间某条单一的路径从起始节点向下进行下去; 仅当搜索到达一个没有后裔的状态时, 才考虑另一条替代的路径。深度优先搜索不是完备的。

## 5.3.2 深度优先搜索策略

### ■ 算法:

- 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的最大深度——**深度界限**；
- 与宽度优先搜索算法最根本的不同：**将扩展的后继节点放在OPEN表的前端**。
- 深度优先搜索算法的**OPEN**表后进先出。



## 5.3.2 深度优先搜索策略（课堂测试）

问题：搜索结束时，**Open**表和**Closed**表包含哪些状态？扩展节点数和生成节点数分别是多少？

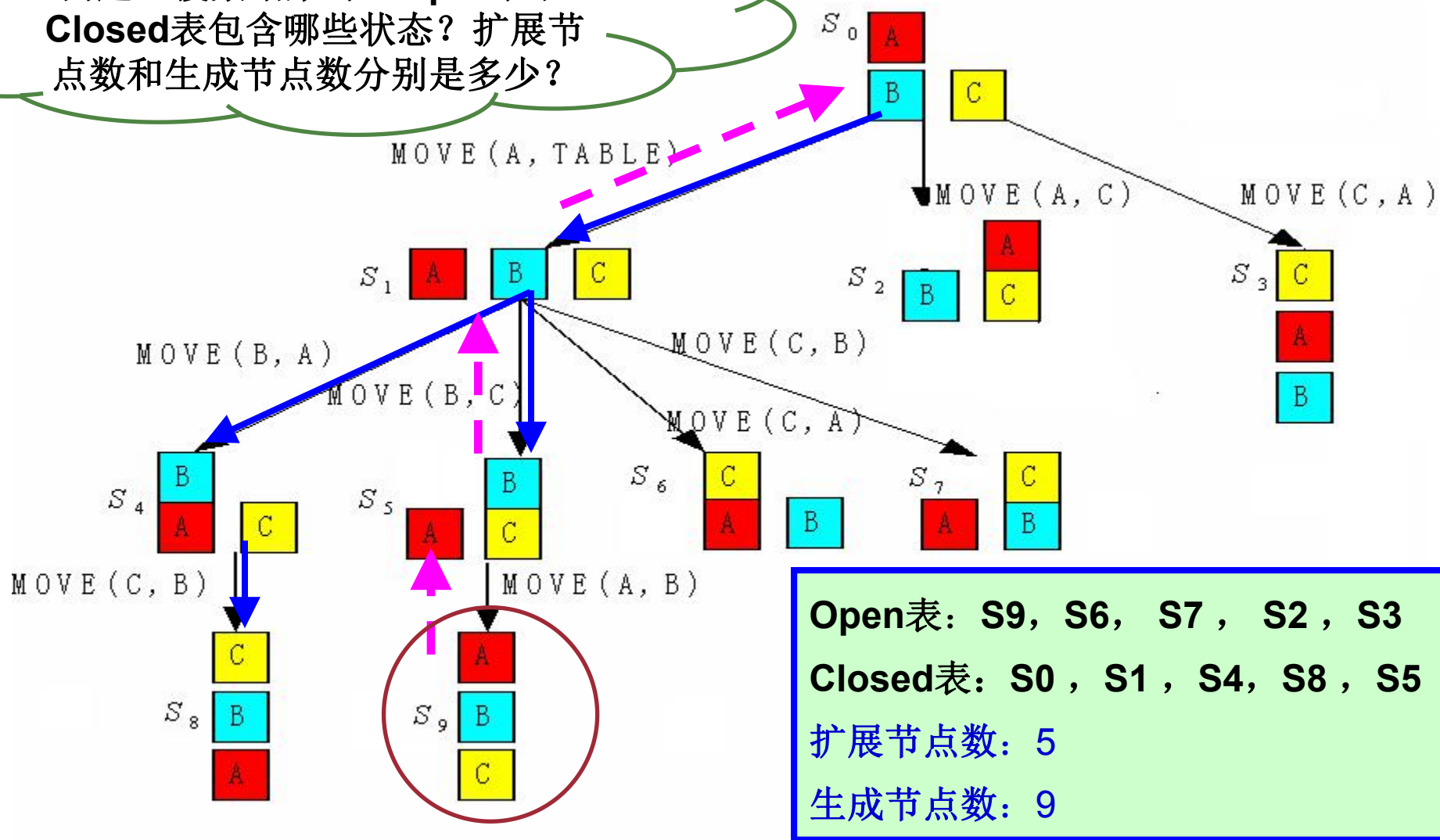


图 5.8 积木问题的深度优先搜索树

## 5.3.1 宽度优先搜索

问题：搜索结束时，**Open**表和**Closed**表包含哪些状态？扩展节点数和生成节点数分别是多少？

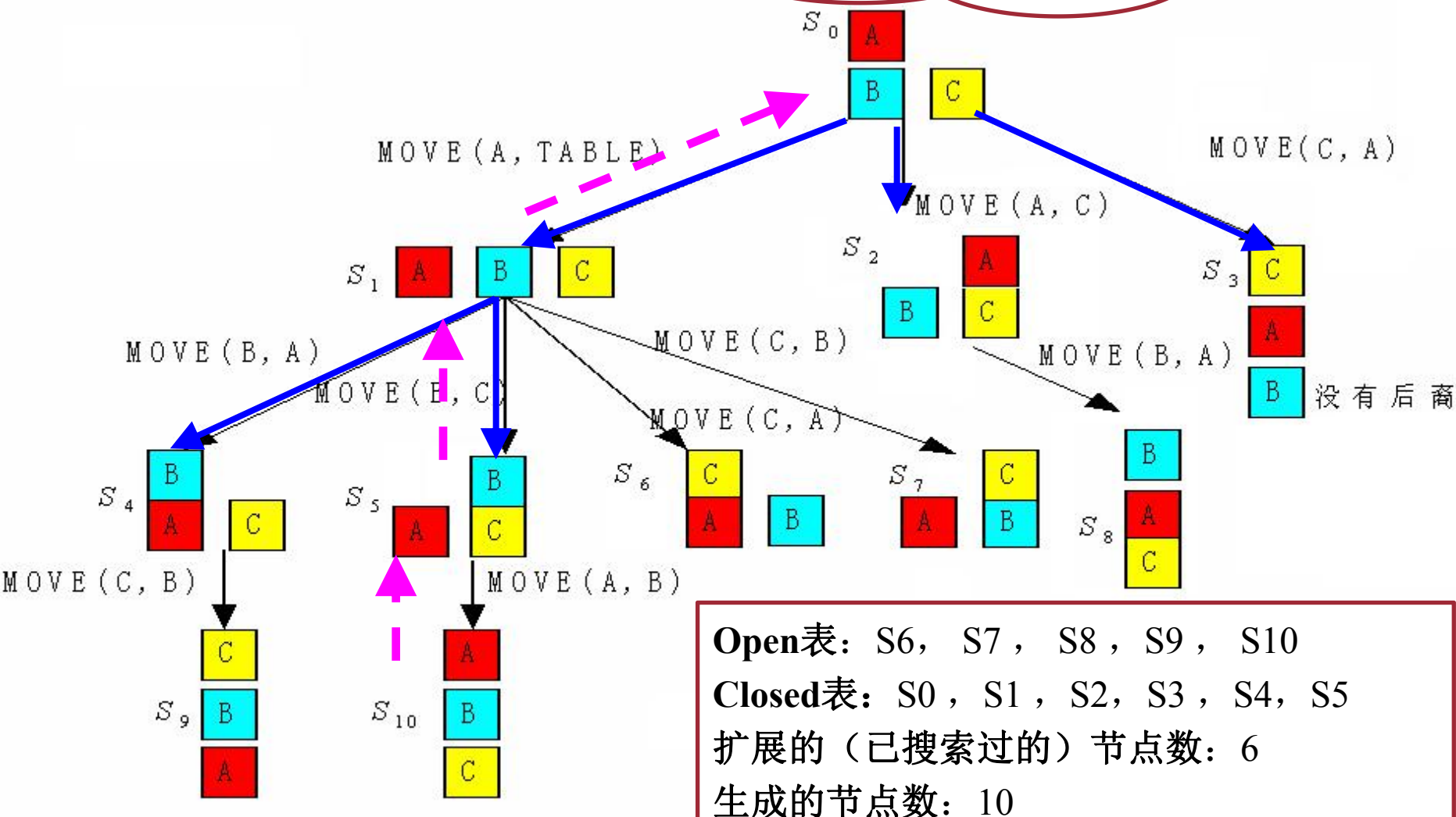


图 5.8 积木问题的宽度优先搜索树

## 5.3 盲目的图搜索策略

### ■ 5.3.1 宽度优先搜索策略

### ■ 5.3.2 深度优先搜索策略

如何选择合适的深度限制值？

完备的，能保证最优解（问题为单位耗散值），搜索过程中需保留已有的搜索结果，所需存储空间随着搜索深度呈几何级数增加

不能保证最优解，但可采用回溯，只保留从初始节点到当前节点一条路径，所需存储空间与搜索深度呈线性关系

### ■ 迭代加深搜索：

- 目的：解决宽度优先策略的空间问题和深度优先策略不能找到解的问题。
- 思想：首先给深度优先搜索一个比较小的深度限制，然后逐渐增加深度限制，直到找到解或找遍所有分支为止；
- 迭代加深搜索是完备的，能保证最优解（问题为单位耗散值）。

# 第5章 搜索求解策略

## ■ 5.1 搜索的概念

## ■ 5.2 状态空间表示及状态空间图

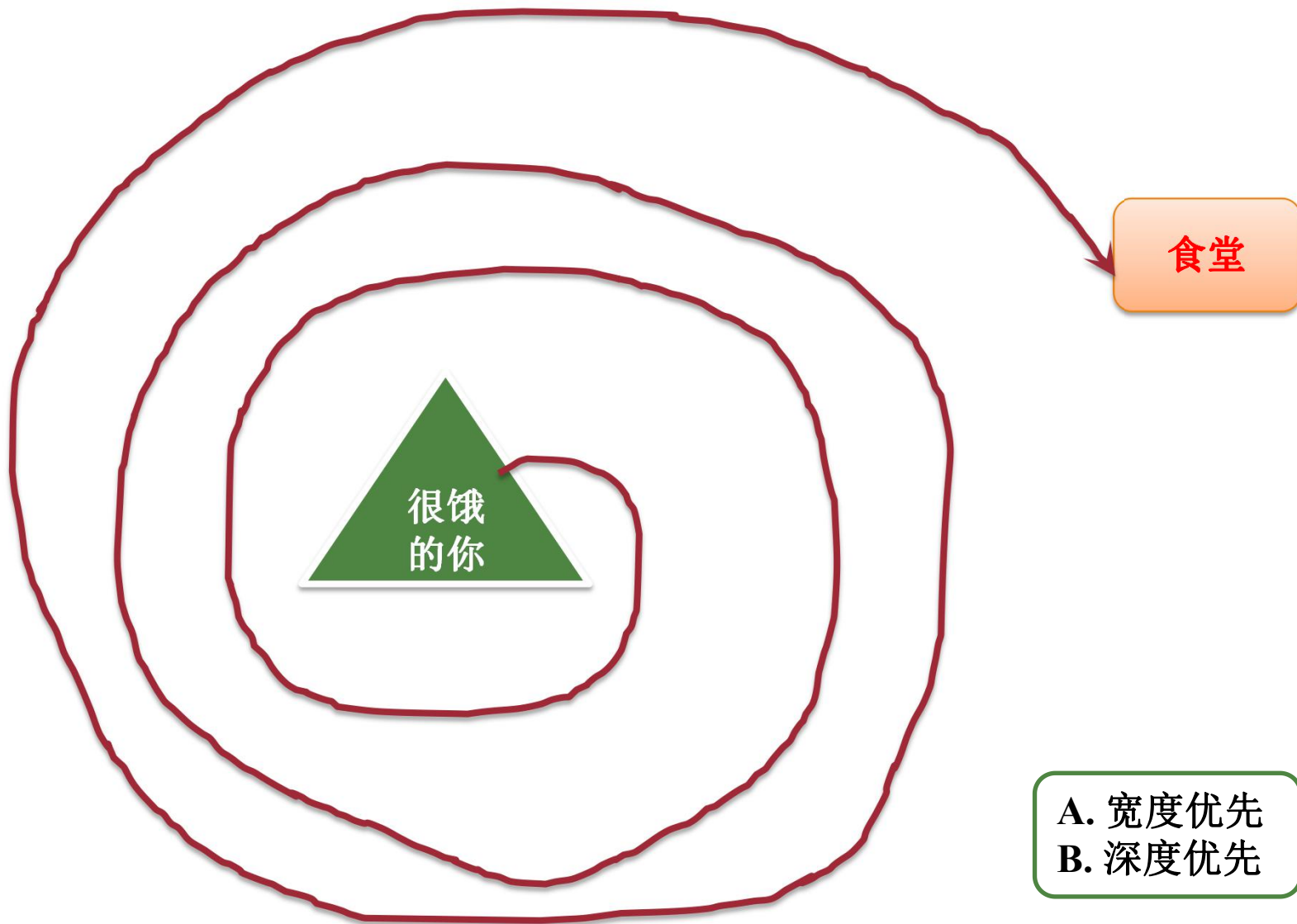
## ■ 5.3 盲目的图搜索策略

## ■ 5.4 启发式图搜索策略

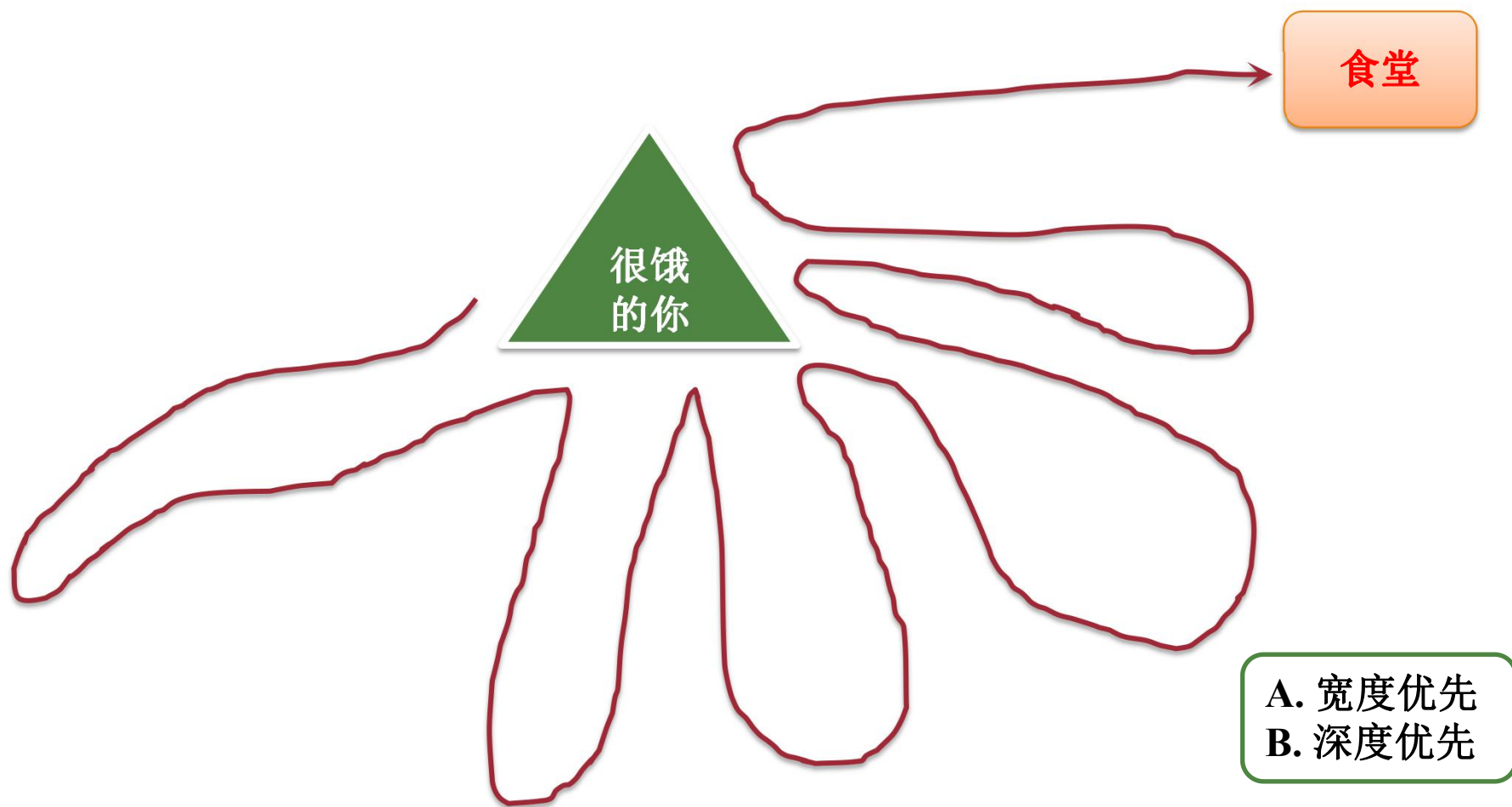
- 宽度优先、深度优先：  
不需重排OPEN表。
- 盲目搜索：
  - 没有启发信息的一种搜索形式；
  - 一般只适用求解比较简单的问题。



## 5.4 启发式图搜索策略



## 5.4 启发式图搜索策略



## 5.4 启发式图搜索策略

- 怎么样才能在中饭前找到食堂吃中饭呢？
  - 走几步，问一下路上同学食堂怎么走。
  - 抬头闻闻空气中的饭香味，哪个方向香就往哪里走。
  - 打开手机地图查一下。
  - .....



## 5.4 启发式图搜索策略

### ■ 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。

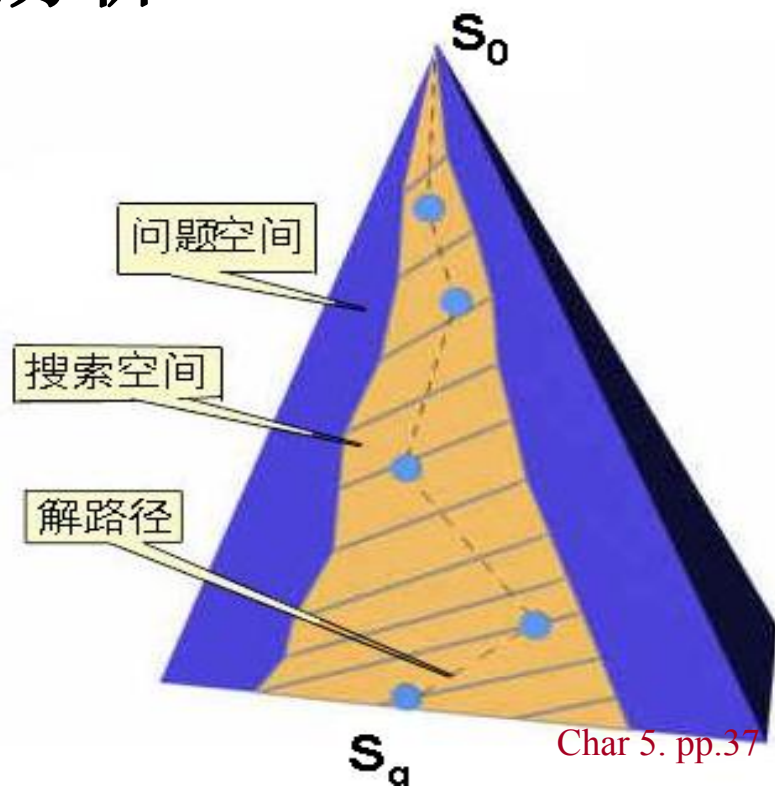


# 5.4 启发式图搜索策略

- 5.4.1 启发信息和估价函数
- 5.4.2 A搜索算法
- 5.4.3 A\*搜索算法及其特性分析
- 补充：博弈搜索策略

➤ 重点：A\*搜索、Min-Max搜索、 $\alpha$ - $\beta$ 剪枝搜索

➤ 难点：估价函数中启发函数的确定、 $\alpha$ - $\beta$ 剪枝搜索



## 5.4.1 启发信息和估价函数

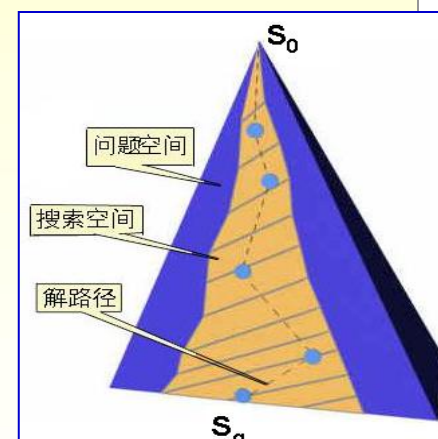
### ■ 按运用的方法分类:

- 1) **陈述性启发信息**: 用于更准确、更精炼地描述状态
- 2) **过程性启发信息**: 用于构造操作算子
- 3) **控制性启发信息**: 表示控制策略的知识

2	8	3
1	6	4
7		5

### ■ 按作用分类:

- 1) **用于扩展节点的选择**, 即用于决定应先扩展哪一个节点, 以免盲目扩展。
- 2) **用于生成节点的选择**, 即用于决定要生成哪些后继节点, 以免盲目生成过多无用的节点。
- 3) **用于删除节点的选择**, 即用于决定删除哪些无用节点, 以免造成进一步的时空浪费。



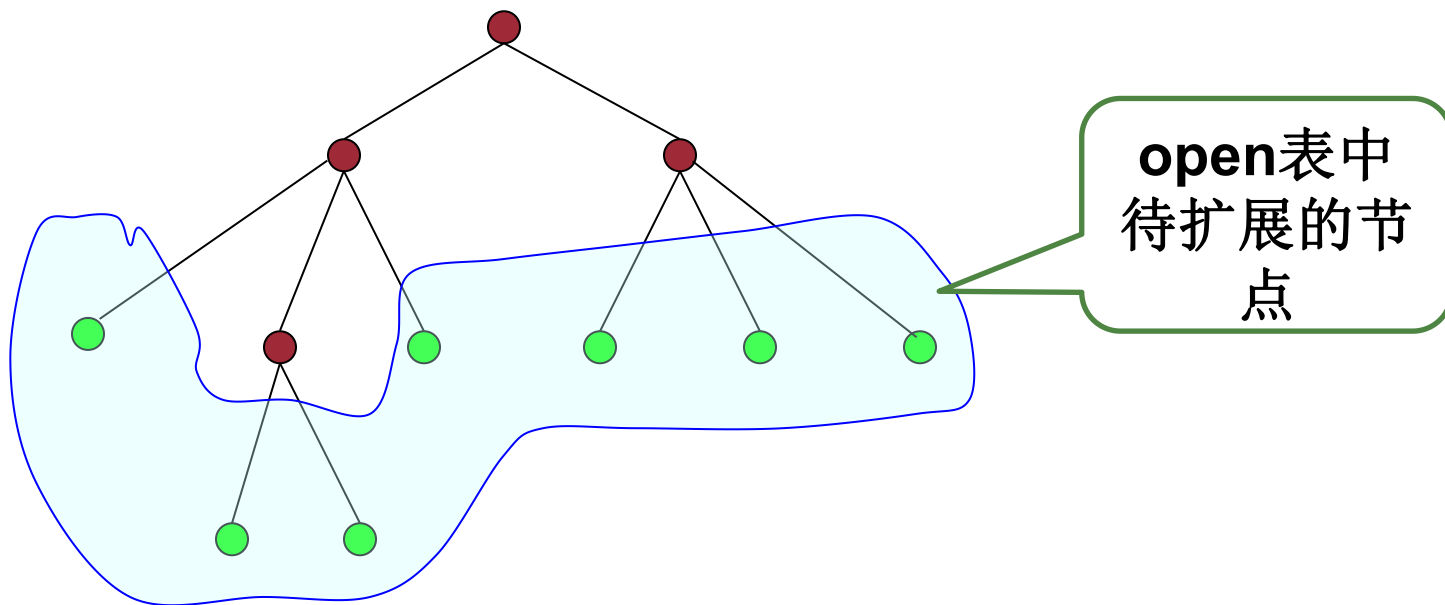
## 5.4.1 启发信息和估价函数

- 用于扩展节点的选择，即用于决定应先扩展哪一个节点，以免盲目扩展。

问题：状态空间图搜索中，待扩展的节点放在哪里？

A.open表    B.closed表

- 思想：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节点来扩展。





## 5.4.1 启发信息和估价函数

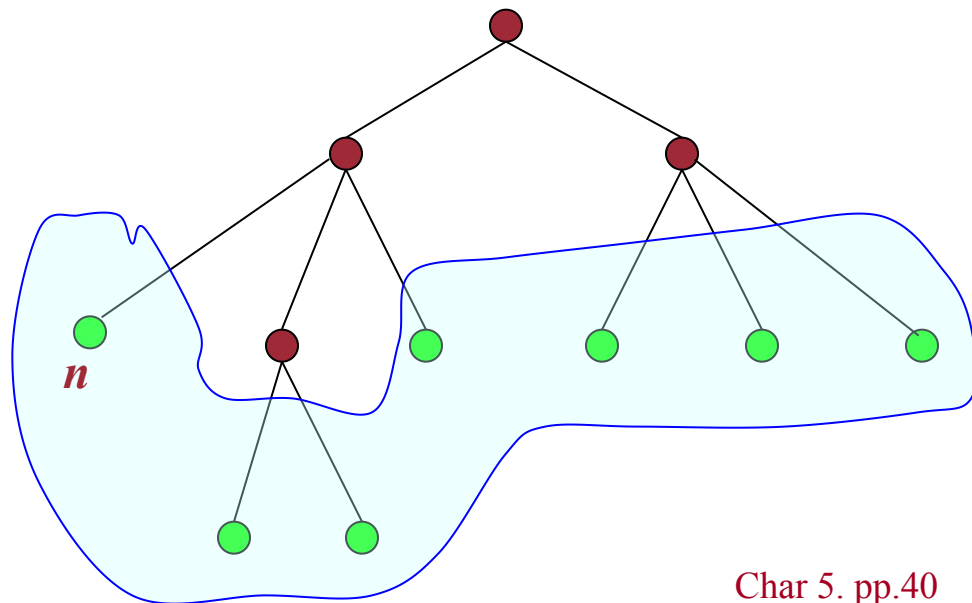
■ 估价函数（evaluation function）：估算节点“希望”程度的量度。

■ 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**。



## 5.4.1 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

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

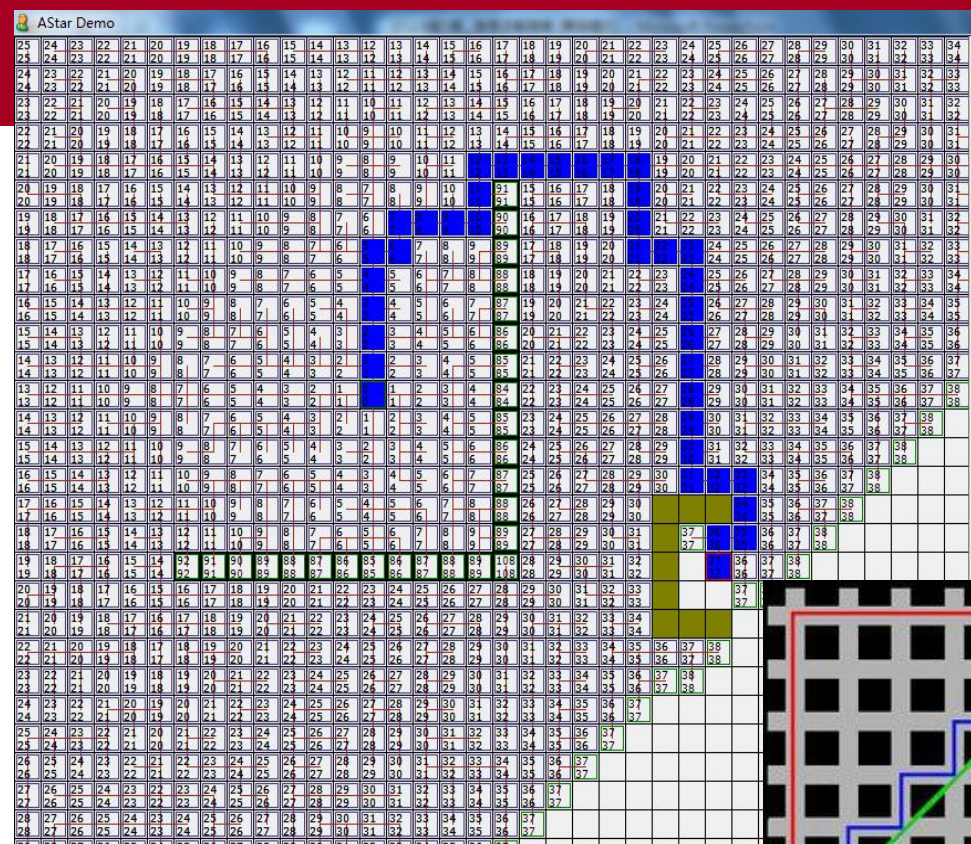
•  $g(n)$ ：从初始节点  $S_0$  到节点  $n$  的**实际代价**；

•  $h(n)$ ：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；

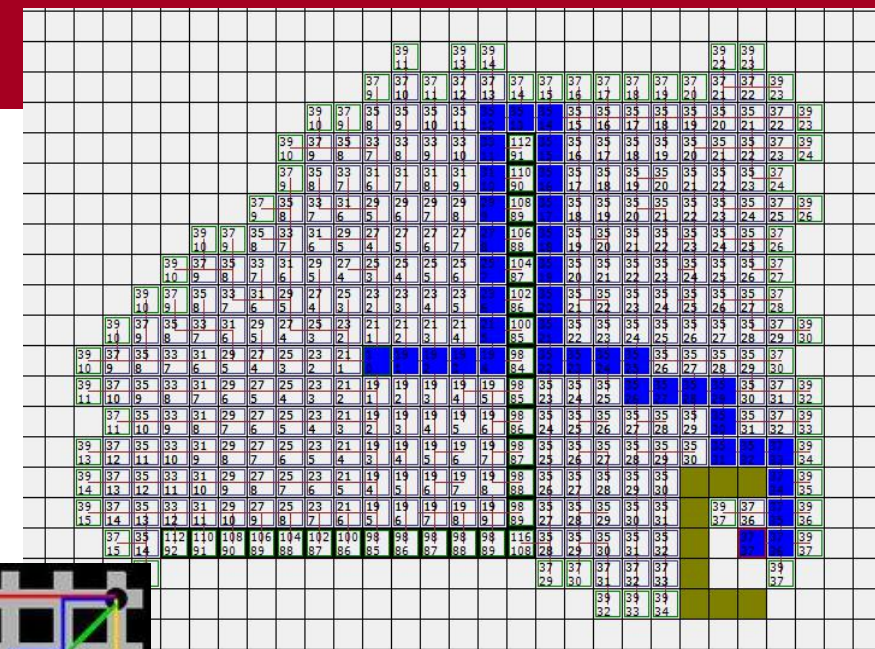
◆  $h(n)$  比重小：一般导致工作量加大，极限情况下变为盲目搜索，但可能可以找到最优解。



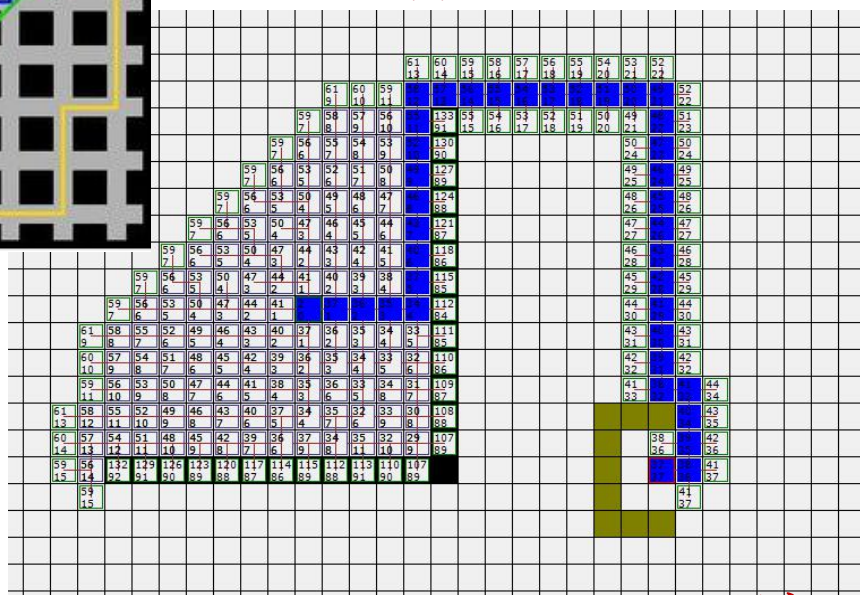


$$h(n) = 0$$

- $g(n)$ : 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**。



$$h(n) = \text{Manhattan距离}$$



$$2h(n) = 2 * \text{Manhattan距离}$$

## 5.4.1 启发信息和估价函数

- 建立启发函数的一般方法：
- 提出任意状态与目标状态之间的距离量度或差别量度；
- 试图确定一个处在最佳路径上的状态的概率；
- 在棋盘式的博弈和难题中根据棋局的某些特点来决定棋局的得分。

- $g(n)$ : 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。

## 5.4.1 启发信息和估价函数

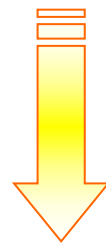
### ■ 例5.7 八数码问题的启发函数:

- 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  $h(S_0) = 8$ 。

好的启发函数的设计是一个**经验问题**, **判断和直觉**是很重要的因素, 但是衡量其好坏的最终标准是在具体应用时的搜索效果。

2	1	3
7	6	4
	8	5

初始棋局



1	2	3
8		4
7	6	5

目标棋局



## 5.4 启发式图搜索策略

- 5.4.1 启发信息和估价函数
- 5.4.2 A搜索算法
- 5.4.3 A\*搜索算法及其特性分析
- 补充：博弈搜索策略

一般的启发式搜索算法：

- 爬山法 (hill climbing)
- 最佳优先搜索 (best-first-search)
- A搜索算法

## 5.4.2 A搜索算法

1. 爬山（hill climbing）法（Pearl, 1984）：一种局部择优或最近择优的启发式方法。  $f(n) = h(n)$ ,  $g(n) = 0$

- 在搜索过程中先扩展当前状态，然后再评估它的孩子，即估计目标状态和它的孩子的“距离”或计算当前节点到它们孩子节点的“距离”。而后选择最佳的孩子做进一步的扩展，不保留它的兄弟姐妹和双亲。

贪婪局部搜索



$h(n)$  表示山顶与当前位置  $n$  之间的高度之差

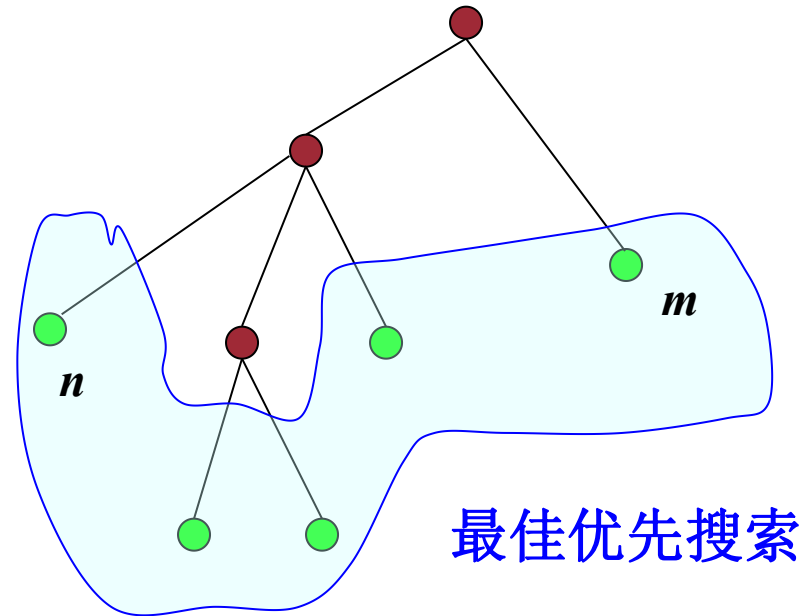
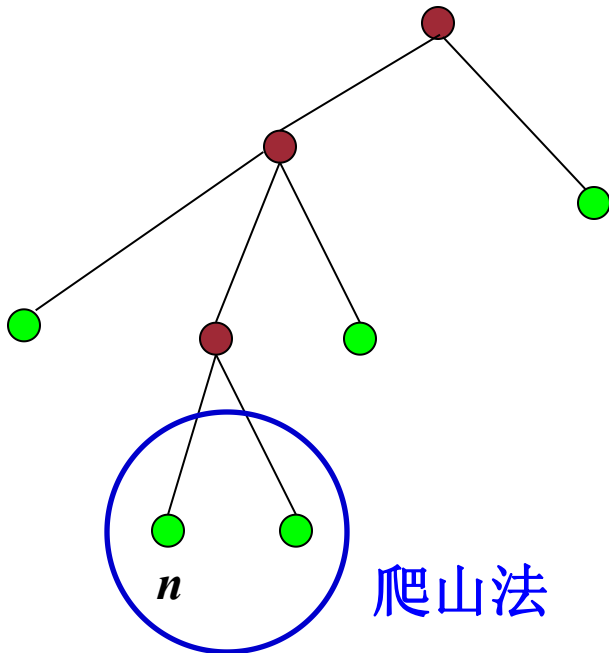


## 5.4.2 A搜索

- **OPEN表**: 保留所有已生成而未扩展的状态;
- **CLOSED表**: 记录已扩展过的状态。

2. 最佳优先搜索 (best-first-search): 一种全局择优的启发式方法。  $f(n) = h(n)$ ,  $g(n) = 0$

对OPEN表中的状态进行排序, 排序的依据是对状态与目标“接近程度”的某种启发式估计。每一轮循环考虑OPEN表中最有希望的状态, 并且生成其所有的儿女节点。



## 5.4.2 A搜索

- **OPEN表**: 保留所有已生成而未扩展的状态;
- **CLOSED表**: 记录已扩展过的状态。

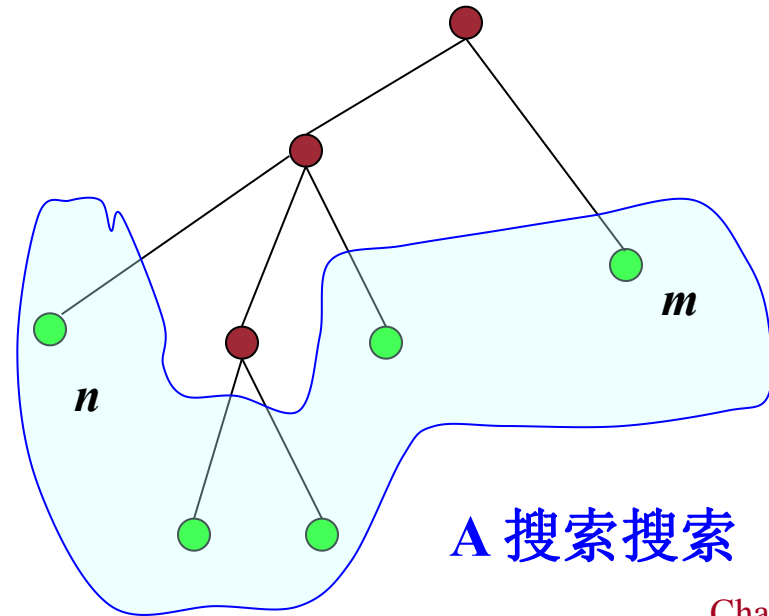
3. A 搜索算法: 使用了**估价函数** $f$ 的最佳优先搜索。

■ 估价函数  $f(n) = g(n) + h(n)$

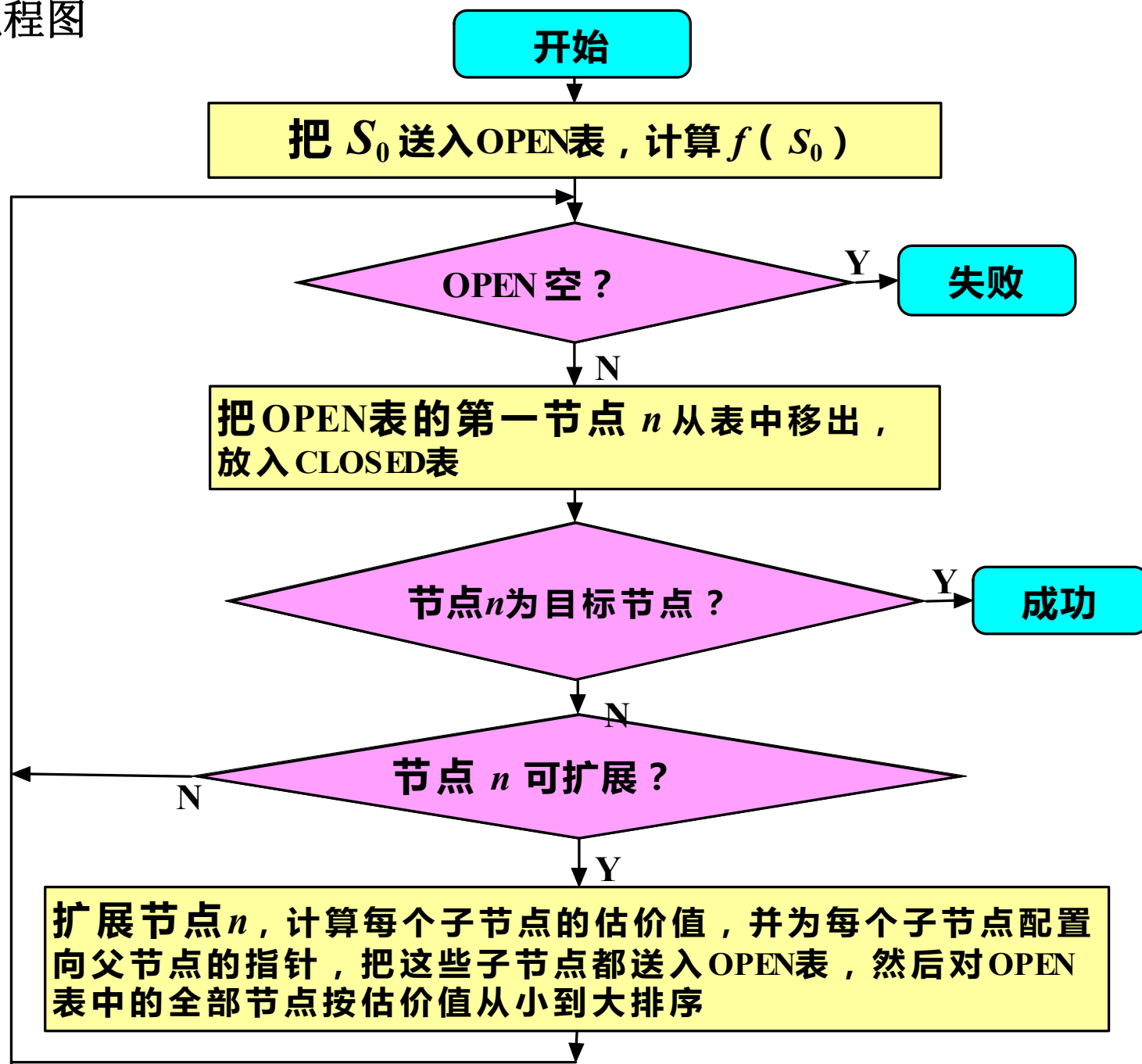
■ 如何寻找并设计启发函数  $h(n)$ ，然后以  $f(n)$  的大小来排列OPEN表中待扩展状态的次序，每次选择  $f(n)$  值最小者进行扩展。

$g(n)$ : 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。



# A 算法流程图

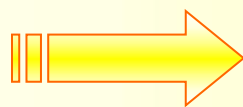


## 5.4.2 A搜索算法

- 例5.8 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  $h(S_0)=5$ 。

2	8	3
1	6	4
7		5

初始状态  $S_0$



1	2	3
8		4
7	6	5

目标状态

操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

1	2	3
8		4
7	6	5

A (1+3=4)

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7		5

S (0+5=5)

估价值  $f$  一样时,  
启发值  $h$  小的优先

B (1+6=7)

2	8	3
1	6	4
7	5	

C (1+6=7)

2	8	3
1	6	4
	7	5

D (2+4=6)

2		3
1	8	4
7	6	5

E (2+5=7)

2	8	3
1	4	
7	6	5

F (2+4=6)

2	8	3
	1	4
7	6	5

2	3	
1	8	4
7	6	5

G (3+5=8)

	2	3
1	8	4
7	6	5

H (3+3=6)

1	2	3
	8	4
7	6	5

J (5+3=8)

1	2	3
7	8	4
	6	5

K (5+0=5)

1	2	3
8		4
7	6	5

解:  $\uparrow, \uparrow, \leftarrow, \downarrow, \rightarrow$

● 本图操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

扩展节点数: 5

生成节点数: 11

● P125图5.16操作算子集:  $\leftarrow, \uparrow, \rightarrow, \downarrow$

扩展节点数: 6

生成节点数: 13

操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

1	2	3
8		4
7	6	5

A (1+3=4)

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7		5

S (0+5=5)

估价值  $f$  一样时,  
启发值  $h$  小的优先

B (1+6=7)

2	8	3
1	6	4
7	5	

C (1+6=7)

2	8	3
1	6	4
	7	5

D (2+4=6)

2		3
1	8	4
7	6	5

E (2+5=7)

2	8	3
1	4	
7	6	5

F (2+4=6)

2	8	3
	1	4
7	6	5

2	3	
1	8	4
7	6	5

G (3+5=8)

	2	3
1	8	4
7	6	5

H (3+3=6)

1	2	3
	8	4
7	6	5

I (4+2=6)

J (5+3=8)

1	2	3
7	8	4
	6	5

K (5+0=5)

1	2	3
8		4
7	6	5

解:  $\uparrow, \uparrow, \leftarrow, \downarrow, \rightarrow$

• 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

## 5.4 启发式图搜索策略

- 5.4.1 启发信息和估价函数
- 5.4.2 A搜索算法
- 5.4.3 A\*搜索算法及其特性分析
- 补充：博弈搜索策略



## 5.4.3 A\*搜索算法及其特性分析

■ A\* 算法 (最佳图搜索算法, Nilsson, 1968)

1) OPEN表中的节点按估价函数  $f(n) = g(n) + h(n)$  的值从小到大排序;

2)  $h(n)$  是  $h^*(n)$  的下界, 即  $h(n) \leq h^*(n)$ 。

■  $h^*(n)$ : 节点  $n$  到目标节点的最短路径的代价

■ 如果某一问题有解, 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解。

操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

1	2	3
8		4
7	6	5

A (1+3=4)

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7		5

S (0+5=5)

估价值  $f$  一样时,  
启发值  $h$  小的优先

B (1+6=7)

2	8	3
1	6	4
7	5	

C (1+6=7)

2	8	3
1	6	4
	7	5

D (2+4=6)

2		3
1	8	4
7	6	5

E (2+5=7)

2	8	3
1	4	
7	6	5

F (2+4=6)

2	8	3
	1	4
7	6	5

2	3	
1	8	4
7	6	5

G (3+5=8)

	2	3
1	8	4
7	6	5

H (3+3=6)

1	2	3
	8	4
7	6	5

J (5+3=8)

1	2	3
7	8	4
	6	5

K (5+0=5)

1	2	3
8		4
7	6	5

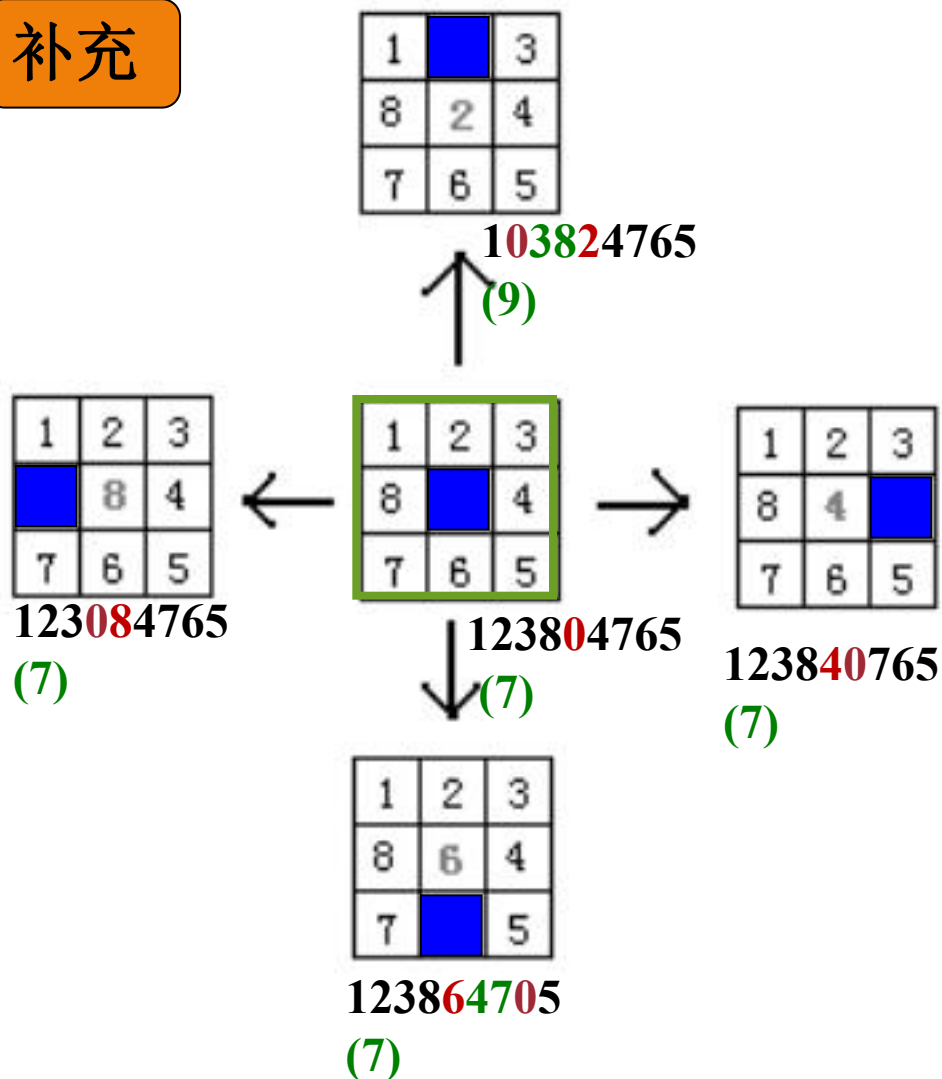
解:  $\uparrow, \uparrow, \leftarrow, \downarrow, \rightarrow$

■  $g(n)$ : 节点  $n$  的深度

■  $h(n)$ : 节点  $n$  与目标棋局不相同的位数 (包括空格),

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

# 补充



- 如果初始状态的逆序对数奇偶性与目标状态的不同，则八数码问题无解，反之有解。

- 目标状态的逆序对数：  
 $0+0+0+0+1+1+2+3=7$
- 空格左移、右移：不改变序列逆序对数的奇偶性；
- 空格上移、下移：假定空格与格子x交换，将会有2个格子（a和b）改变与x的前后关系。若：
  - $x < a$  且  $x < b$ ，逆序对数改变2次
  - $x > a$  且  $x > b$ ，逆序对数改变2次
  - $a < x < b$  或  $b < x < a$ ，逆序对数改变0次
- 空格移动不会改变序列的逆序对数奇偶性。

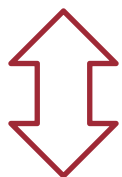
## 5.4.3 A\*搜索算法及其特性分析

### 1. 可采纳性

■ 当一个搜索算法在最短路径存在时能保证找到它，就称该算法是可采纳的。

■ A\*搜索算法是可采纳的。

■ A\*搜索算法(  $h(n)=0$  )



宽度优先搜索算法

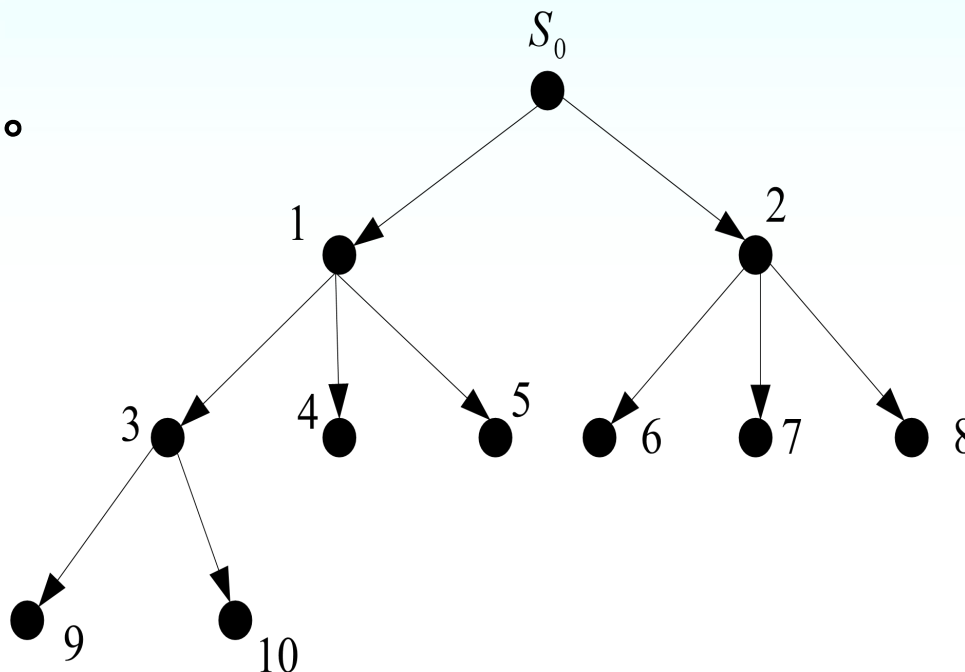


图5.6 宽度优先搜索法中状态的搜索次序

## 5.4.3 A\*搜索算法及其特性分析

### ■ 2. 单调性

■ 如果对启发函数 $h(n)$ 加上单调性的限制，就可以总从祖先状态沿着最佳路径到达任一状态。

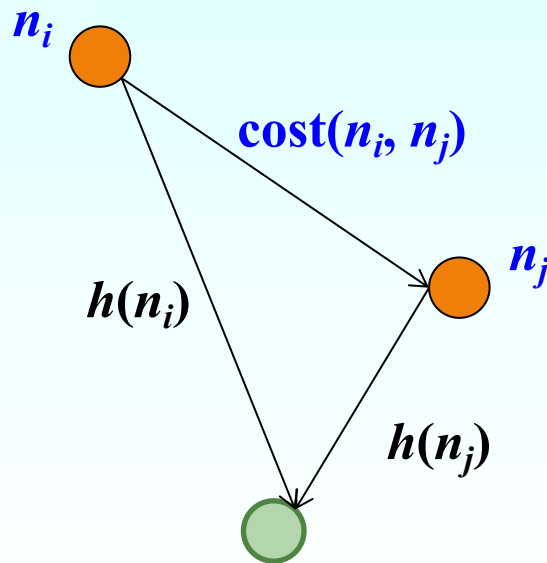
■ 如果某一启发函数 $h(n)$ 满足：

1) 对所有状态 $n_i$ 和 $n_j$ ，其中 $n_j$ 是 $n_i$ 的后裔，满足 $h(n_i) - h(n_j) \leq \text{cost}(n_i, n_j)$ ，其中 $\text{cost}(n_i, n_j)$ 是从 $n_i$ 到 $n_j$ 的实际代价。

2) 目的状态的启发函数值为0。

■ 则称 $h(n)$ 是单调的。

■ A\*搜索算法中采用单调性启发函数，可以减少比较代价和调整路径的工作量，从而减少搜索代价。



操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

1	2	3
8		4
7	6	5

A (1+3=4)

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7		5

S (0+4=4)

估价值  $f$  一样时,  
启发值  $h$  小的优先

B (1+5=6)

2	8	3
1	6	4
7	5	

C (1+5=6)

2	8	3
1	6	4
	7	5

D (2+3=5)

2		3
1	8	4
7	6	5

F (2+3=5)

2	8	3
	1	4
7	6	5

E (2+4=6)

2	8	3
1	4	
7	6	5

✓ 采用 “不在位数”  
(不包括空格) 作为  $h(n)$

✓ 因为  $h(n_i) - h(n_j) \leq 1$ ,  
所以该  $h$  是单调的。

2	3	
1	8	4
7	6	5

G (3+4=7)

	2	3
1	8	4
7	6	5

H (3+2=5)

1	2	3
	8	4
7	6	5

I (4+1=5)

J (5+2=7)

1	2	3
7	8	4
	6	5


K (5+0=5)

1	2	3
8		4
7	6	5

$n_i$  不在位  $\rightarrow n_j$  在位:  $h(n_i) - h(n_j) = 1$   
 $n_i$  不在位  $\rightarrow n_j$  不在位:  $h(n_i) - h(n_j) = 0$   
 $n_i$  在位  $\rightarrow n_j$  在位:  $h(n_i) - h(n_j) = 0$   
 $n_i$  在位  $\rightarrow n_j$  不在位:  $h(n_i) - h(n_j) = -1$   
 所以  $h(n_i) - h(n_j) \leq 1$

## 5.4.3 A\*搜索算法及其特性分析

### ■ 3. 信息性

- 在两个A\*启发策略的 $h_1$ 和 $h_2$ 中，如果对搜索空间中的任意状态 $n$ 都有 $h_1(n) \leq h_2(n)$ ，就称策略 $h_2$ 比 $h_1$ 具有更多的信息性。
- 如果某一搜索策略的 $h(n)$ 越大，则A\*算法搜索的信息性越多，所搜索的状态越少。
- 但更多的信息性需要更多的计算时间，可能抵消减少搜索空间所带来的益处。



操作算子集:  $\uparrow, \downarrow, \rightarrow, \leftarrow$

估价值  $f$  一样时,  
启发值  $h$  小的优先

1	2	3
8		4
7	6	5

A (1+3=4)

2	8	3
1	6	4
7		5

S (0+4=4)

2	8	3
1		4
7	6	5

B (1+5=6)

2	8	3
1	6	4
	7	5

C (1+5=6)

D (2+3=5)

2		3
1	8	4
7	6	5

2	8	3
1	4	
7	6	5

F (2+3=5)

2	8	3
	1	4
7	6	5

E (2+4=6)

2	3	
1	8	4
7	6	5

G (3+4=7)

	2	3
1	8	4
7	6	5

H (3+2=5)

1	2	3
	8	4
7	6	5

I (4+1=5)

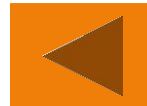
J (5+2=7)

1	2	3
7	8	4
	6	5

K (5+0=5)


1	2	3
8		4
7	6	5

- $h_1(n)$ : “不在位数” (不包括空格),  $h_1(A)=3$
- $h_2(n)$ : 和目的状态之间的距离,  $h_2(A)=4$
- $h_2(n) \geq h_1(n)$



## 5.4.3 A\*搜索算法及其特性分析

- $h_1(n)$ : “不在位数”（不包括空格）；
- $h_2(n)$ : 和目的状态之间的距离；
- $h_2(n) \geq h_1(n)$



2	8	3
1	6	4
7		5

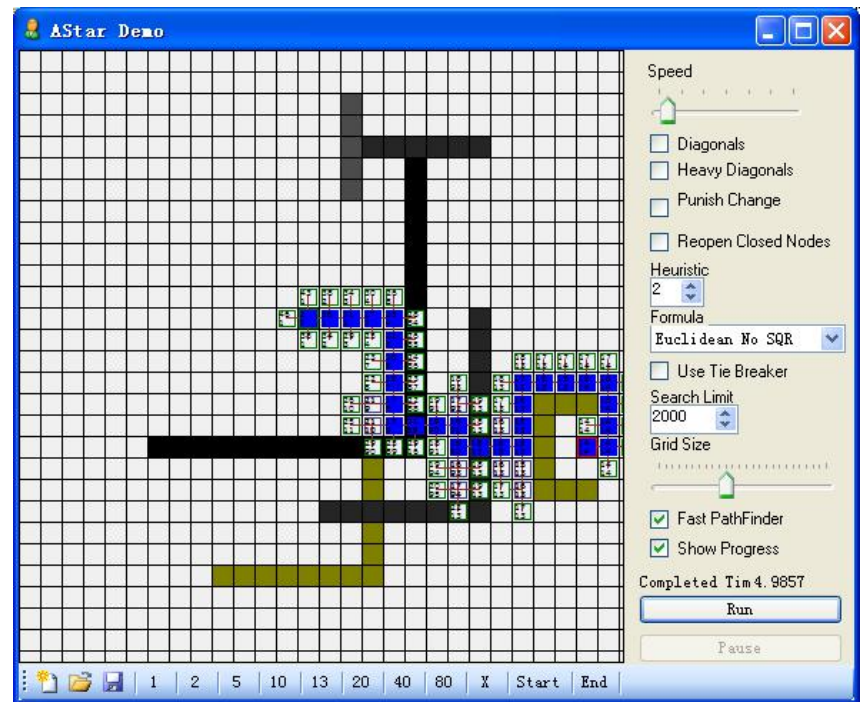
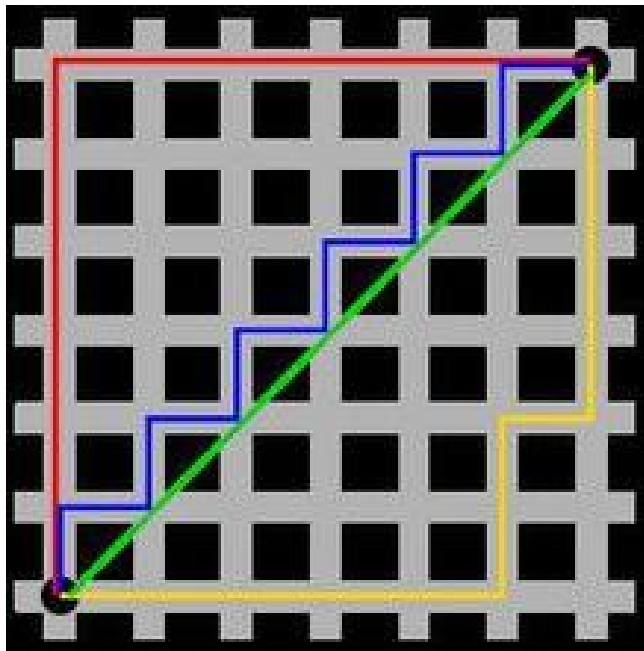
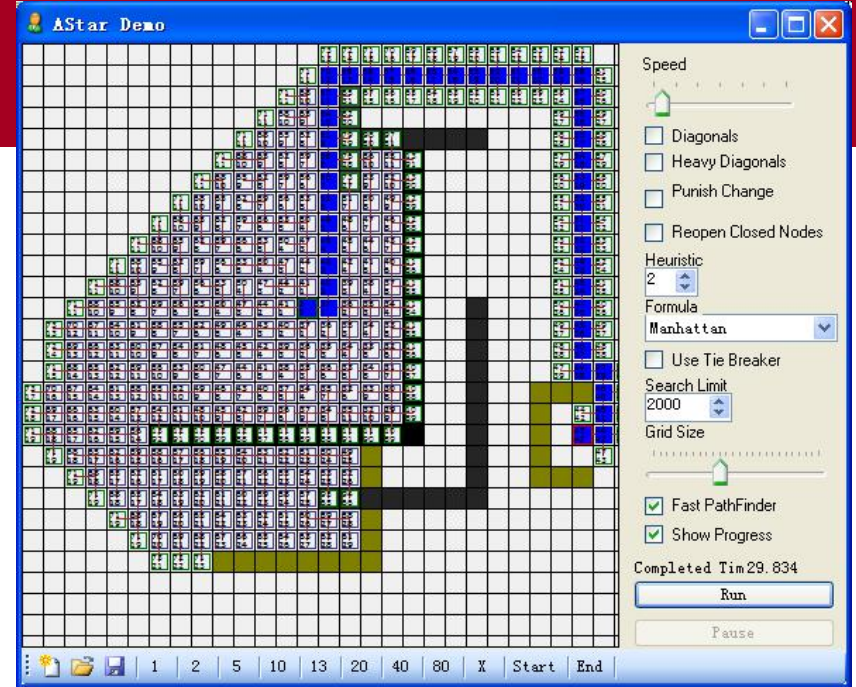
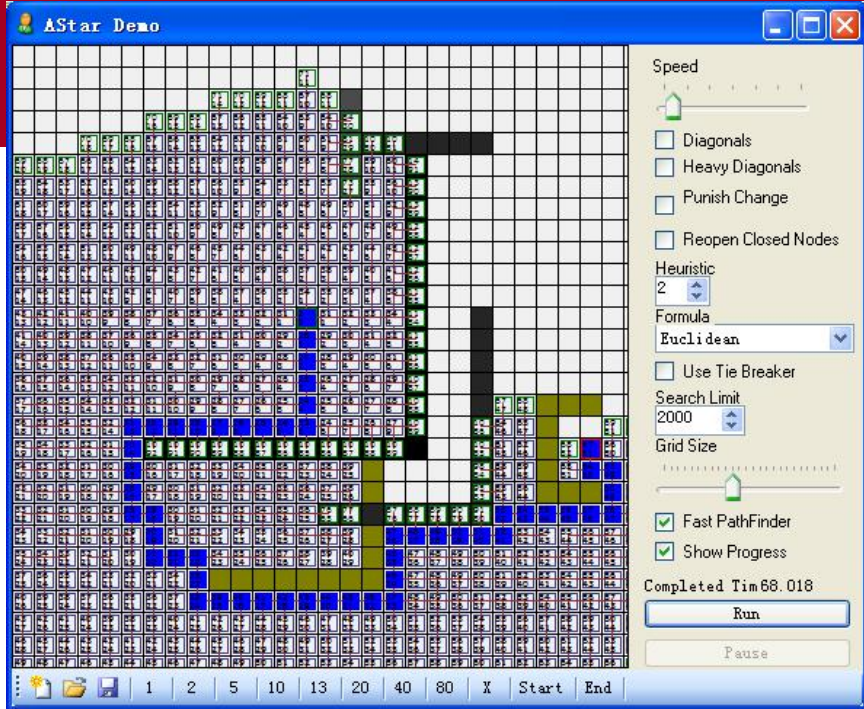
初始状态  $S_0$

1	2	3
8		4
7	6	5

目标状态

■ 对该八数码问题取不同启发函数，应用A\*算法求得最佳解时所扩展生成的节点数。（操作符顺序：←, ↑, →, ↓）

启发函数	$h(n)=0$	$h(n)=h_1(n)$	$h(n)=h_2(n)$
扩展节点数	37	6	5
生成节点数	66	13	11



# A\*搜索算法例子

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

评估函数      当前最小开销代价      后续最小开销代价

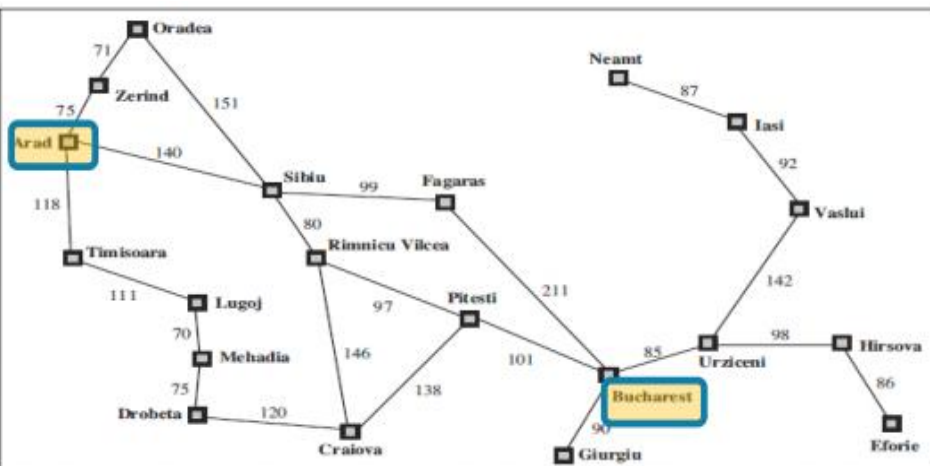


Figure 3.2 A simplified road map of part of Romania.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.22 Values of  $h_{SLD}$ —straight-line distances to Bucharest.

辅助信息：任意一个城市与Bucharest之间的直线距离

(a) 初始状态

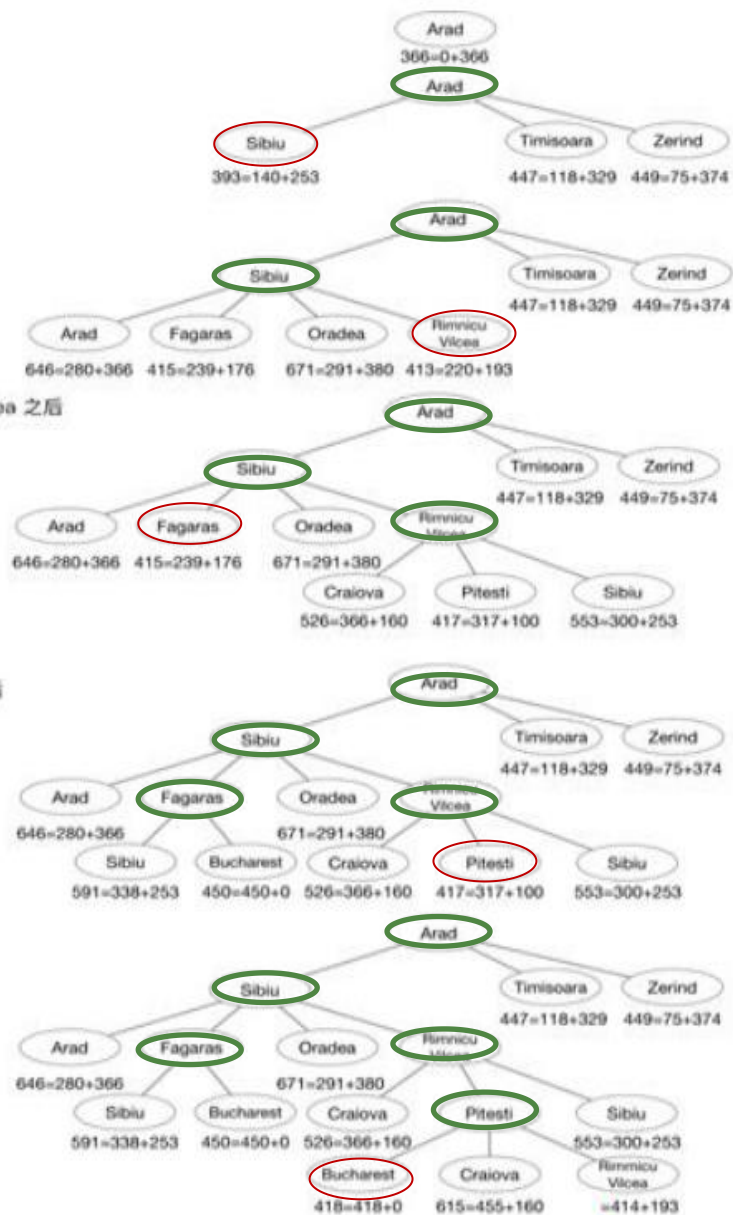
(b) 扩展 Arad 之后

(c) 扩展 Sibiu 之后

(d) 扩展 Rimnicu Vilcea 之后

(e) 扩展 Fagaras 之后

(f) 扩展 Pitesti 之后





## 5.4 启发式图搜索策略

- 5.4.1 启发信息和估价函数
- 5.4.2 A搜索算法
- 5.4.3 A\*搜索算法及其特性分析
- 补充：博弈搜索策略

✓ 对A\*的改进：

可采纳性不变；不多扩展节点；不增加算法的复杂性