



# 人工智能概述

主讲人：高 灿

计算机视觉所，致腾楼936

[davidgao@szu.edu.cn](mailto:davidgao@szu.edu.cn)



深圳大学 计算机与软件学院

College of Computer Science and Software Engineering of Shenzhen University

# 内容回顾

**知识**是人们在长期的生活及社会实践中、在科学研究及实验中积累起来的对**客观世界的认识与经验**。

## 知识的形式

- 事实
- 因果关联

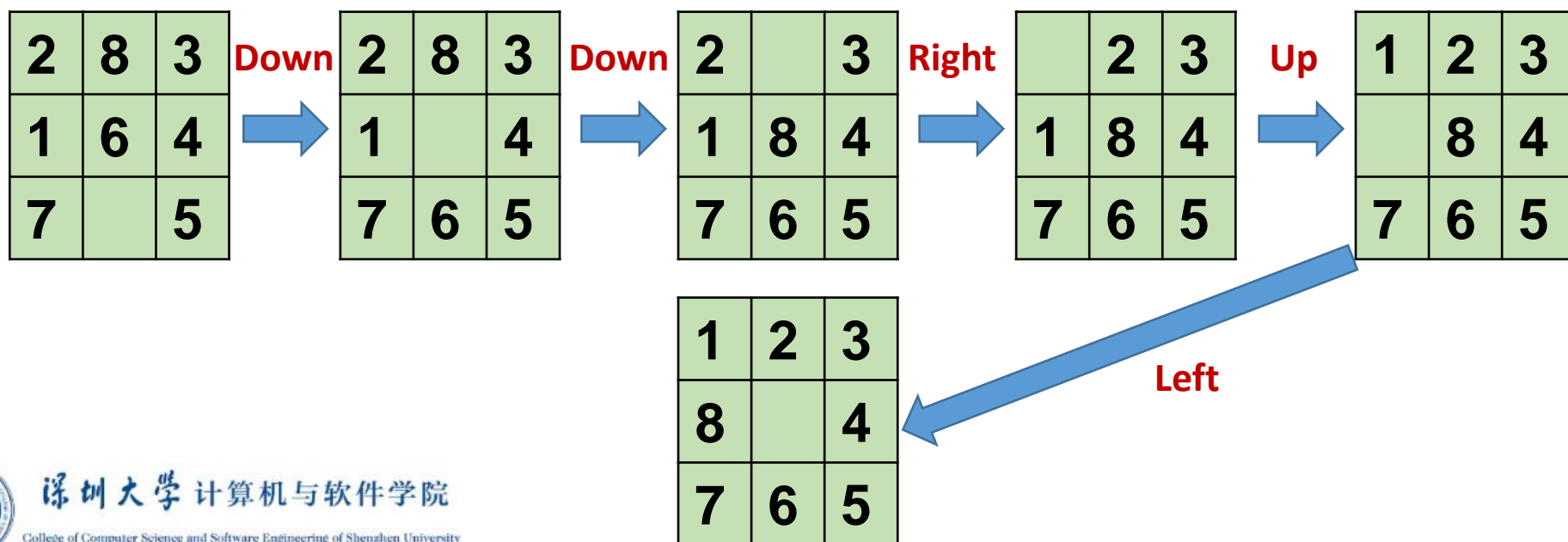
## 知识的特性

- 相对正确性（明确前题）
- 不确定性（真假之外）
- 可表示性（形式描述）
- 可利用性（学以致用）

# 知识表示

**状态空间方法**：基于解答空间的问题表示和求解方法，它以状态和算符为基础来表示和求解问题。

- (1) **状态**(state)：表示问题解法中每一步问题状况的数据结构；
- (2) **操作**(operator)：将问题从一种状态变换为另一种状态的手段；
- (3) **状态空间**(state space)是一个表示该问题全部可能状态及其关系的图



# 知识表示

## 状态空间法定义

状态空间方法可形式化为四元组表示：(S, O, S<sub>0</sub>, G)

其中，S 是状态的集合，O 是操作算子的集合，S<sub>0</sub> 是初始状态，G 是目标状态。

2	8	3
1	6	4
7		5

初始状态

2		3
1	8	4
7	6	5

中间状态

1	2	3
8		4
7	6	5

目标状态

Left  
Right  
Up  
Down

操作算子

**状态空间的解：**从初始状态 S<sub>0</sub> 到目标状态 G 的**操作算子序列**

**Down → Down → Right → Up → Left**

# 目录

1

搜索问题求解

2

无信息搜索\*\*

3

有信息搜索\*\*\*

4

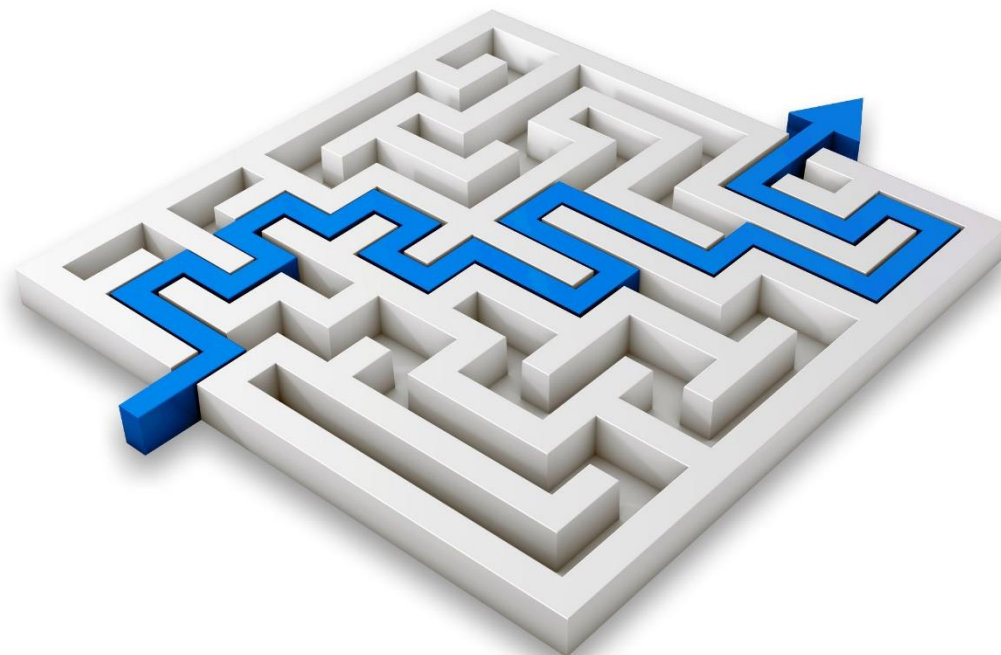
A/A\*搜索\*\*\*\*

5

基本搜索总结

6

习题及实验



# 1. 搜索问题求解

## 问题求解

AI中每个研究领域都有其各自的特点和规律,但都可抽象为一个问题求解过程。

某些问题的求解过程实际上是一个**搜索**。

AI为什么要研究搜索:

问题没有直接的解法

需要探索地求解

根据问题的实际情况不断寻找可利用的知识,构造出一条代价较少的推理路线,使问题得到圆满解决的过程称为搜索。

# 1. 搜索问题求解

## 问题求解

AI中每个研究领域都有其各自的特点和规律,但都可抽象为一个问题求解过程。

某些问题的求解过程实际上是一个**搜索**。

搜索问题求解包括两个重要的方面：**表示和搜索**

表示是基础，搜索必须要在表示的基础上进行，表示也关系到搜索的效率。

知识表示：**状态空间**、问题归约、谓词逻辑、产生式  
本章主要讨论**基于搜索的问题求解**（状态空间）



# 1. 搜索问题求解

## 搜索策略评价准则:

**完备性:** 如果存在一个解答，该策略是否保证能够找到？

**最优性:** 如果存在不同的几个解答，该策略可以发现是否最高质量的解答？

**时间复杂性:** 需要多长时间可以找到解答？

**空间复杂性:** 执行搜索需要多大存储空间？





# 1. 搜索问题求解

## 状态空间图

**图定义：**由非空有限集合 $V$ 和有限集 $E$ 组成的二元组 $G = \langle V, E \rangle$ ，其中：

(1)  $V = \{v_1, v_2, \dots, v_n\}$ 是有限非空集，称为结点集， $V$ 中的每个元素 $v_i$ 称为图 $G$ 的一个**结点**，简称为点。

(2)  $E$ 是由 $V$ 中的结点对为元素组成的有限集（可重集），称为边集， $E$ 中的每个元素称为图 $G$ 的一条**边**。

图中结点对既可无序，也可有序。

给定图 $G = \langle V, E \rangle$ ，若 $E$ 中的每条边都是**无序**的，则称该图为**无向图**，否则称为**有向图**。

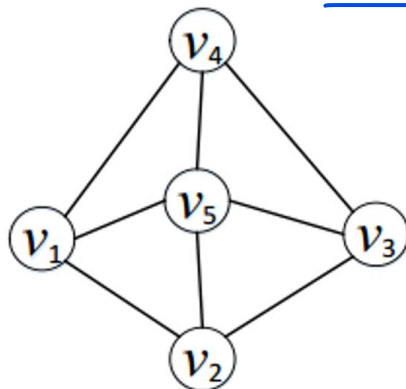
$$e = (u, v) = (v, u)$$



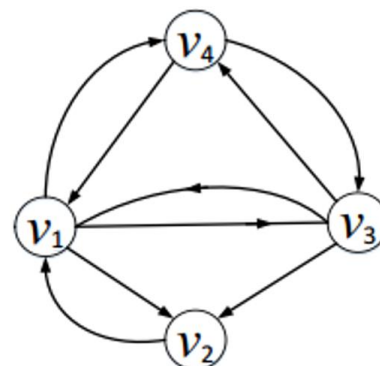
# 1. 搜索问题求解

## 状态空间图

例：将下图 所示的 $G_1$ 和 $G_2$ 表示成 $G = \langle V, E \rangle$ 的形式。



(a) 图  $G_1$



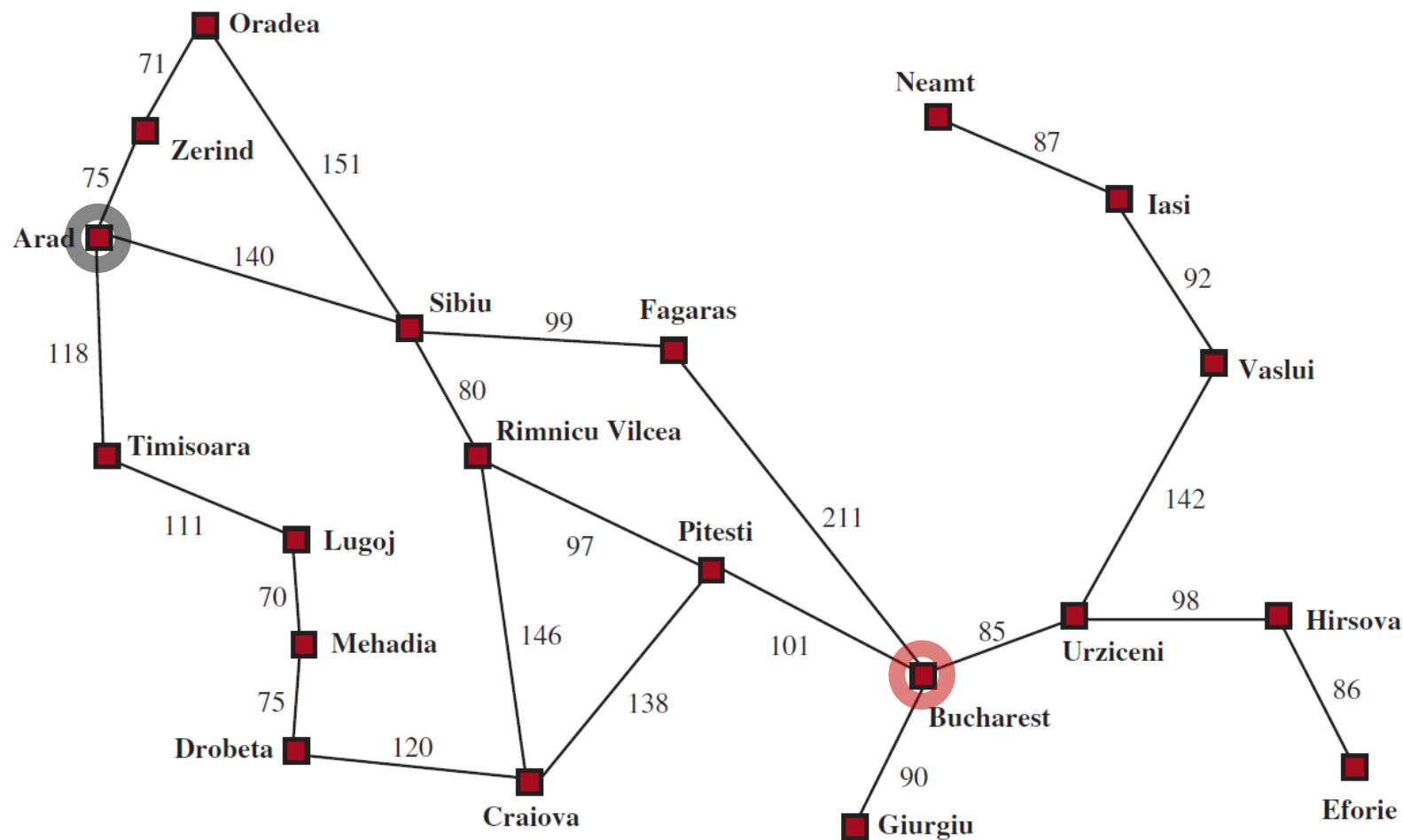
(b) 图  $G_2$

$$V_1 = \{v_1, v_2, v_3, v_4, v_5\}, E_1 = \{(v_1, v_2), (v_1, v_4), (v_1, v_5), (v_2, v_3), (v_2, v_5), \\ (v_3, v_4), (v_3, v_5), (v_4, v_5)\}$$

$$V_2 = \{v_1, v_2, v_3, v_4\}, E_2 = \{\langle v_1, v_2 \rangle, \langle v_1, v_3 \rangle, \langle v_1, v_4 \rangle, \langle v_2, v_1 \rangle, \langle v_3, v_1 \rangle, \\ \langle v_3, v_2 \rangle, \langle v_3, v_4 \rangle, \langle v_4, v_1 \rangle, \langle v_4, v_3 \rangle\}$$

# 1. 搜索问题求解

## 状态空间图



罗马尼亚旅行问题

# 1. 搜索问题求解

## 状态空间图

状态空间图是搜索问题的一种数学描述

**结点：**问题的某个状态

**边：**表示操作符（动作）

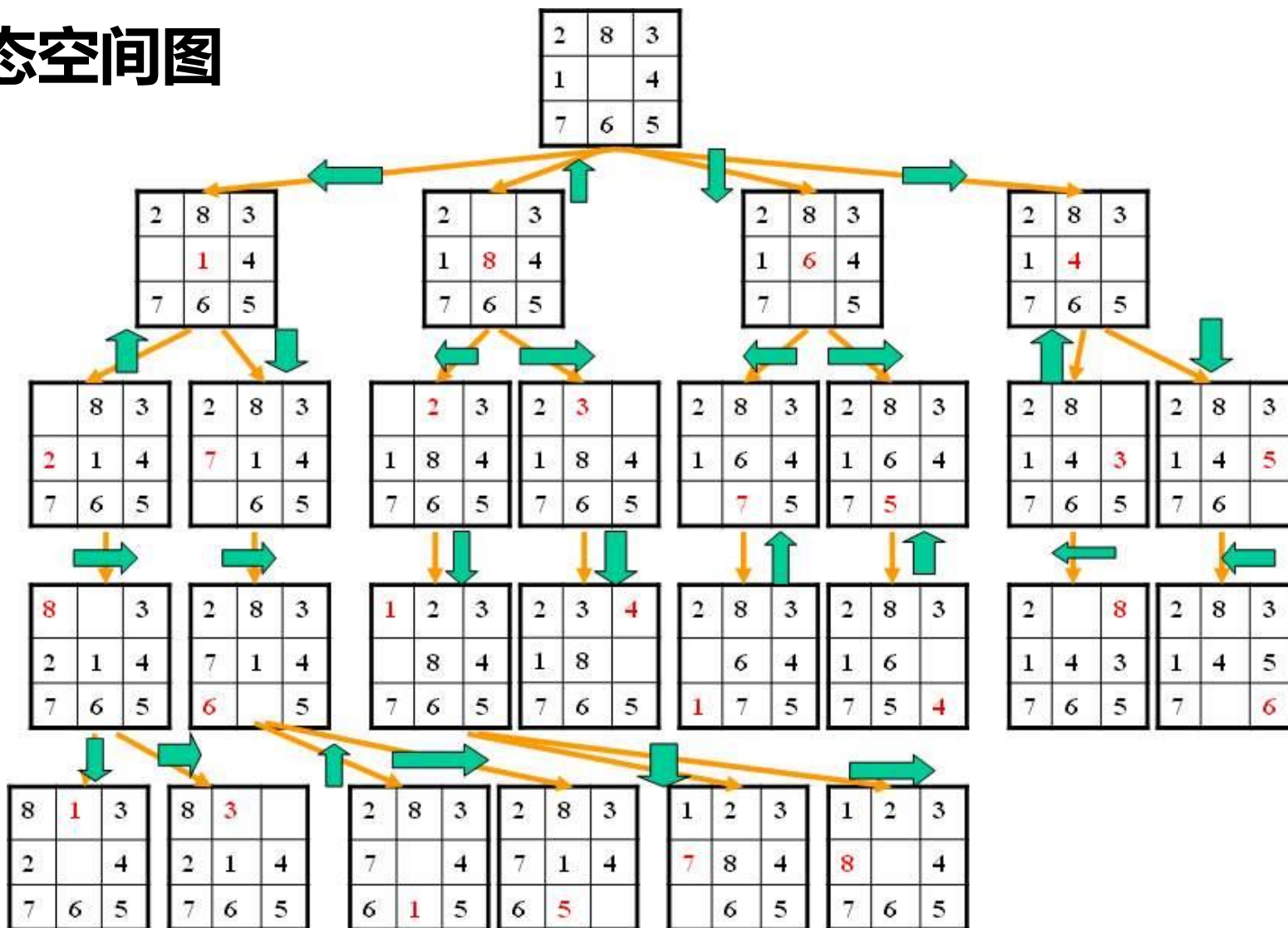
**目标：**一组结点，通常只有一个

在状态空间图中，每个状态仅出现一次

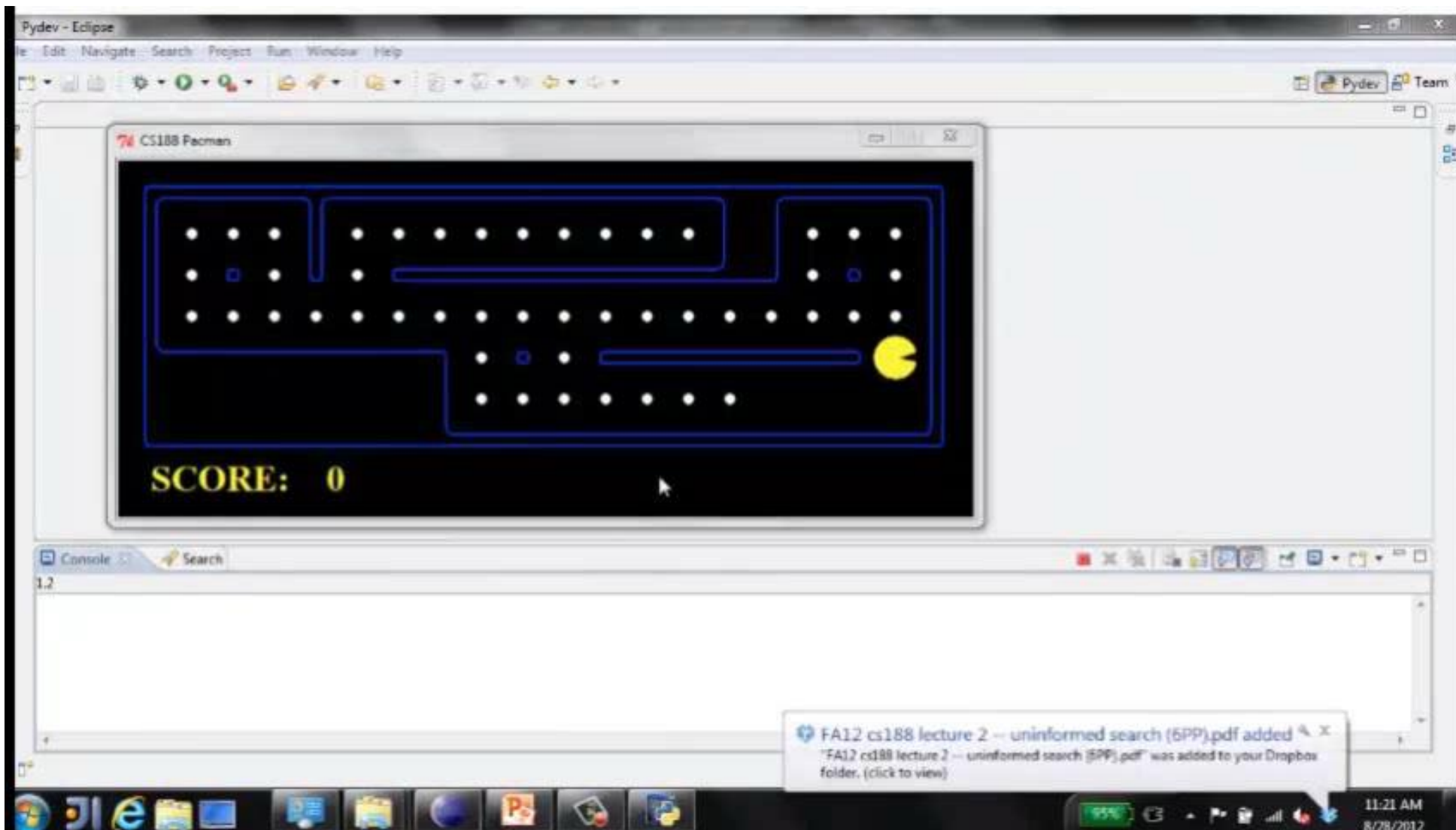
状态空间图一般较大，很少生成全图，但它是一种搜索问题的有效表示方法

# 1. 搜索问题求解

## 状态空间图

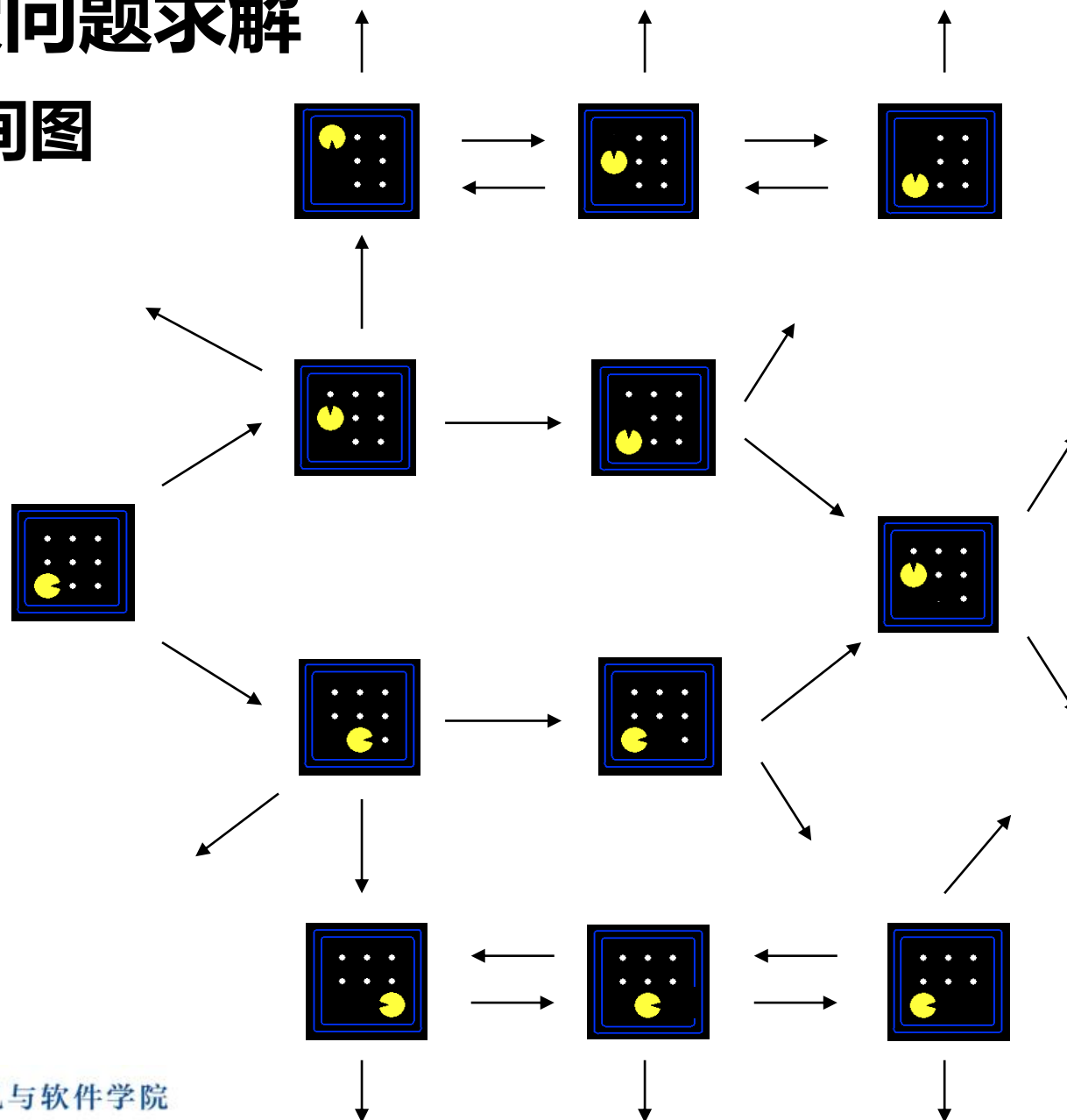


# 1. 搜索问题求解



# 1. 搜索问题求解

## 状态空间图

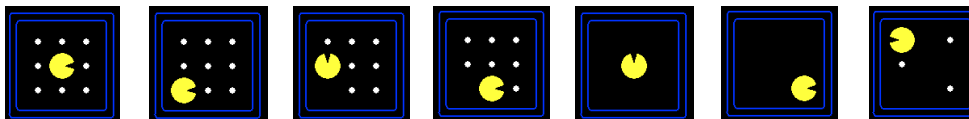


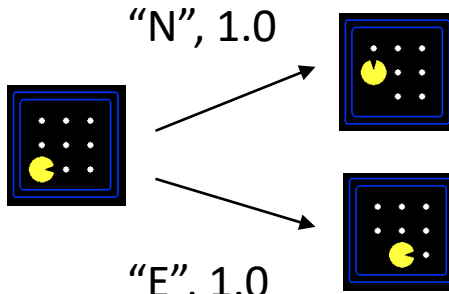


# 1. 搜索问题求解

## 状态空间图搜索

搜索问题由以下构成：

状态空间：

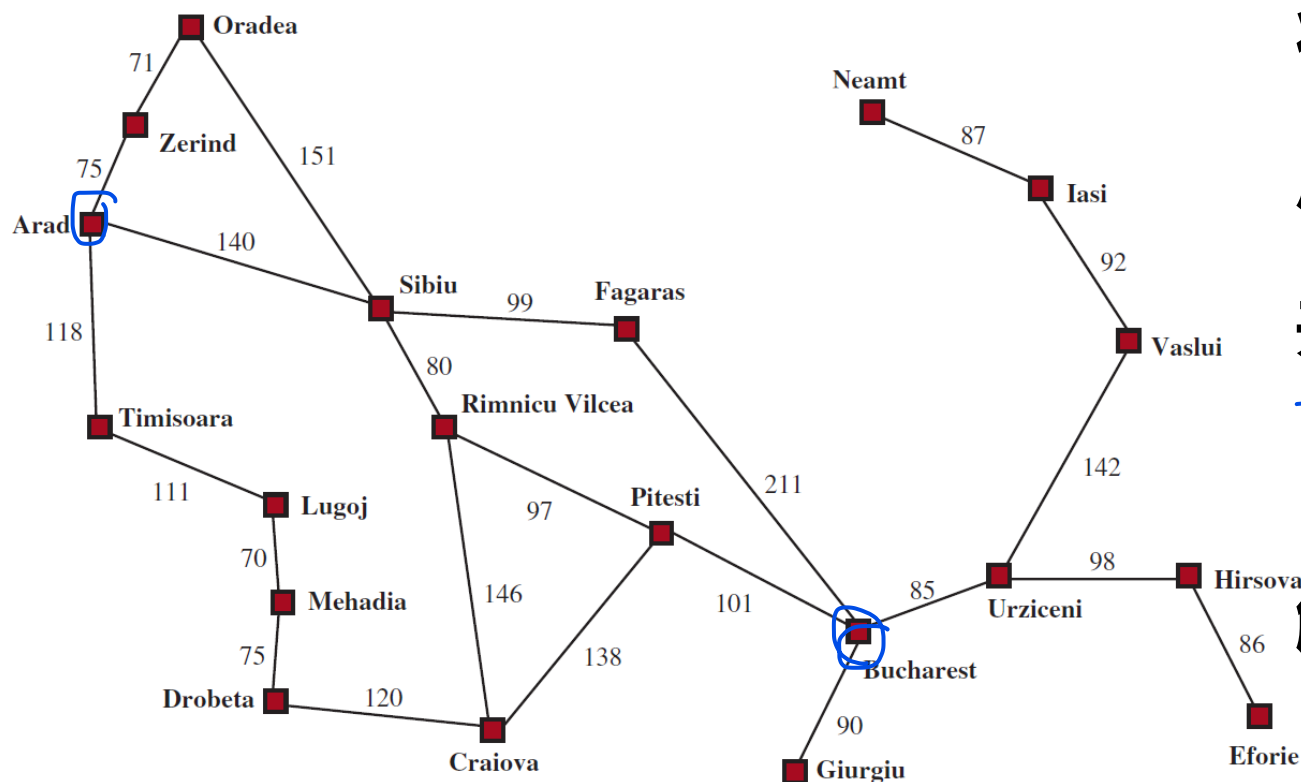
后继函数(动作或代价)：

开始和结束状态：

问题的解：将开始状态转换为目标状态的一系列动作

# 1. 搜索问题求解

## 状态空间图搜索



罗马尼亚旅行问题

状态空间: ?

后继函数: ?

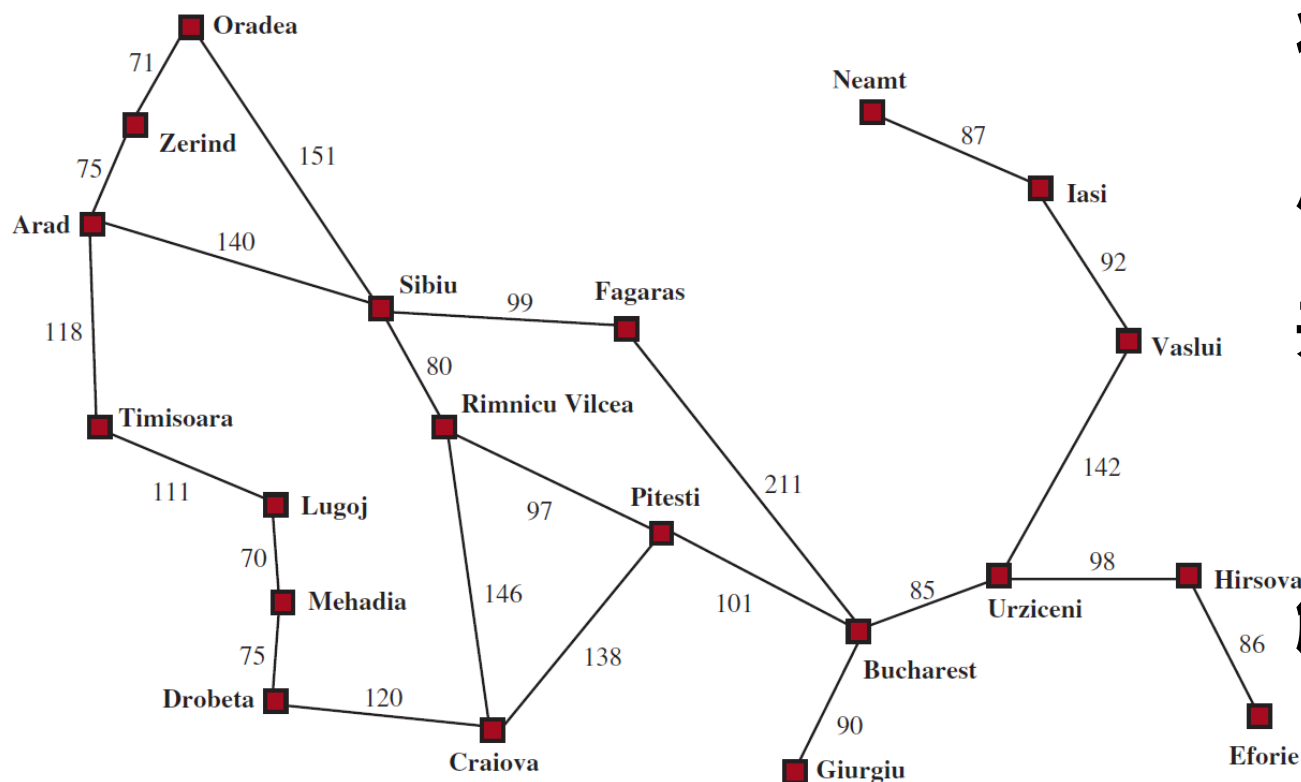
开始状态: Arad

目标测试: Bucharest?

解: ? (城市数/代价)

# 1. 搜索问题求解

## 状态空间图搜索



状态空间: 城市

后续函数: 道路

开始状态: Arad

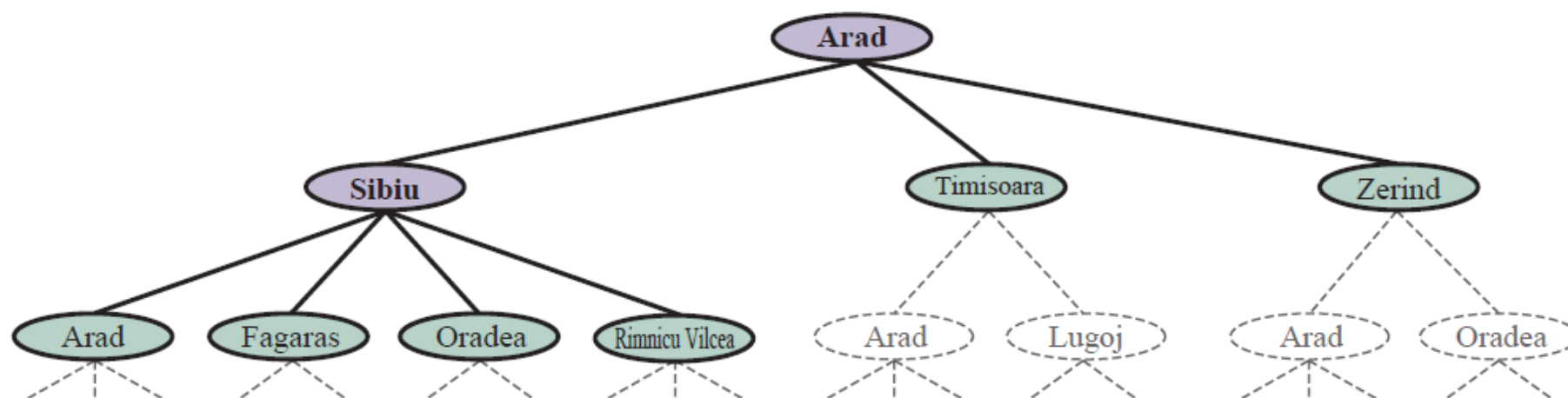
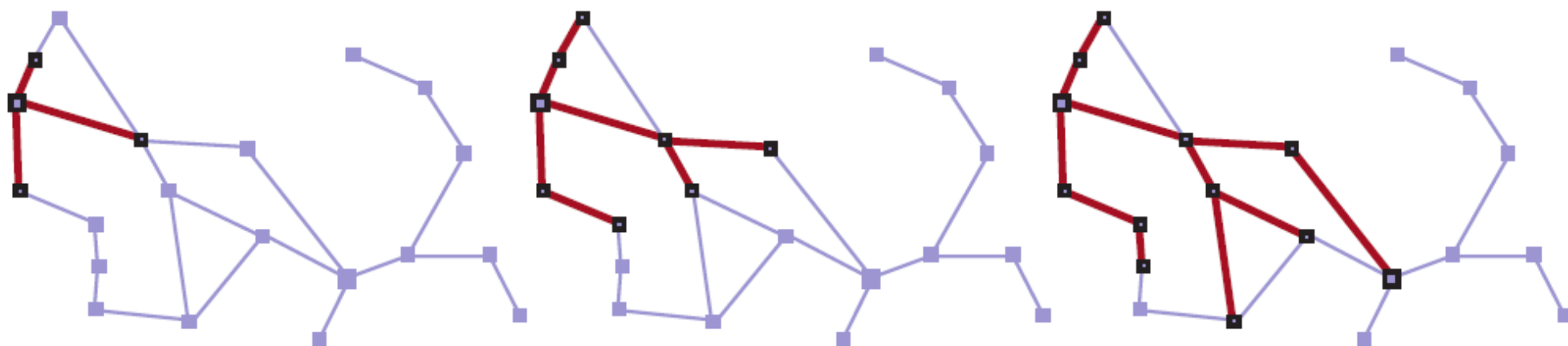
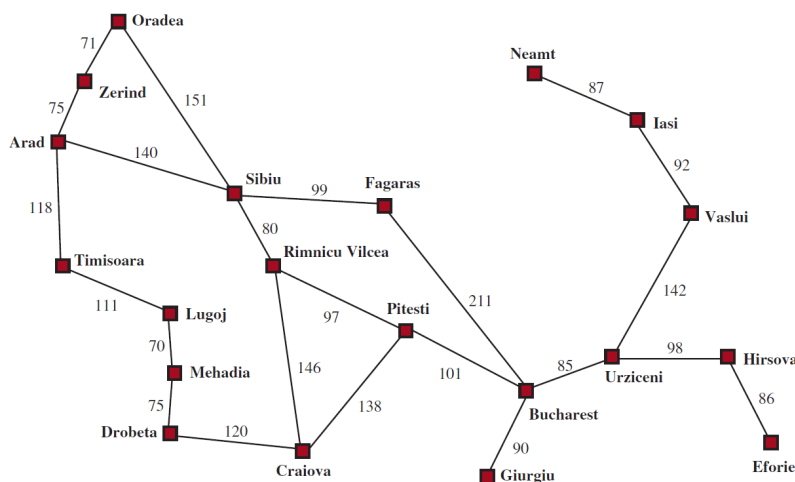
目标测试: Bucharest?

解: Ar-Si-Ri-Pi-Bu

罗马尼亚旅行问题

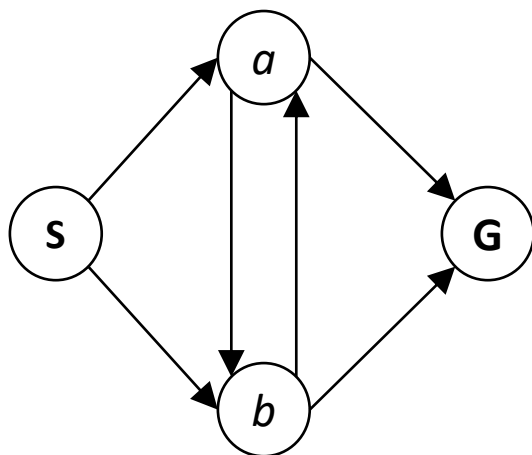
# 1. 搜索问题求解

## 状态空间图-搜索树



# 1. 搜索问题求解

## 状态空间图-搜索树



4个状态的状态空间图



且不

搜索树多大?

## 2. 无信息搜索

### 状态空间图搜索

US

图搜索策略主要分为**无信息搜索**(Uninformed Search)和**启发式搜索**(Heuristic Search)。  
HS

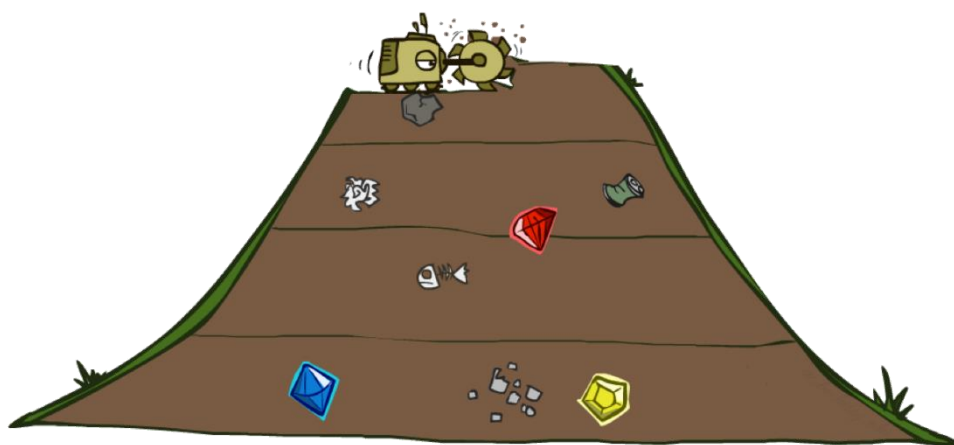
**无信息搜索**：也称为盲目搜索，即只按预定的控制策略进行搜索，在搜索过程中获得的中间信息**不会**用来改进控制策略。

**启发式搜索**：在搜索中加入了与问题有关的**启发性信息**，用于**指导**搜索朝着最有希望的方向进行，加速问题的求解过程并找到**最优解**。



## 2. 无信息搜索

无信息搜索一般是按**预定**的搜索策略进行搜索。由于这种搜索总是按预定的路线进行，没有考虑到问题本身的特性，所以这种搜索具有很大的盲目性，效率不高，适应于简单问题求解。



宽度优先



深度优先

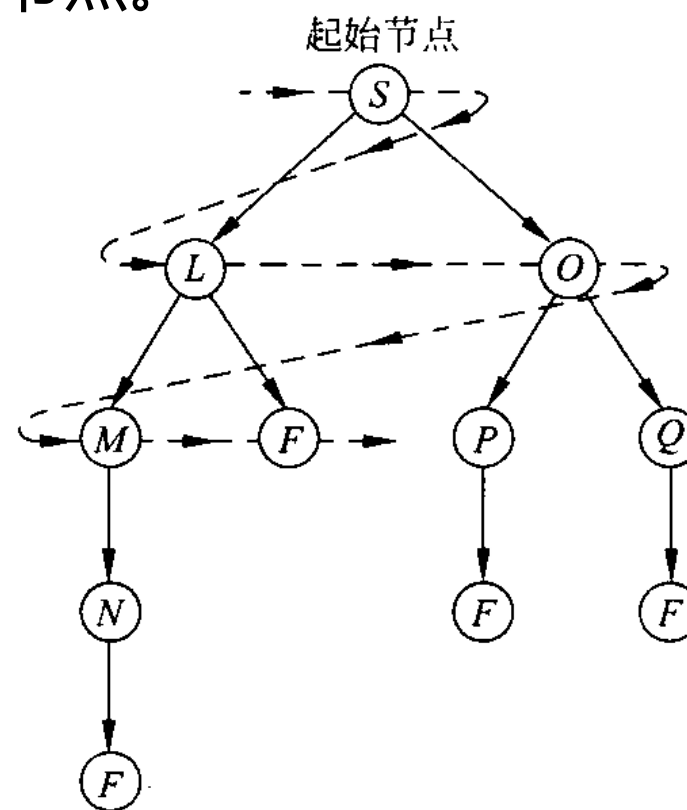
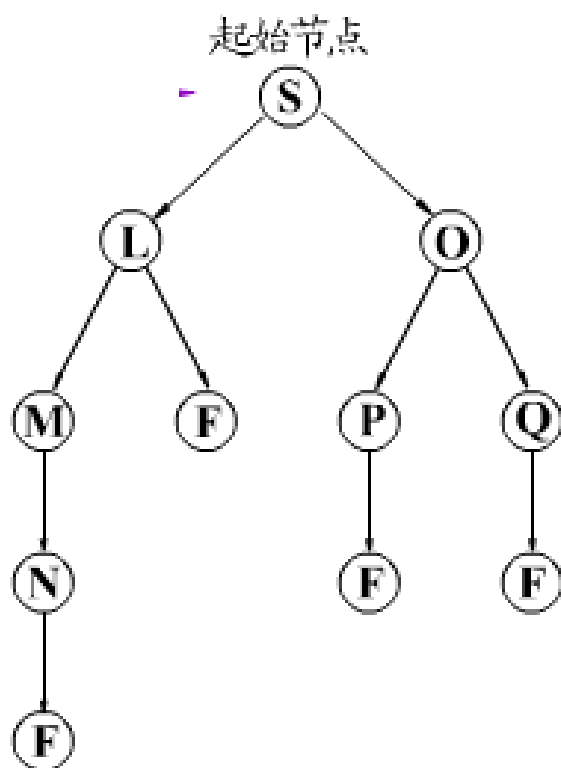


## 2. 无信息搜索

### 宽度优先搜索(Breadth-First Search)

**定义：**以接近起始节点的程度依次扩展节点的搜索方法

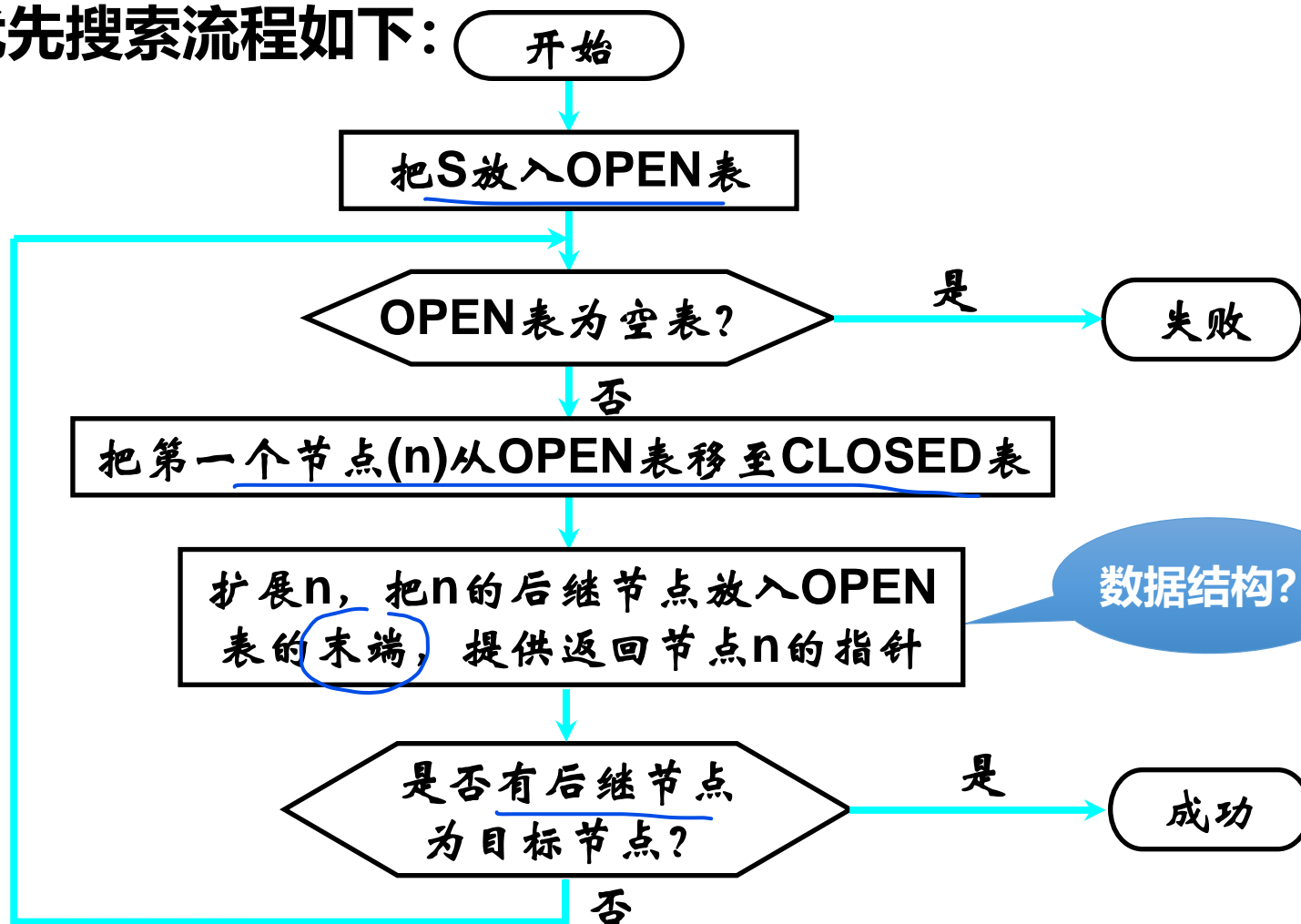
宽度优先搜索是逐层进行的；在对下一层的任一节点进行搜索之前，必须搜索完本层的所有节点。



## 2. 无信息搜索

### 宽度优先搜索(Breadth-First Search)

宽度优先搜索流程如下：



## 2. 无信息搜索

### 宽度优先搜索(Breadth-First Search)

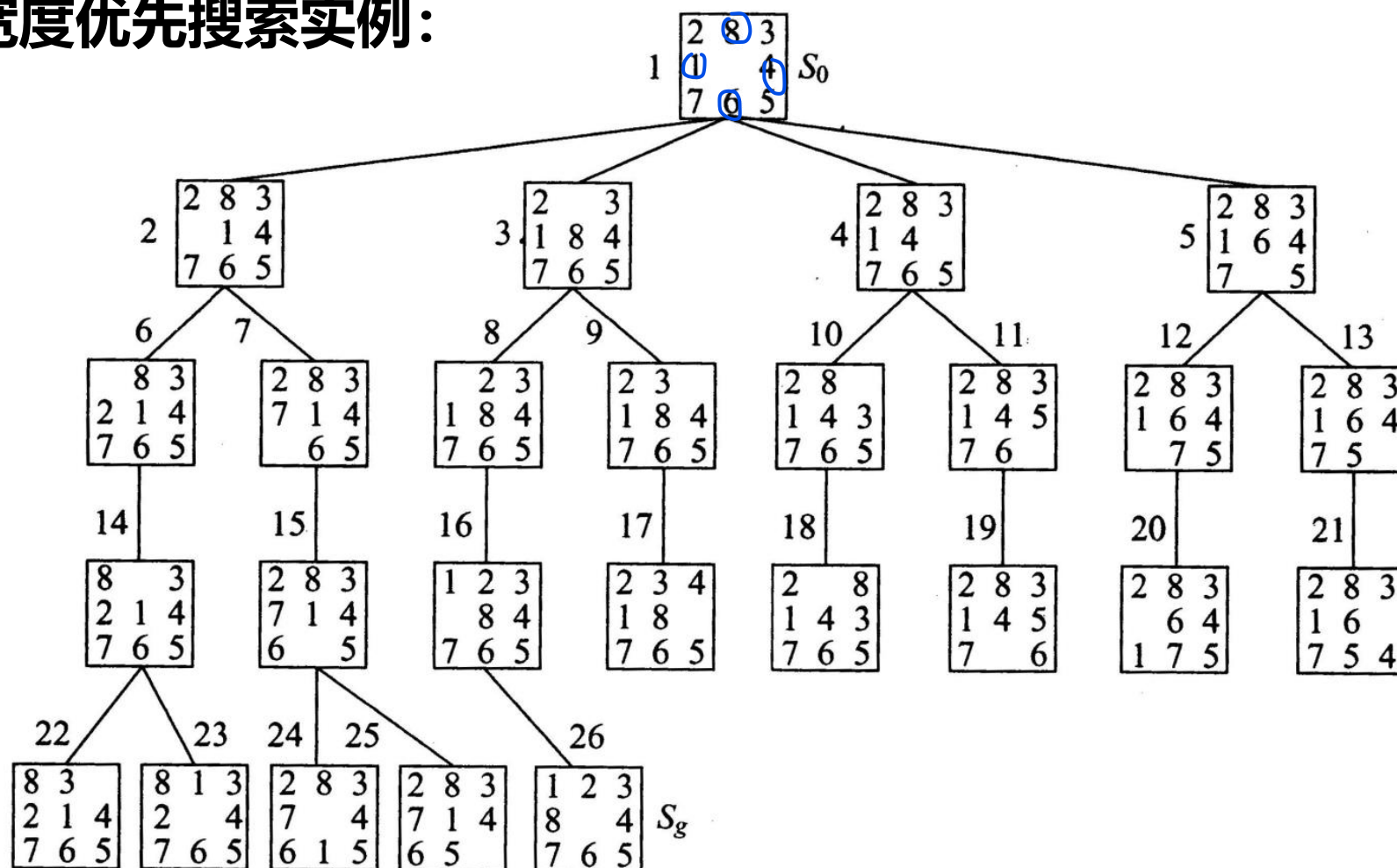
宽度优先搜索算法如下：

- (1) 把起始节点S放到OPEN表中(如果该起始节点为一目标节点，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) 把第一个节点(节点n)从OPEN表移出，并把它放入CLOSED扩展节点表中。
- (4) 扩展节点n。如果没有后继节点，则转向上述第(2)步。
- (5) 把n的所有后继节点放到OPEN表的末端，并提供从这些后继节点回到n的指针。
- (6) 如果n的任一个后继节点是个目标节点，则找到一个解答，成功退出；否则转向第(2)步。

## 2. 无信息搜索

### 宽度优先搜索(Breadth-First Search)

宽度优先搜索实例：



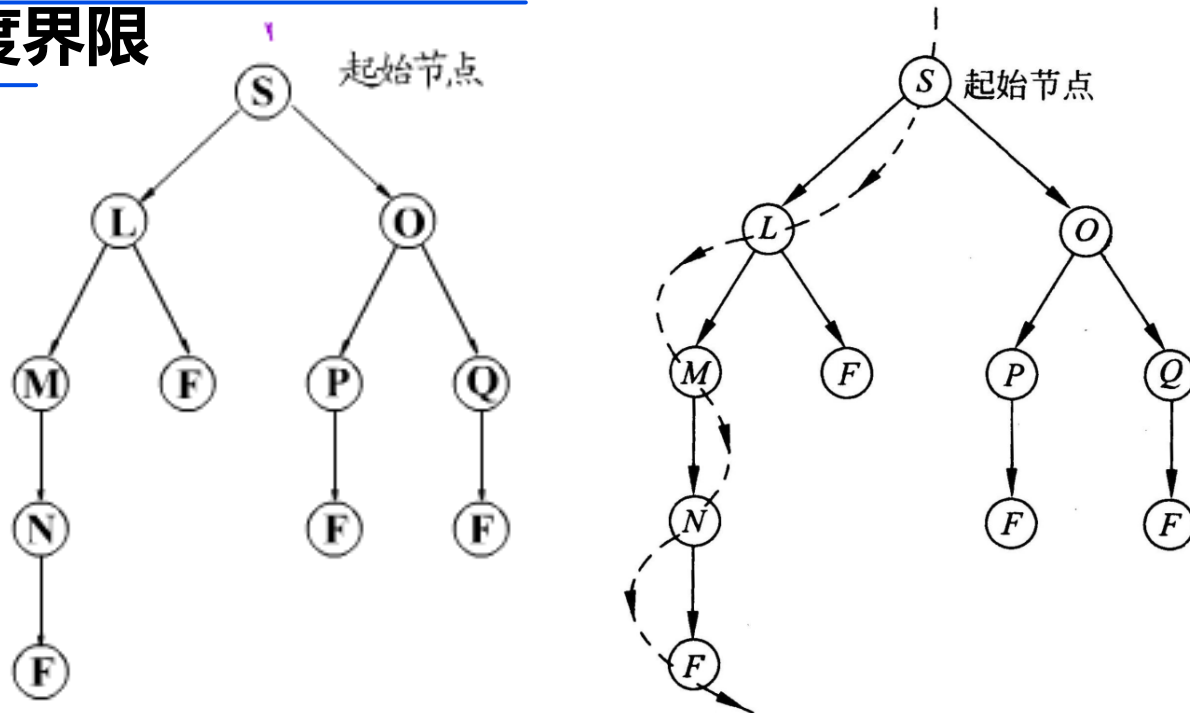
## 2. 无信息搜索

### 深度优先搜索(Depth-First Search)

**定义：** 优先扩展最新产生的(即最深的)节点的搜索方法

深度优先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；只有当搜索到达一个没有后裔的状态时，它才考虑另一条替代的路径。

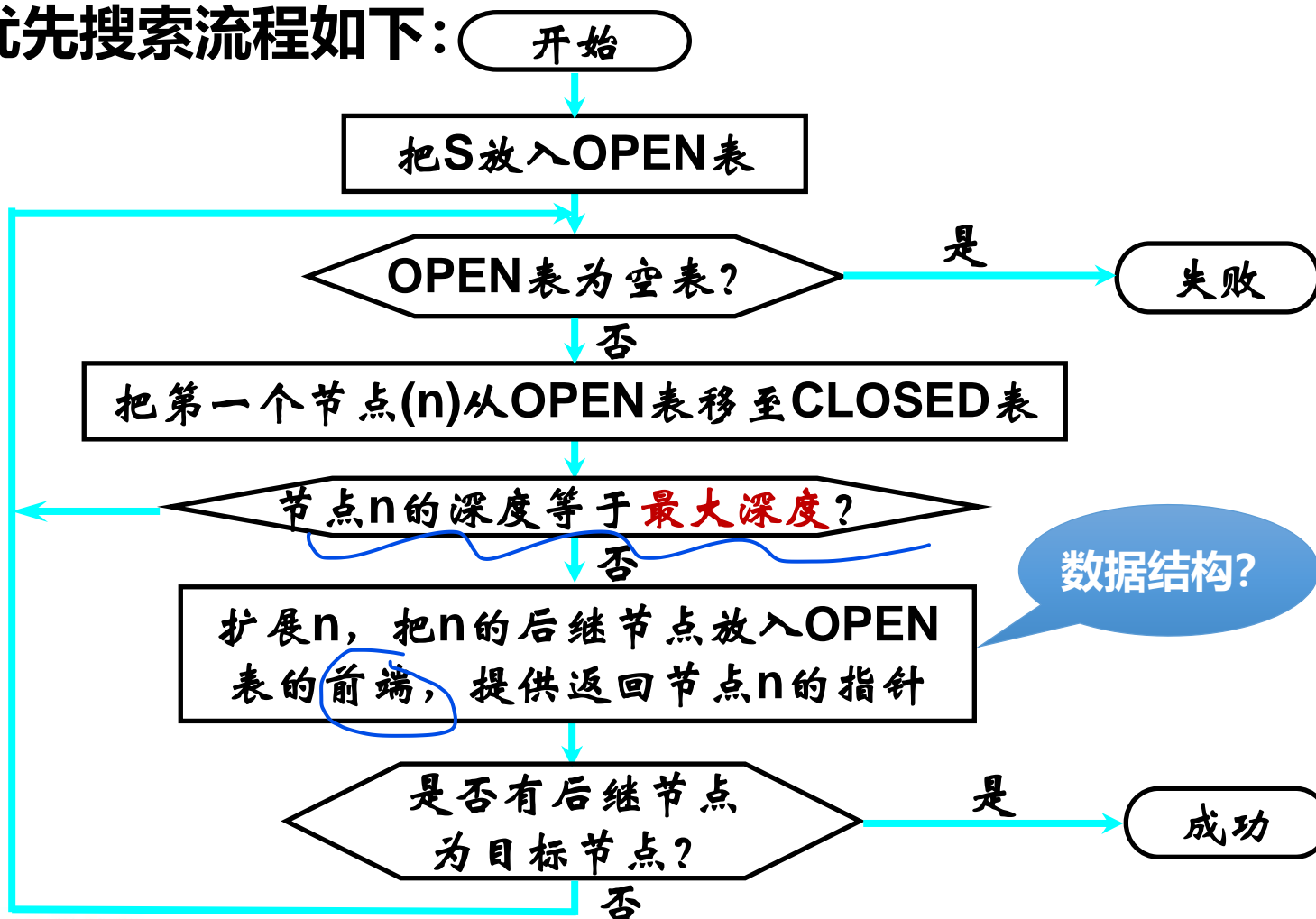
状态空间搜索树的深度可能为无限深，往往给出一个节点扩展的最大深度——深度界限



## 2. 无信息搜索

### 深度优先搜索(Depth-First Search)

有界深度优先搜索流程如下:



## 2. 无信息搜索

### 深度优先搜索(Depth-First Search)

有界深度优先搜索算法如下：

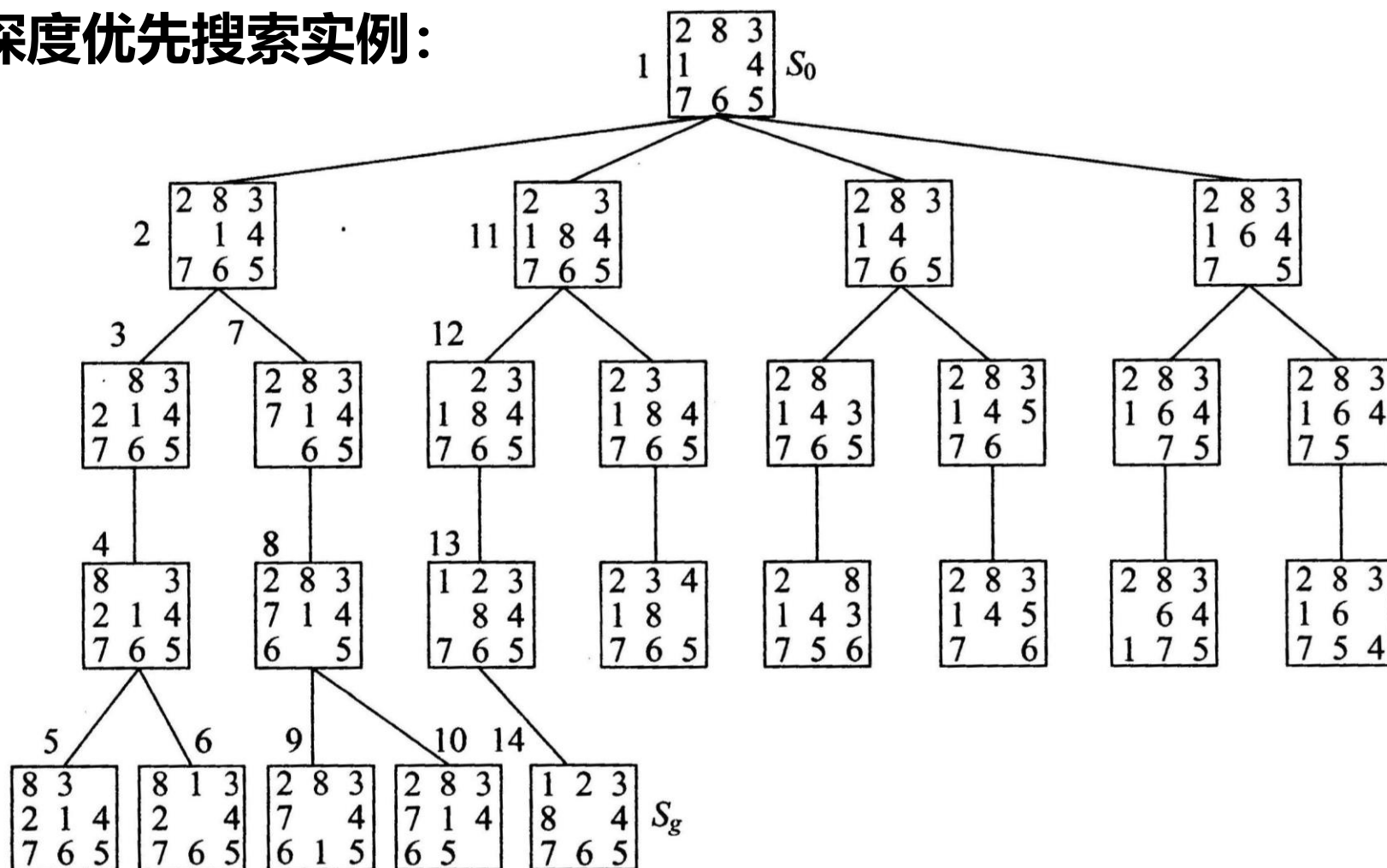
- (1) 把起始节点S放到未扩展节点OPEN表中。如果此节点为一目标节点，则得到一个解。
- (2) 如果OPEN为一空表，则失败退出。
- (3) 把第一个节点(节点n)从OPEN表移到CLOSED表。
- (4) 如果节点n的深度等于**最大深度**，则转向(2)。
- (5) 扩展节点n，产生其全部后裔，并把它们放入OPEN表的前端，提供返回节点n的指针，如果没有后裔，则转向(2)。
- (6) 如果后继节点中有任一个为目标节点，则求得一个解，成功退出；否则，转向(2)。



## 2. 无信息搜索

### 深度优先搜索(Breadth-Frist Search)

深度优先搜索实例：

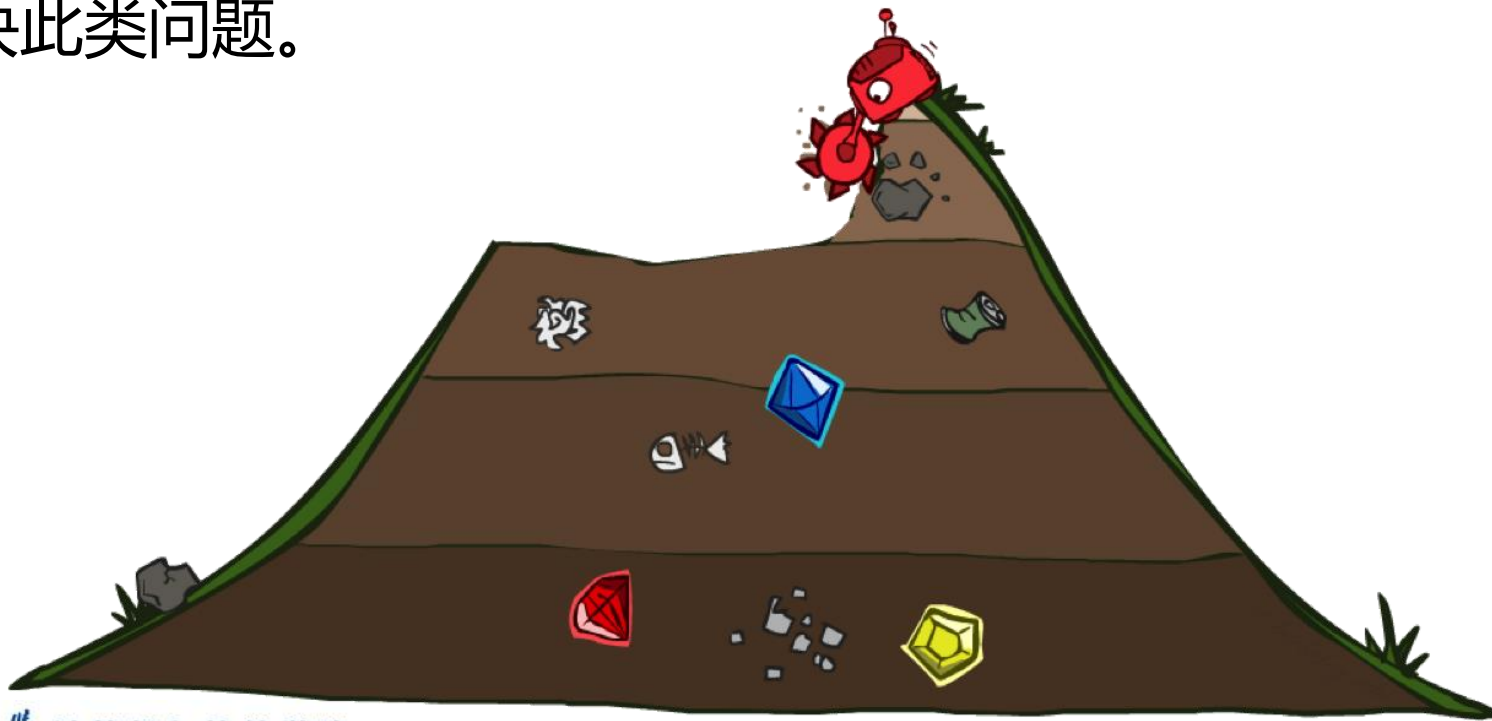


## 2. 无信息搜索

### 一致代价搜索(Uniform-Cost Search)

**定义：**搜索从起始状态至目标状态的具有最小代价解的方法

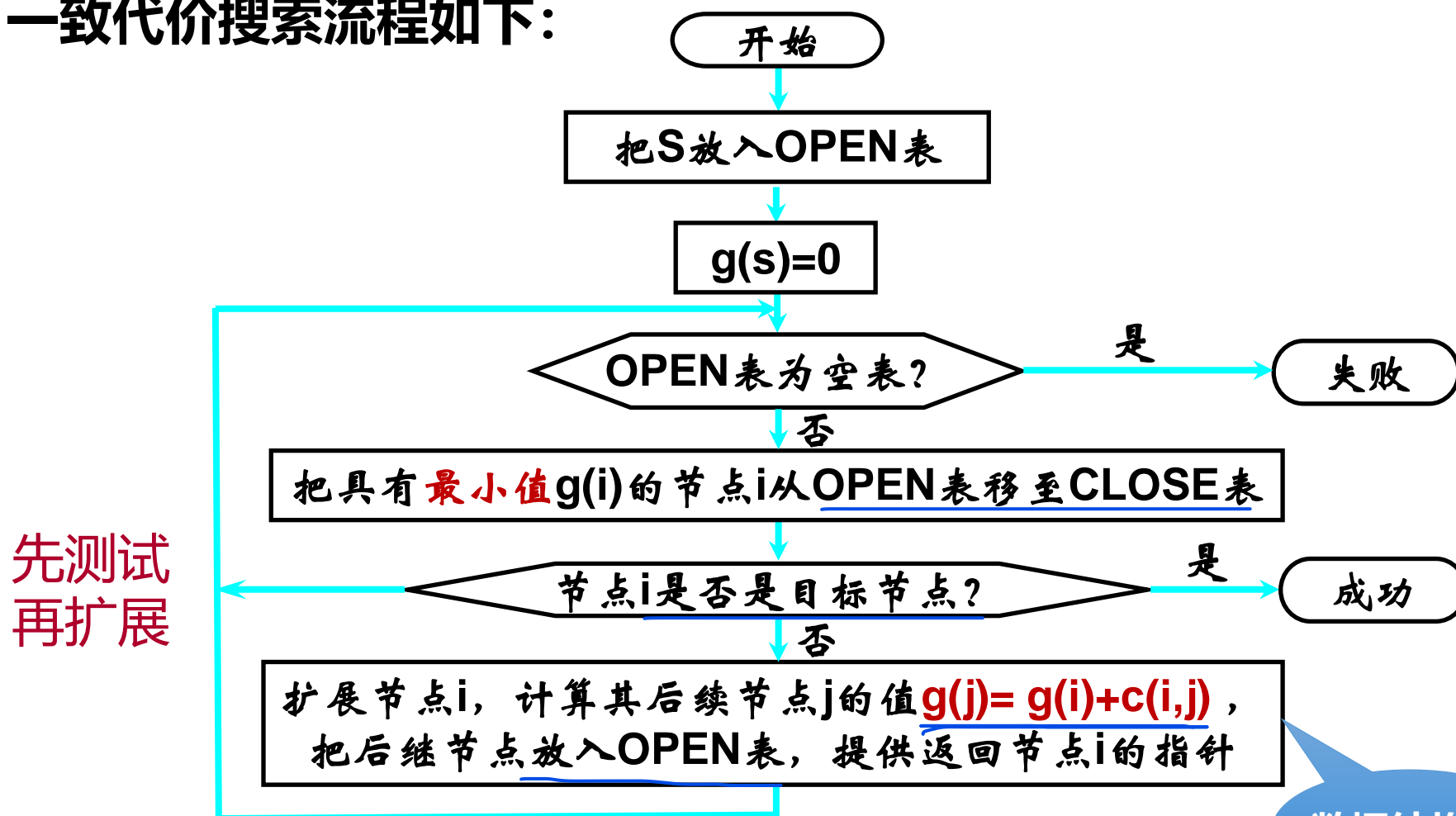
搜索树中每条连接弧线表示有关代价，求得具有最小代价的解答路径。宽度优先（代价都为1）搜索可被推广用来解决此类问题。



## 2. 无信息搜索

### 一致代价搜索(Uniform-Cost Search)

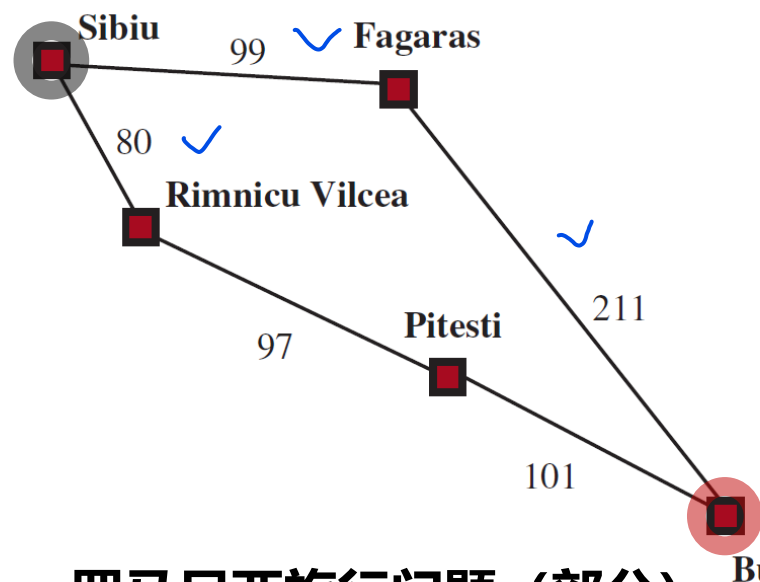
一致代价搜索流程如下：



## 2. 无信息搜索

### 一致代价搜索(Uniform-Cost Search)

#### 一致代价搜索实例：



罗马尼亚旅行问题（部分）

Sibiu(0)

Fagaras=0+99, Rimnicu Vilcea=0+80

Rimnicu Vilcea

Fagaras=0+99, Pitesti=80+97

Fagaras

Bucharest=99+211, Pitesti=80+97

是否终止？

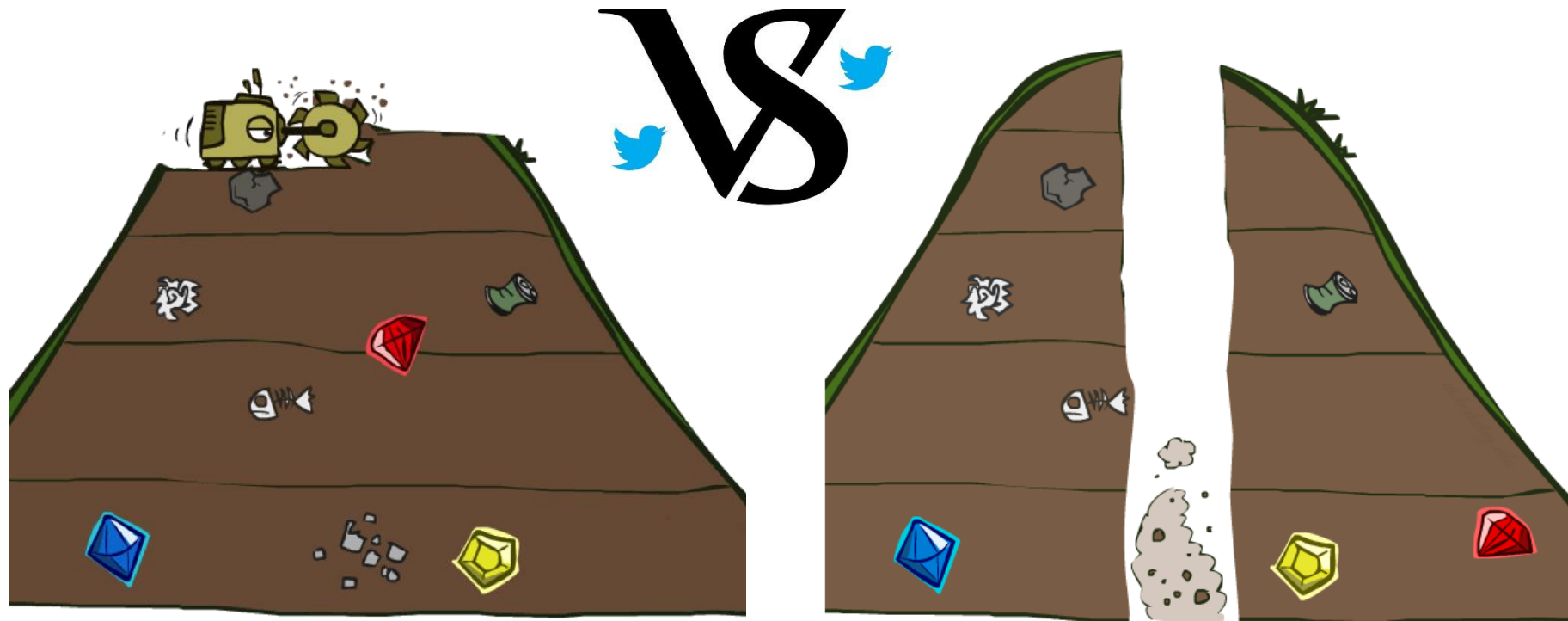
Bucharest(Pitesti)=80+97+101=278

一致代价搜索终止条件？

能找到

## 2. 无信息搜索

宽度优先(BFS) VS. 深度优先(DFS)



## 2. 无信息搜索

### 搜索算法评价

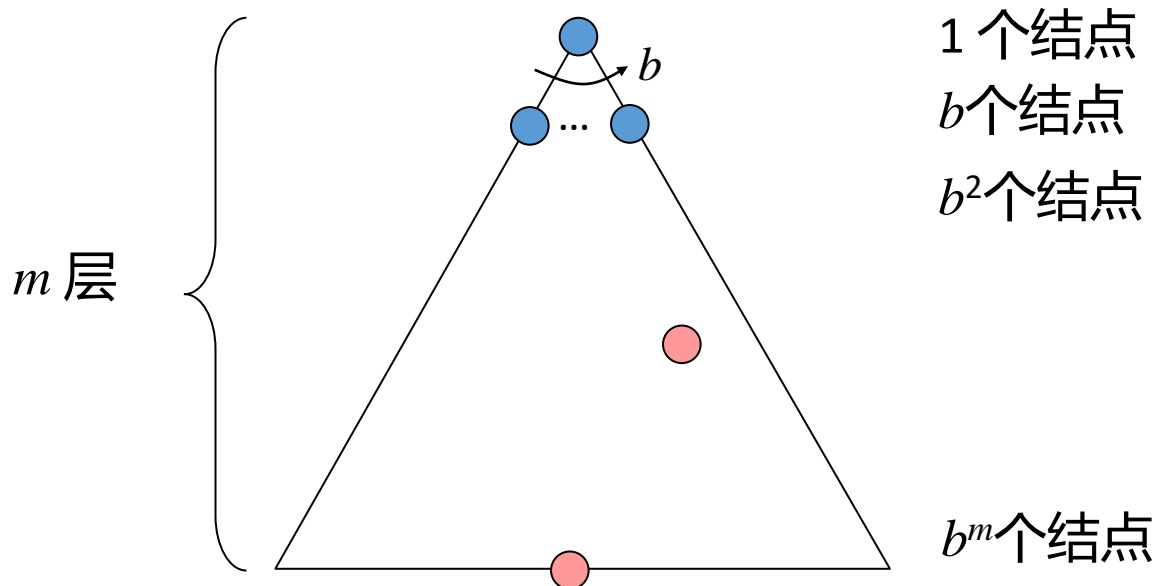
- 完备性
- 最优性
- 时间复杂度
- 空间复杂度

### 搜索树

- $b$ 分支数目
- $m$ 最大深度
- $ball$ 解的深度

整个树的结点数目?

$$1 + b + b^2 + \dots + b^m = O(b^m)$$



## 2. 无信息搜索

### 宽度优先算法特性

#### BFS扩展的结点?

- 最浅解以上所有结点
- 时间复杂度 $O(b^s)$

#### 扩展结点集空间大小?

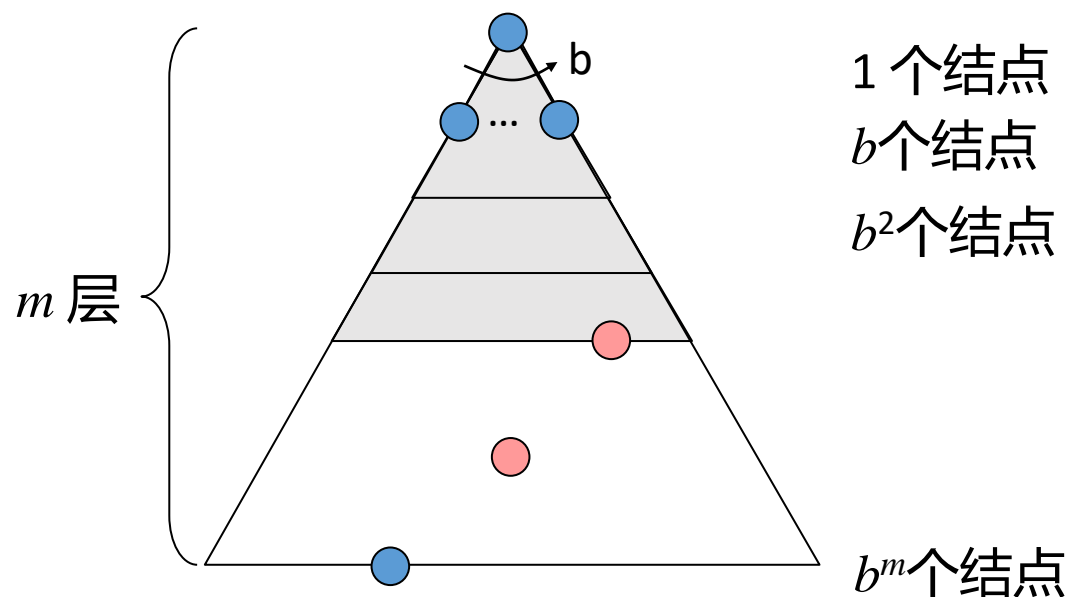
- $O(b^s)$

#### 完备性

- ?

#### 最优性

- ?





## 2. 无信息搜索

### 宽度优先算法特性

#### BFS扩展的结点?

- 最浅解以上所有结点
- 时间复杂度  $O(b^s)$

#### 扩展结点集空间大小?

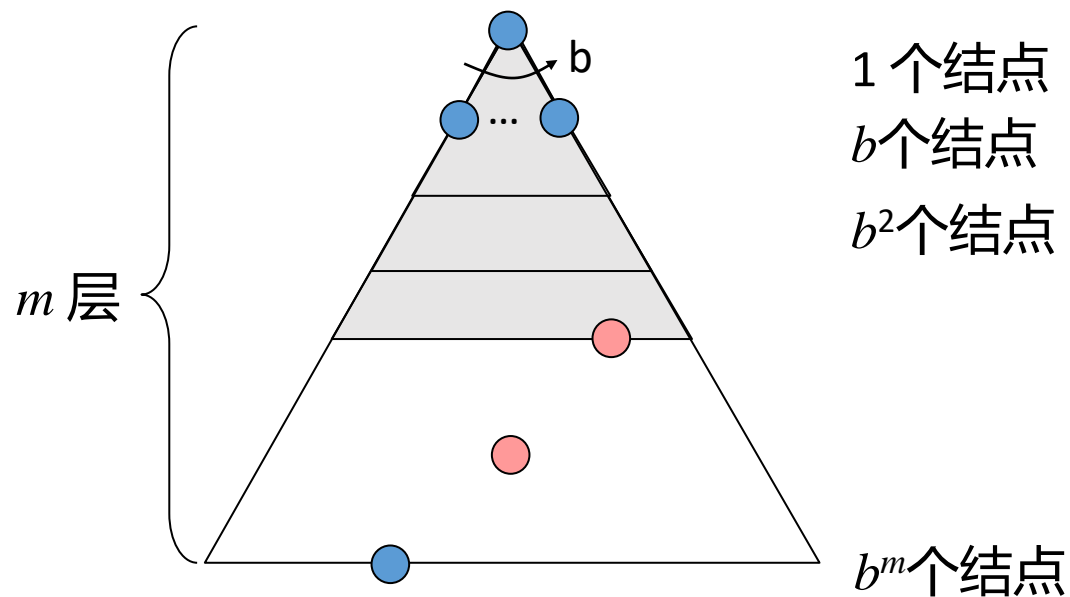
- $O(b^s)$

#### 完备性

- 完备

#### 最优性

- 是



## 2. 无信息搜索

### 深度优先算法特性

#### DFS扩展的结点?

- 树的左边结点
- 可能需要扩展整个树
- 时间复杂度 $O(b^m)$

#### 扩展结点集空间大小?

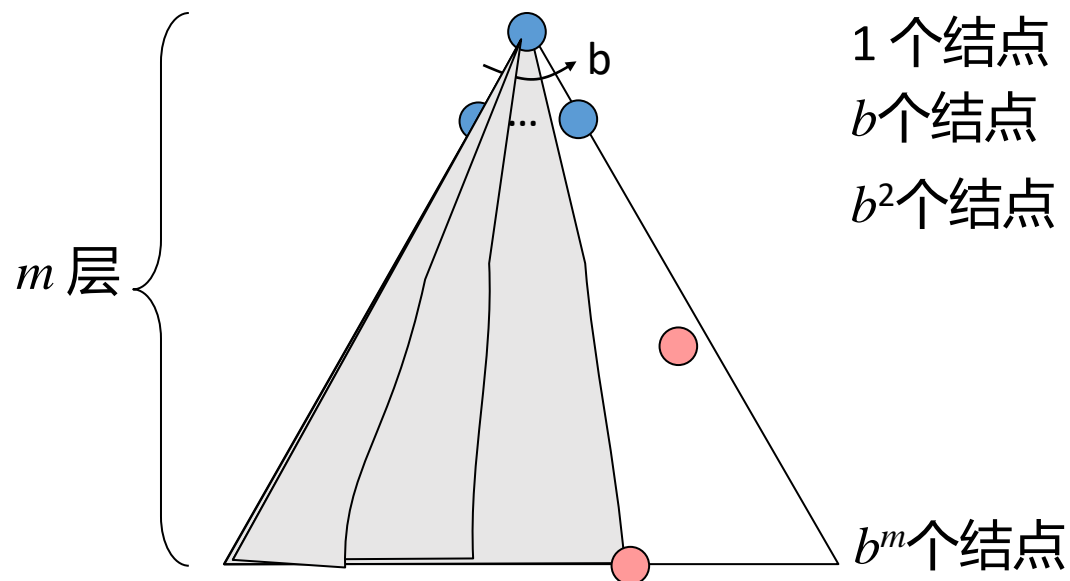
- $O(bm)$

#### 完备性

- ?

#### 最优性

- ?



## 2. 无信息搜索

### 深度优先算法特性

#### DFS扩展的结点?

- 树的左边结点
- 可能需要扩展整个树
- 时间复杂度 $O(b^m)$

#### 扩展结点集空间大小?

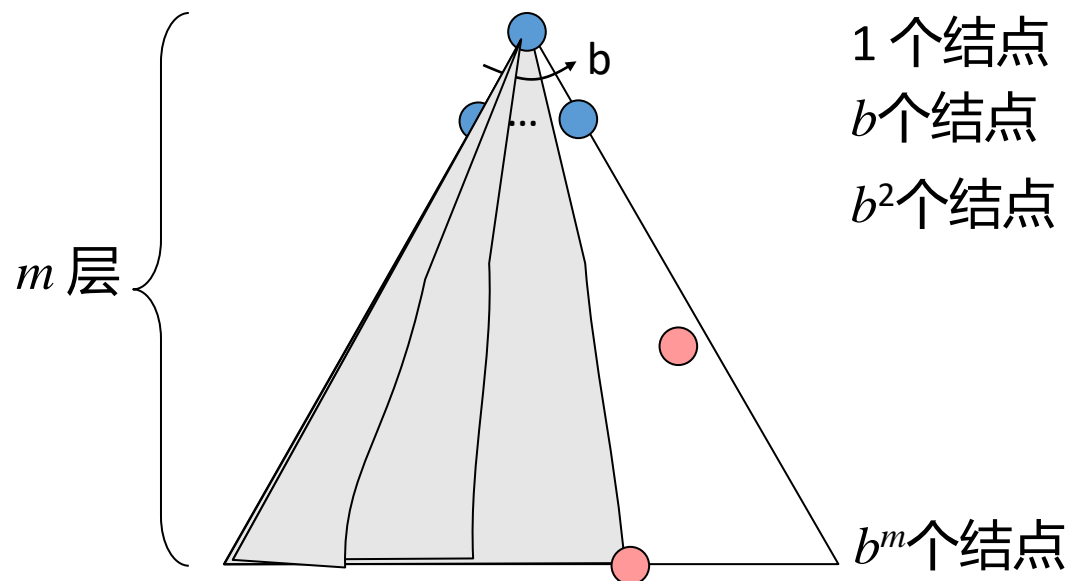
- $O(bm)$

#### 完备性

- 可能不完备 (图有环, 需进行约束)

#### 最优性

- 否, 最左的解 ✕



## 2. 无信息搜索

### 一致代价搜索算法特性

#### UCS扩展的结点?

- 小于最小代价解的所有结点
- 如果解代价为 $C^*$ , 边最小代价为 $\varepsilon$
- 时间复杂度 $O(b^{C^*/\varepsilon})$

#### 扩展结点集空间大小?

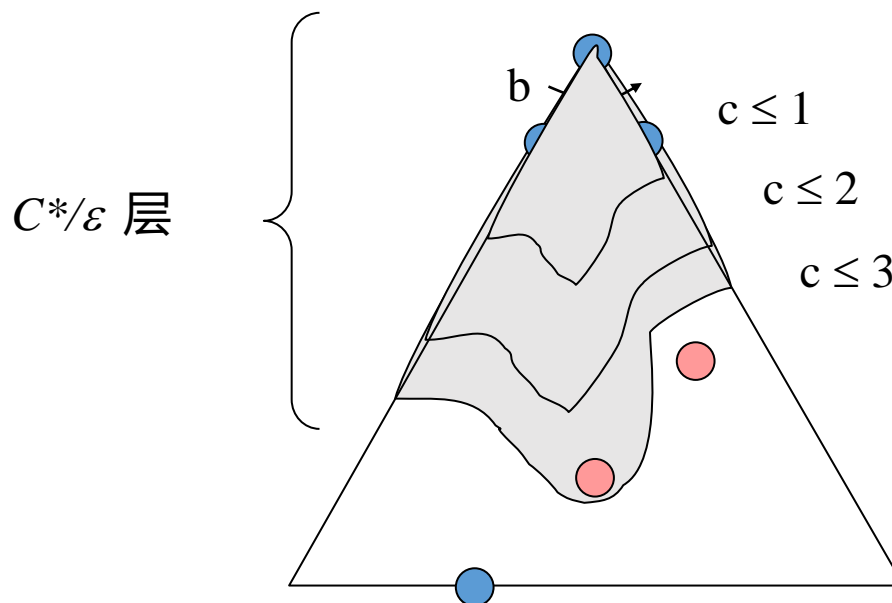
- $O(b^{C^*/\varepsilon})$

#### 完备性

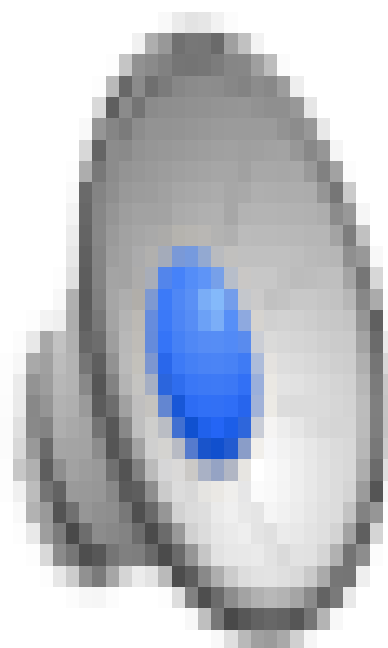
- 完备 (边代价 $>0$ )

#### 最优性

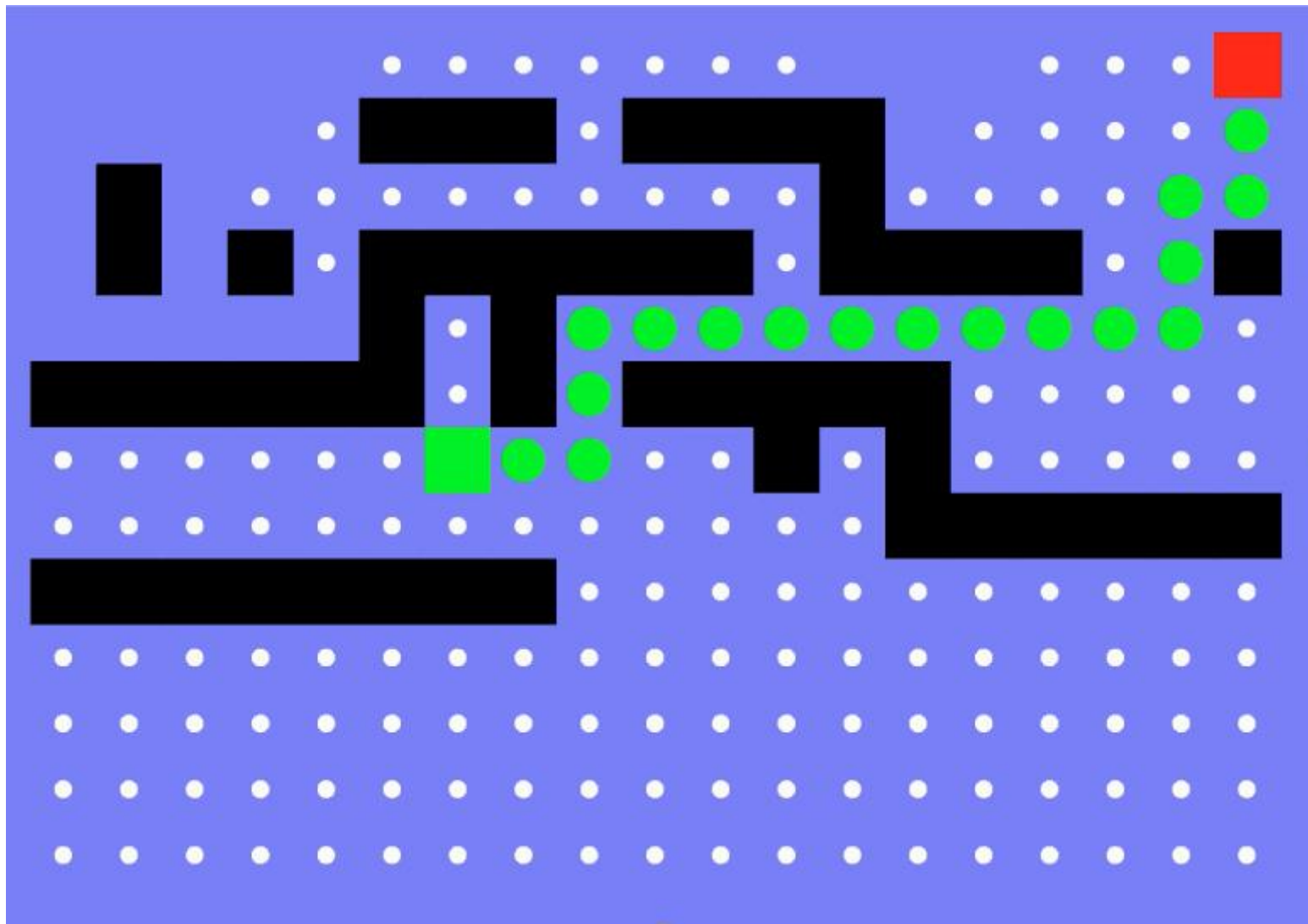
- 是



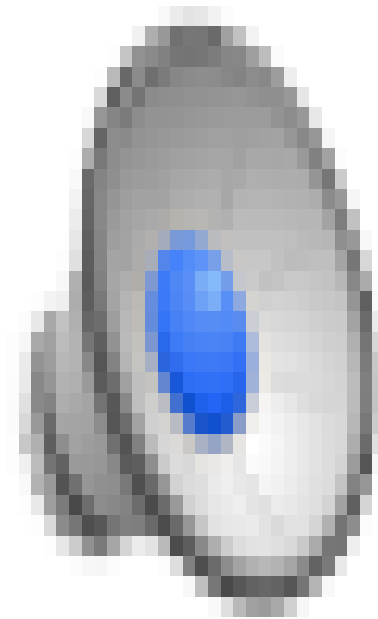
## 2. 无信息搜索



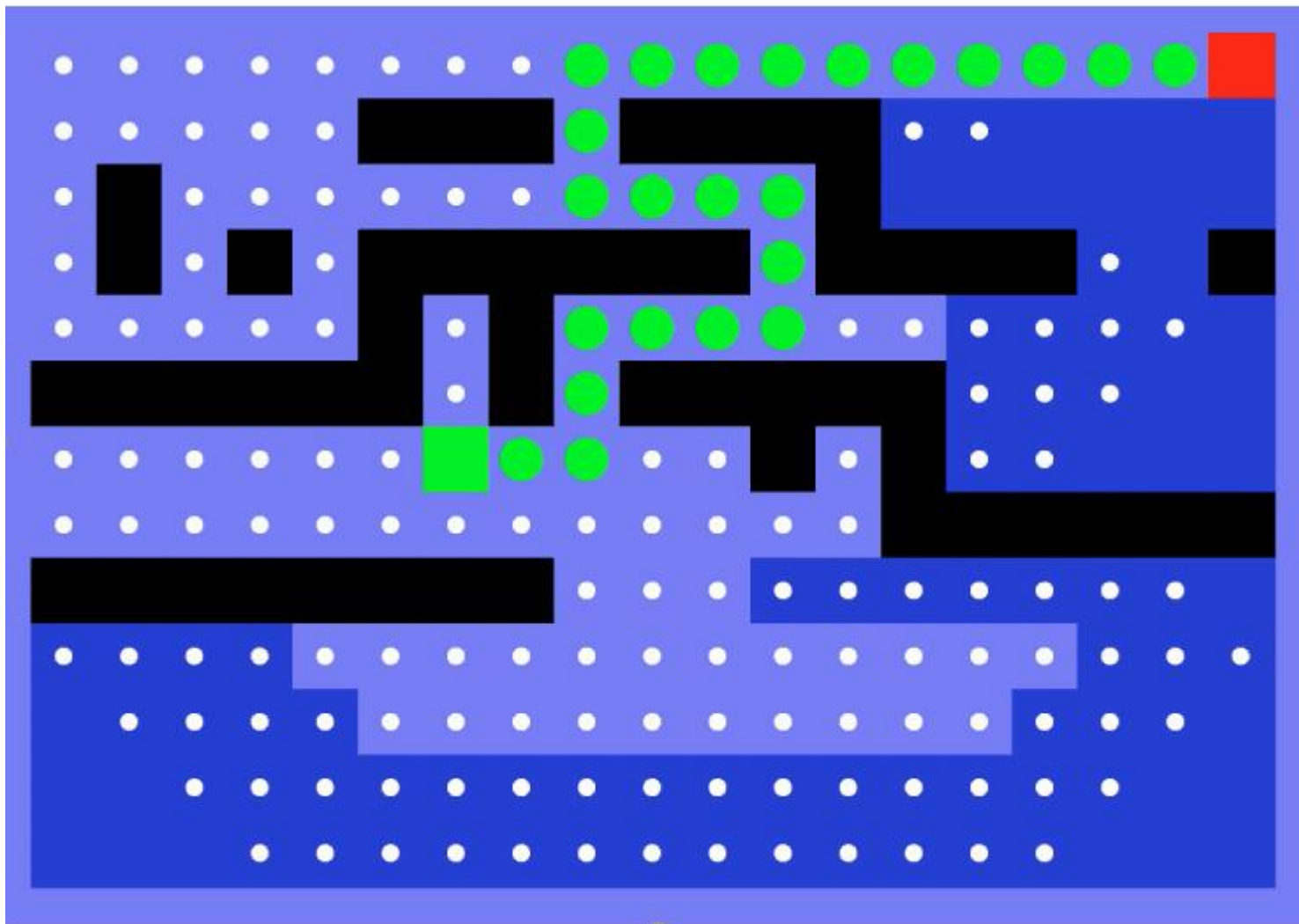
## 2. 无信息搜索



## 2. 无信息搜索

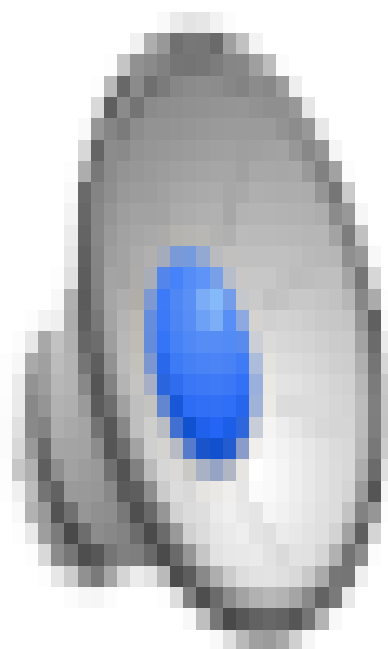


## 2. 无信息搜索

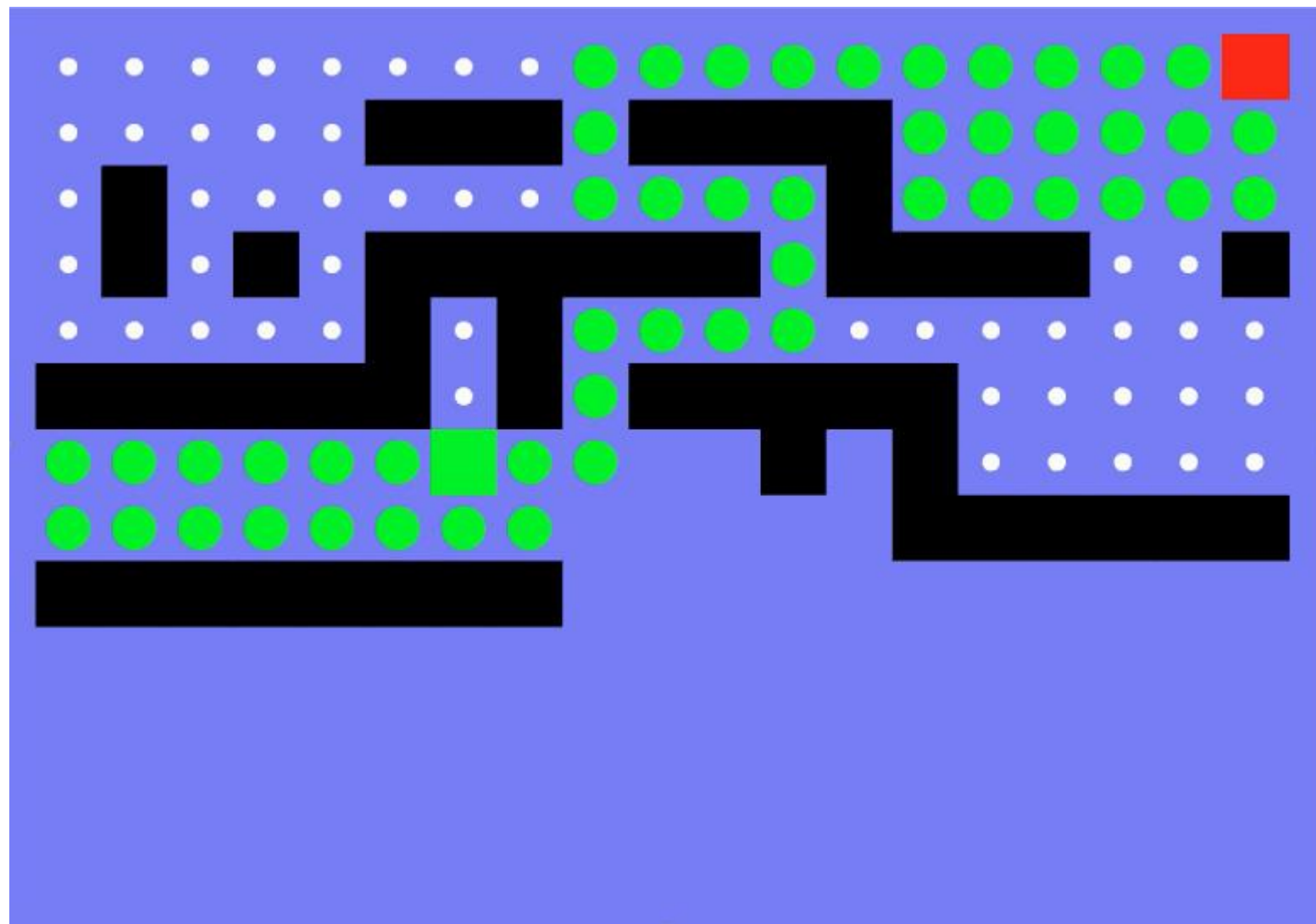




## 2. 无信息搜索



## 2. 无信息搜索

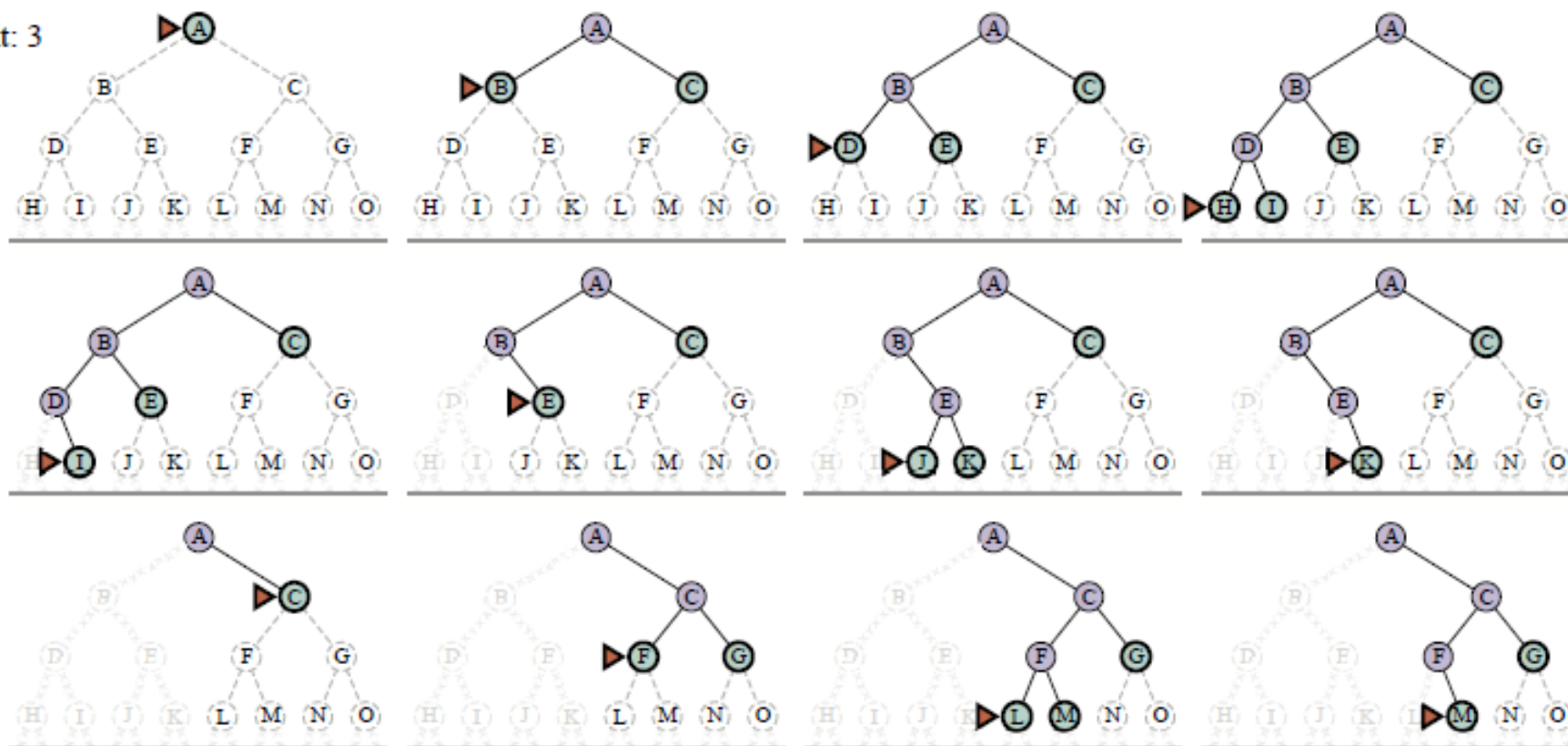


## 2. 无信息搜索

### 其它无信息搜索

### 迭代加深

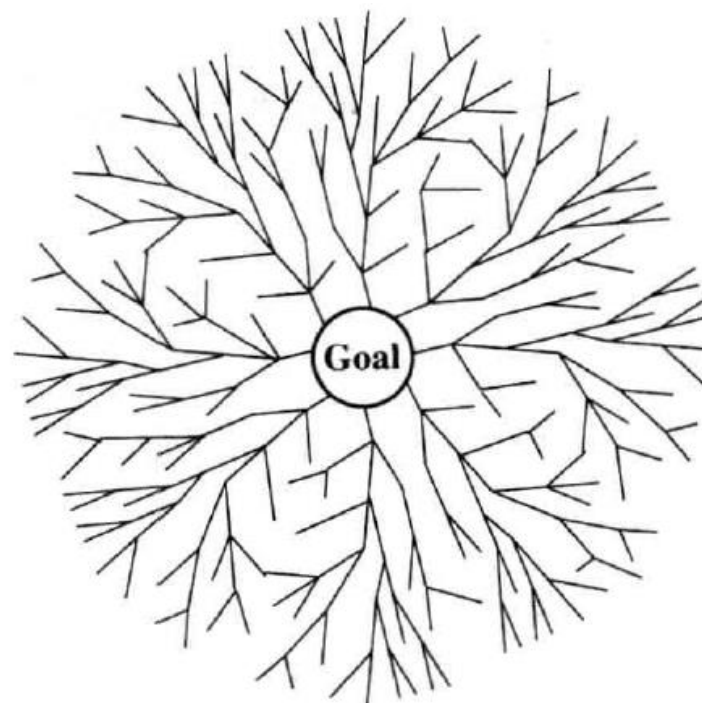
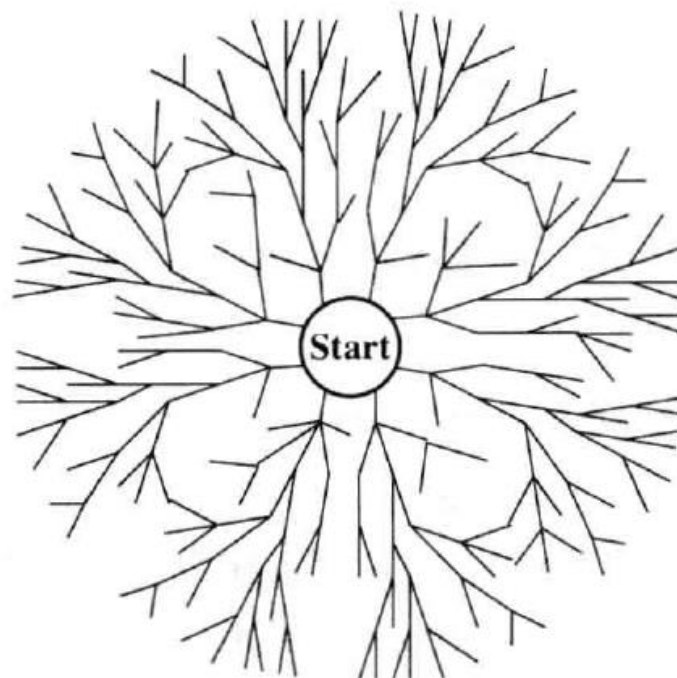
limit: 3



## 2. 无信息搜索

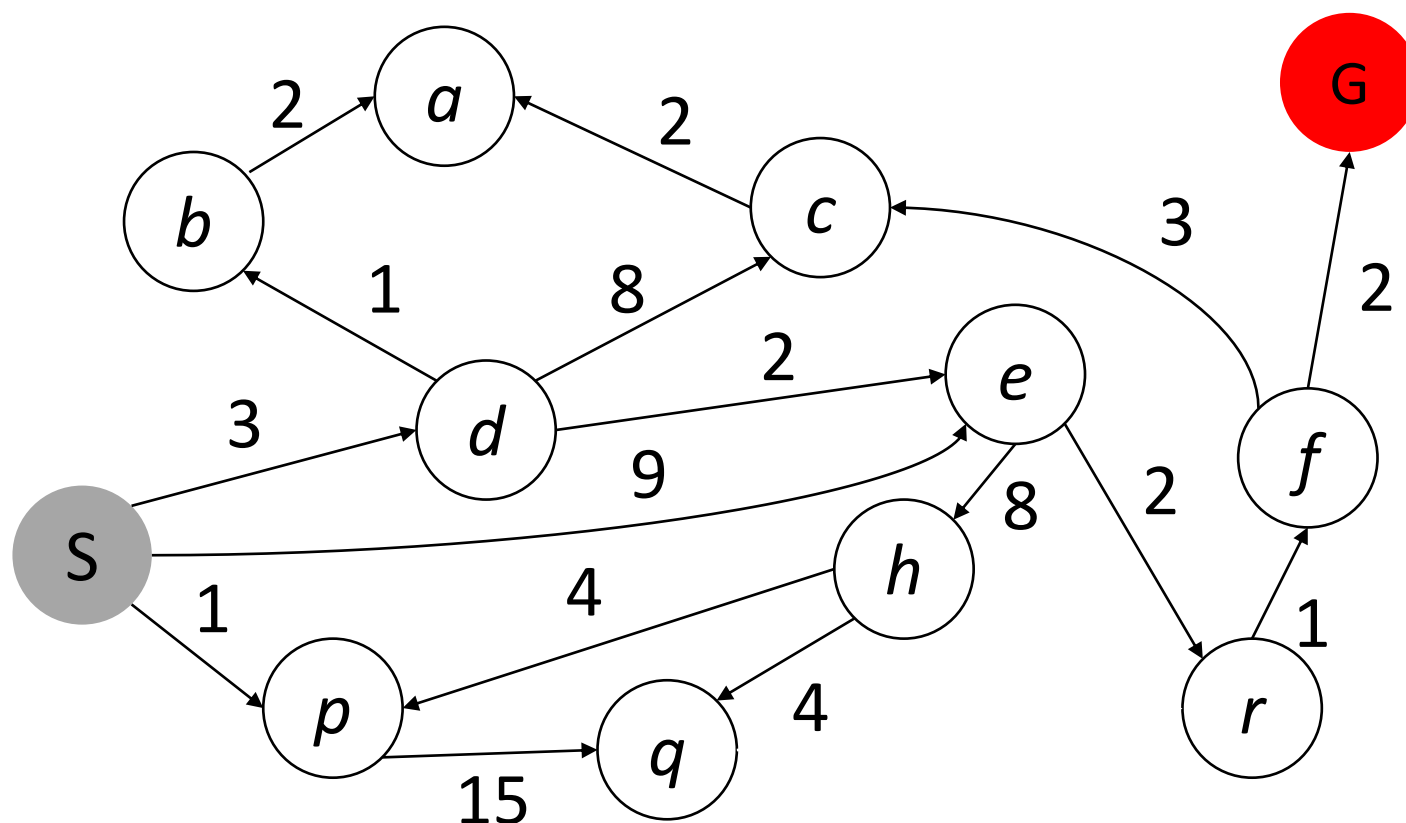
### 其它无信息搜索

### 双向搜索



## 2. 无信息搜索

### 练一练



*Spedqhrbcfag*

# 目录

1

搜索问题求解

2

无信息搜索\*\*

3

有信息搜索\*\*\*

4

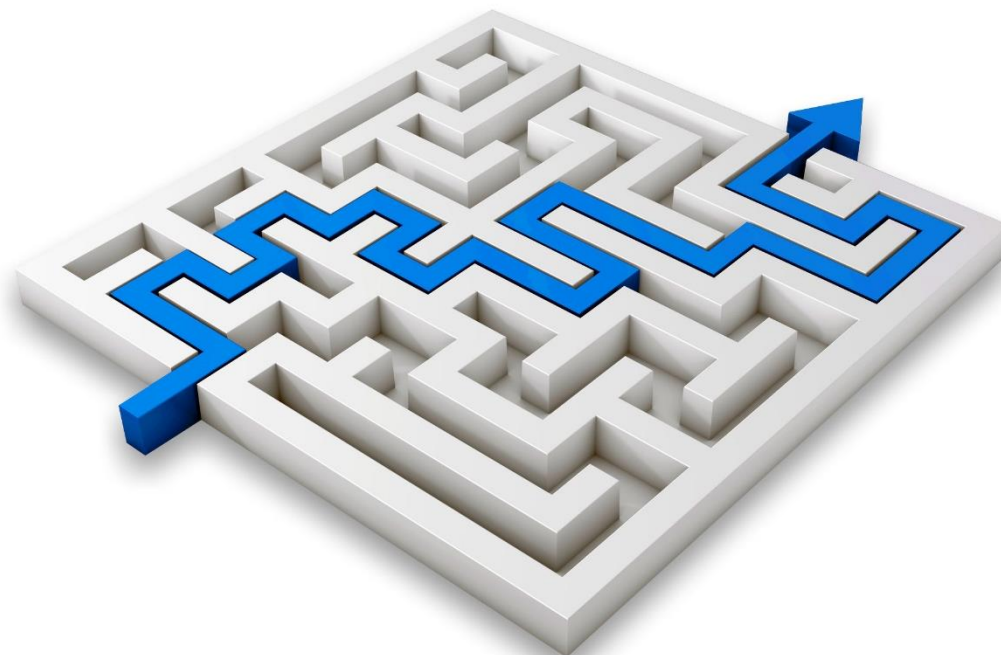
A/A\*搜索\*\*\*\*

5

基本搜索总结

6

习题及实验



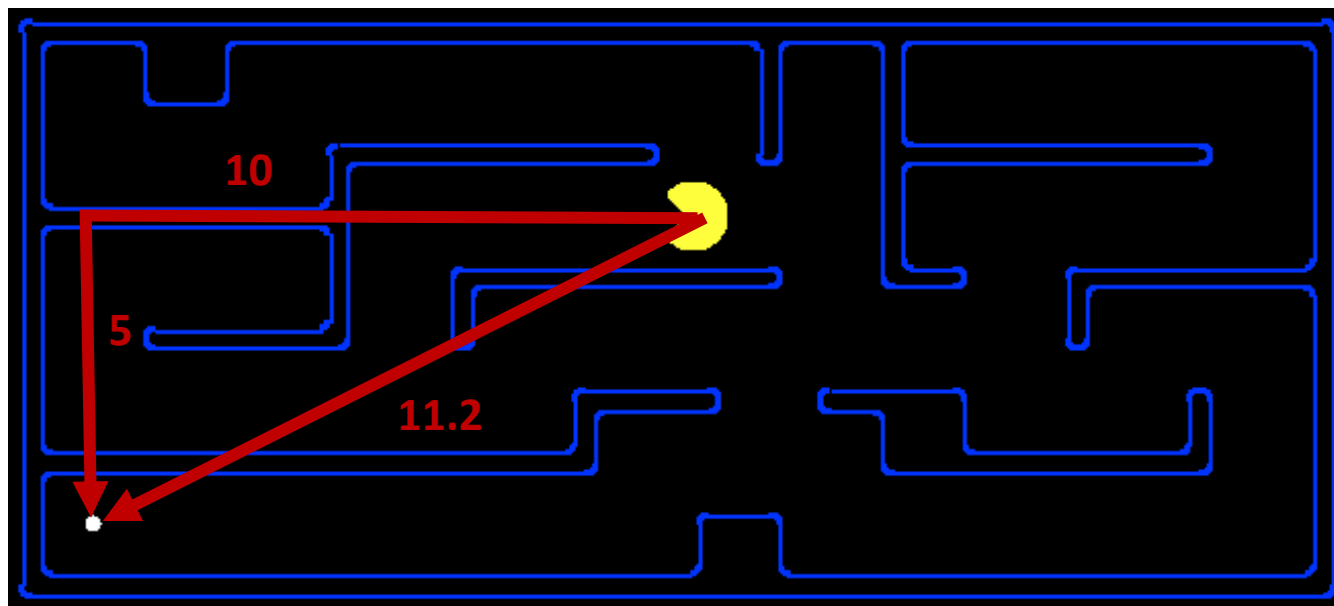
### 3. 有信息搜索

**启发式信息(Heuristic Information):**某些有关具体问题领域的特性信息

**启发式搜索:** 利用与问题有关的启发式信息引导搜索。

**估价函数 $f(n)$ :**用来估算节点希望程度的度量。

信息→度量→排序



2	8	3
1	6	4
7		5



不在位数  
4

1	2	3
8		4
7	6	5



# 3. 有信息搜索

## 贪婪搜索

### 搜索策略：

扩展看起来离目标**最近**的结点

### 估价函数：

$f(n) = h(n)$  (启发式：估计结点 $n$ 到目标的代价)





# 3. 有信息搜索

## 从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(x)$

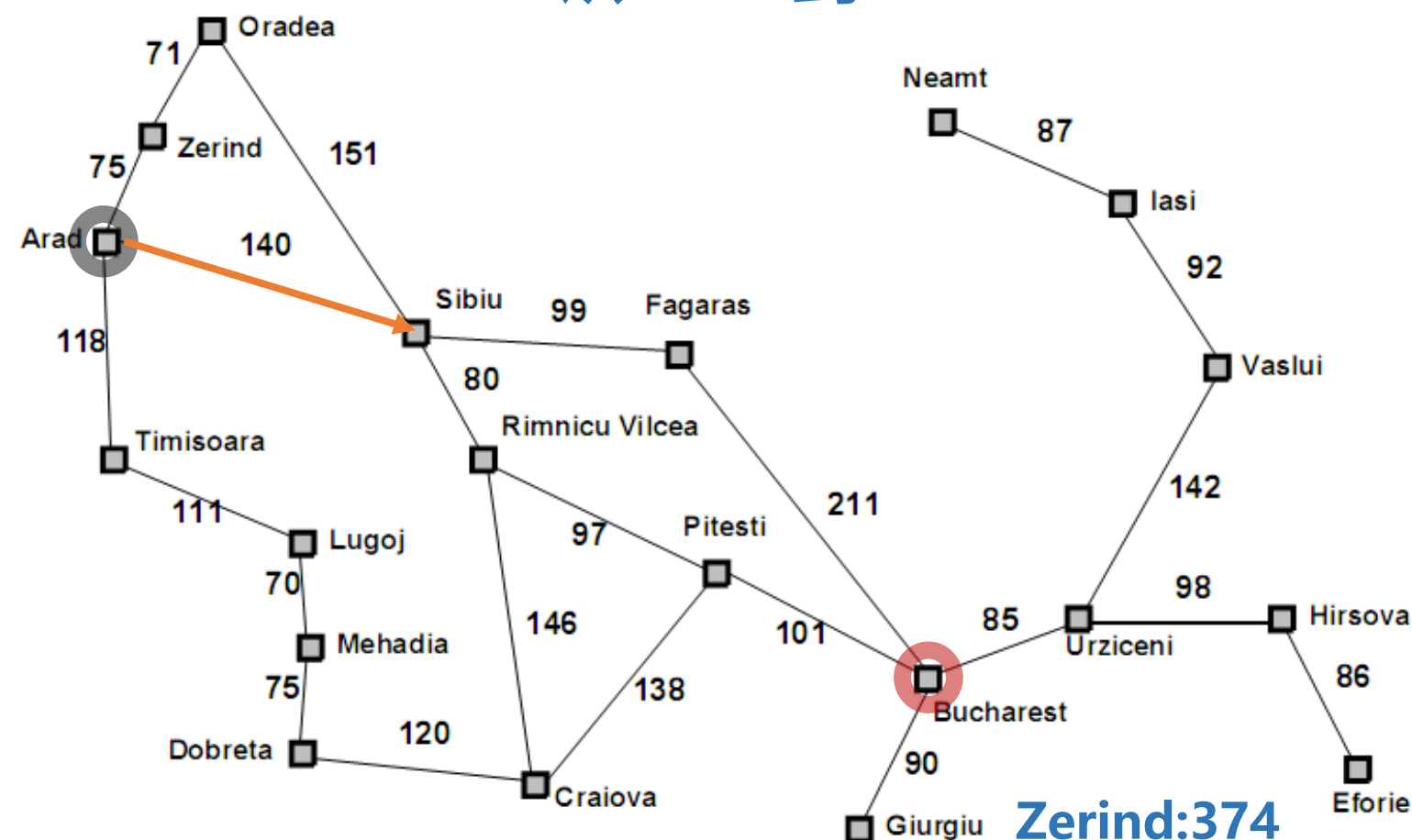
Arad



Zerind:374

Sibiu:253

Timisoara:329



# 3. 有信息搜索

## 从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(x)$

Sibiu

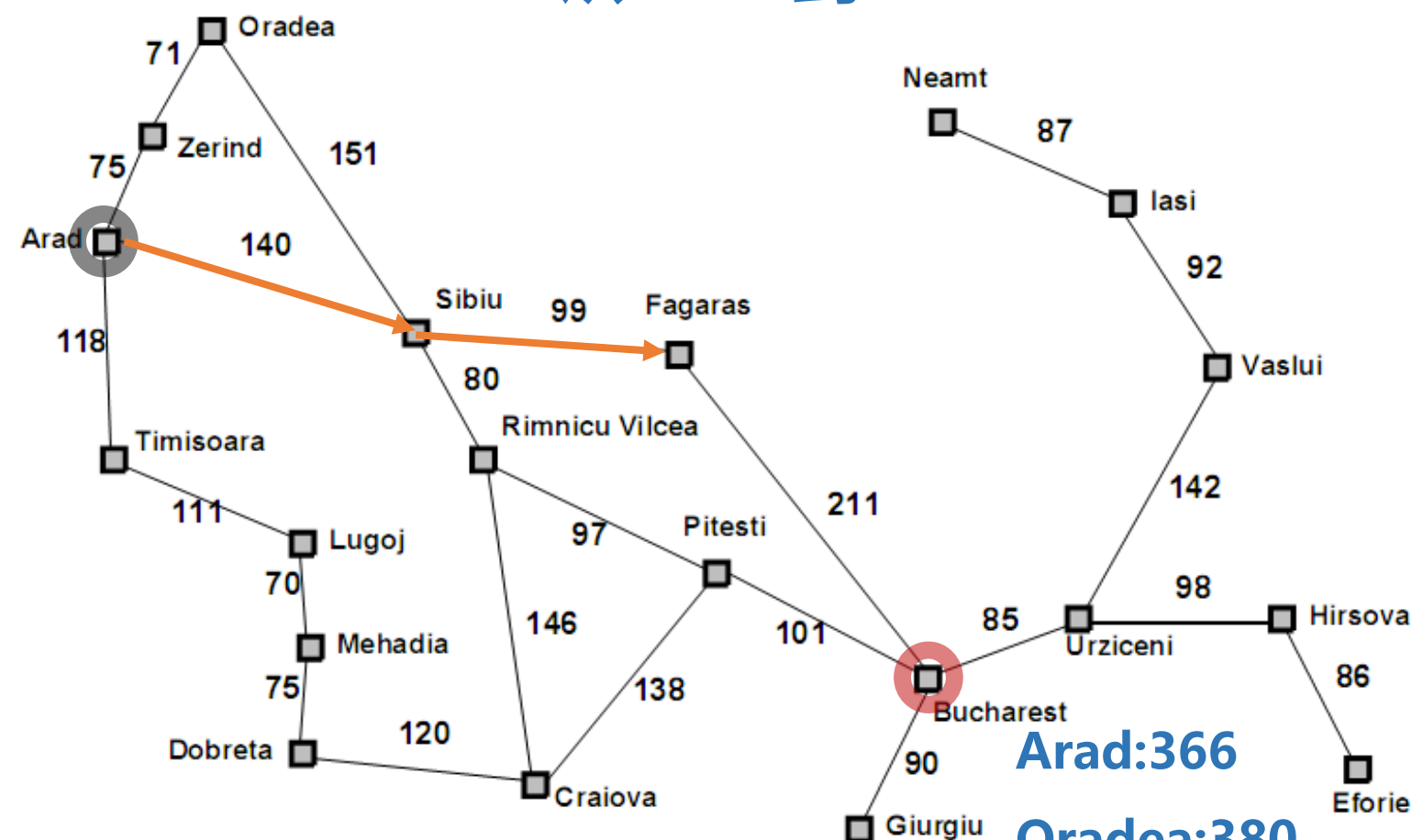


Arad:366

Oradea:380

Fagaras:178

Rimnicu Vilcea:193



# 3. 有信息搜索

## 从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

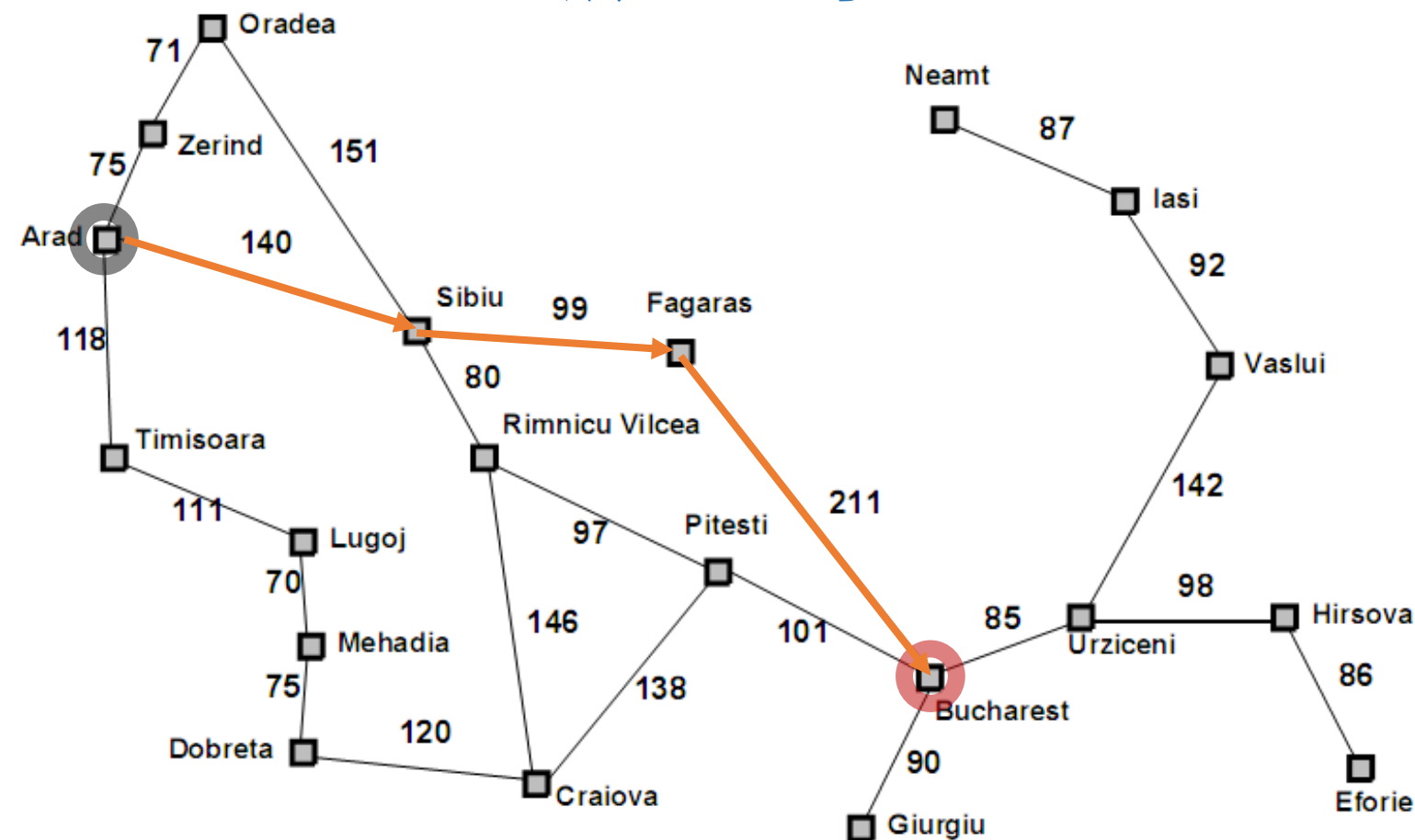
$h(x)$

Fagaras



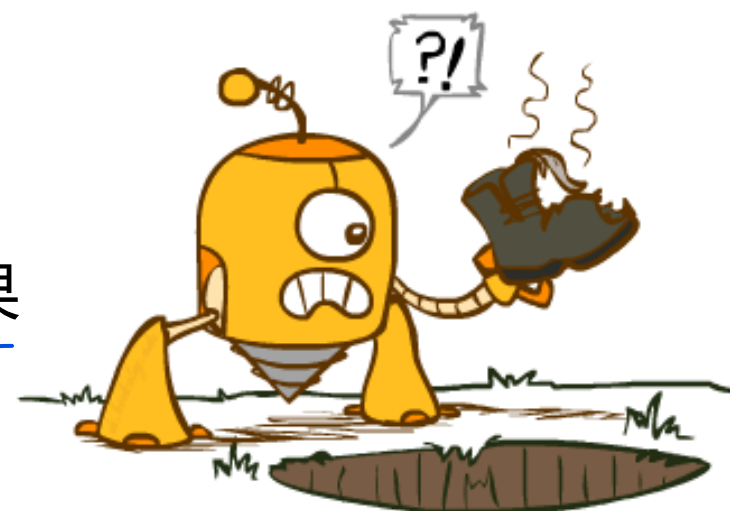
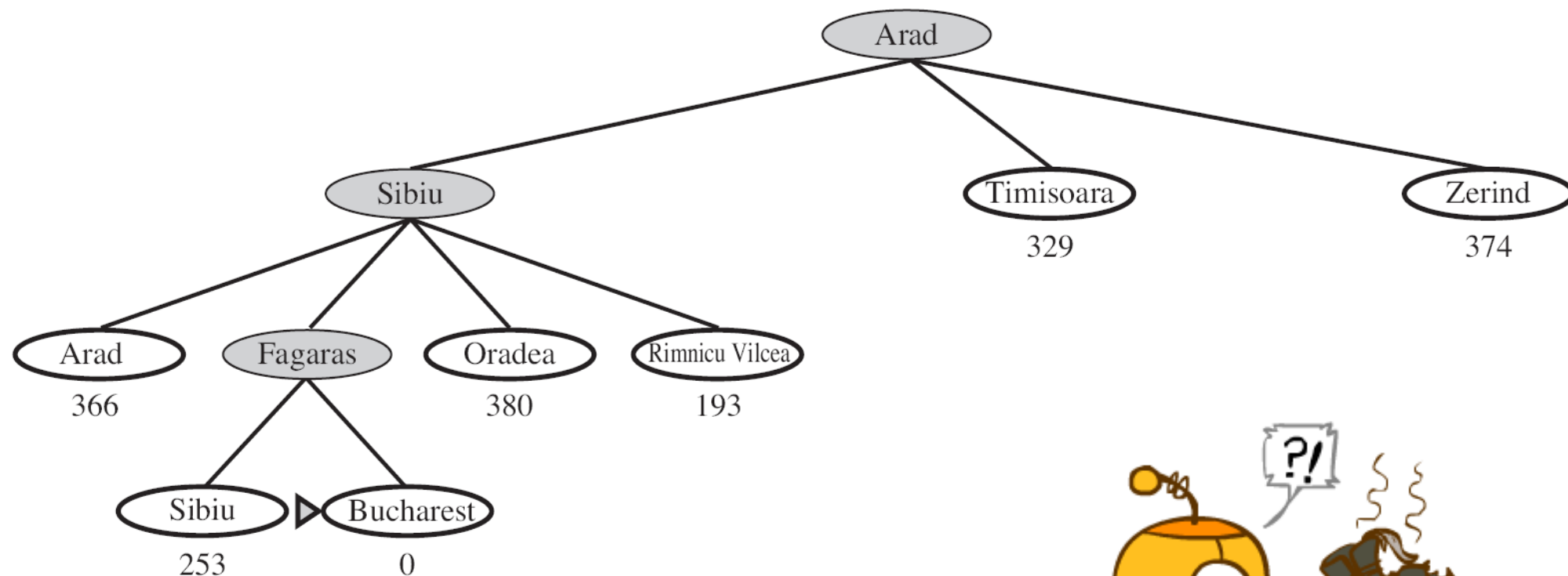
**Bucharest:0**

**Sibiu:253**



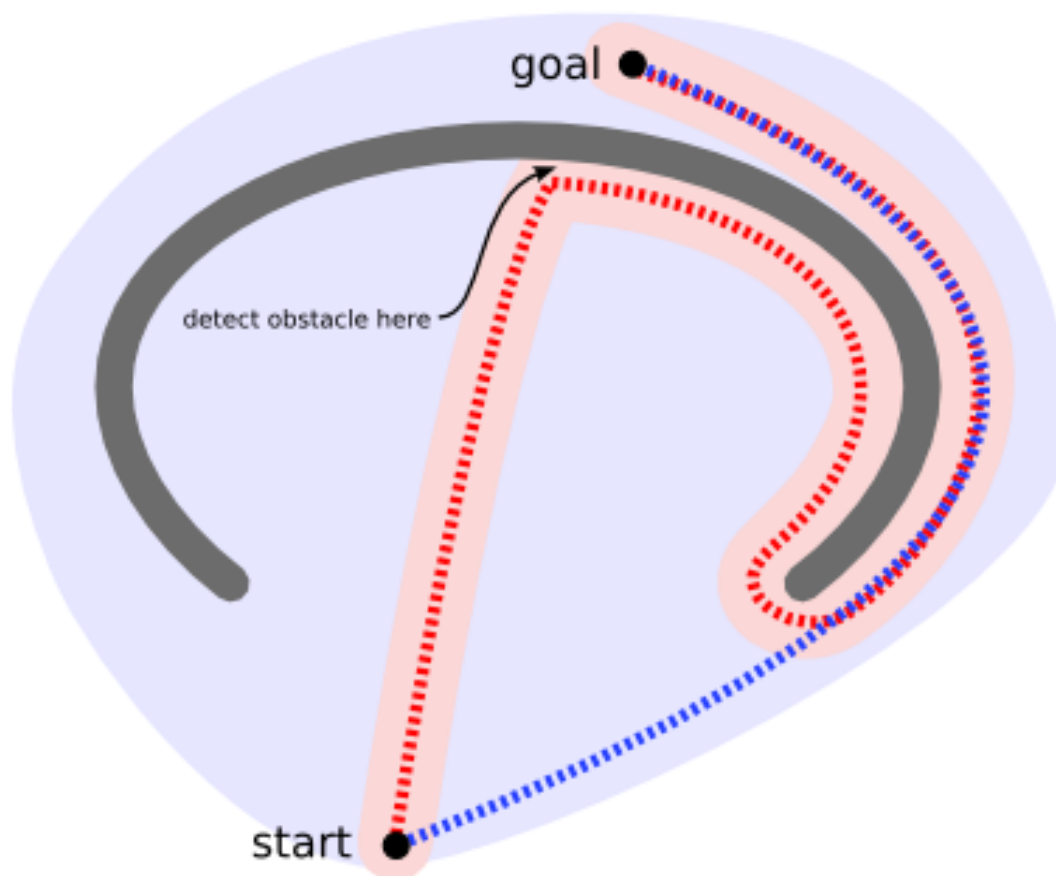
# 3. 有信息搜索

## 从Arad到Bucharest



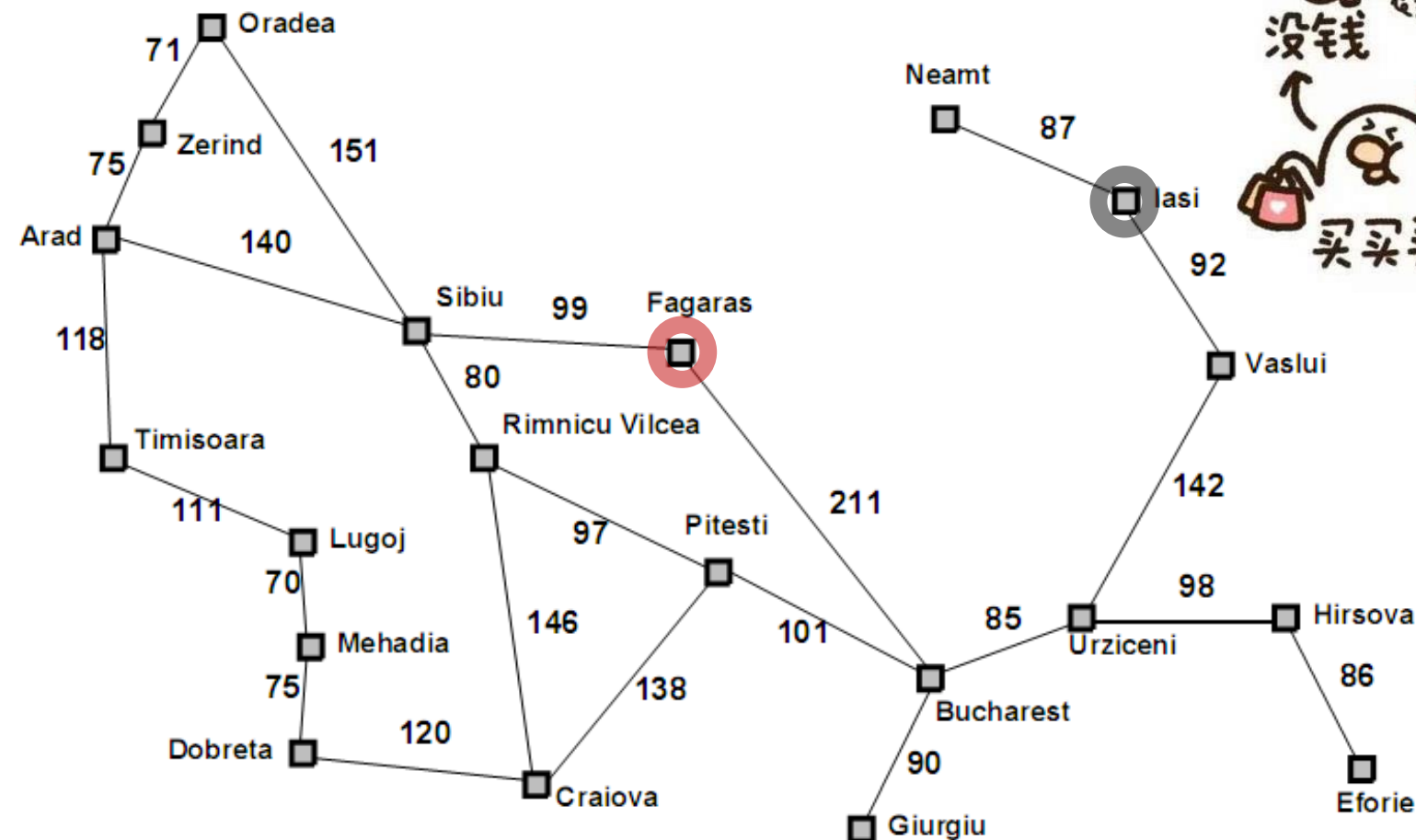
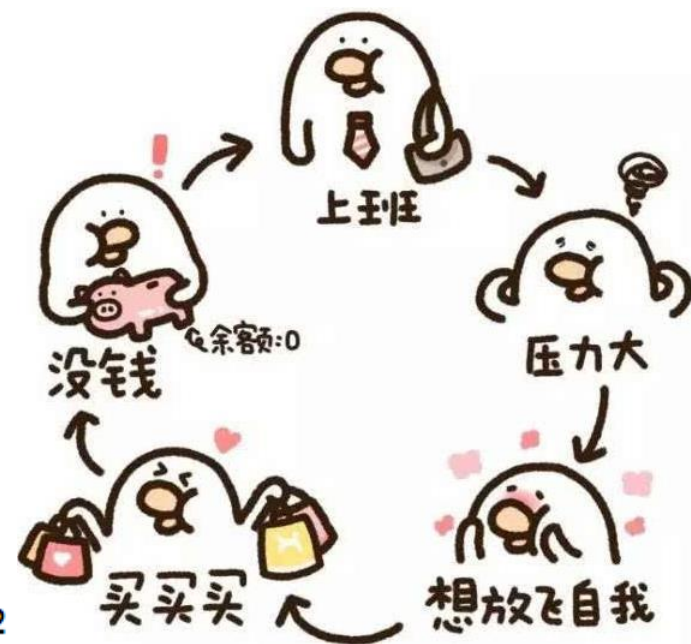
- 贪婪搜索有可能引导算法获坏的结果
- 最坏情况: 退化为DFS

### 3. 有信息搜索

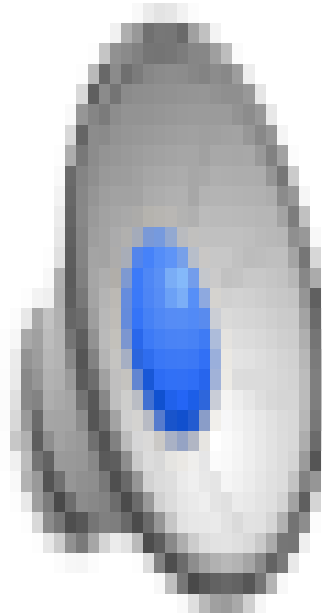


# 3. 有信息搜索

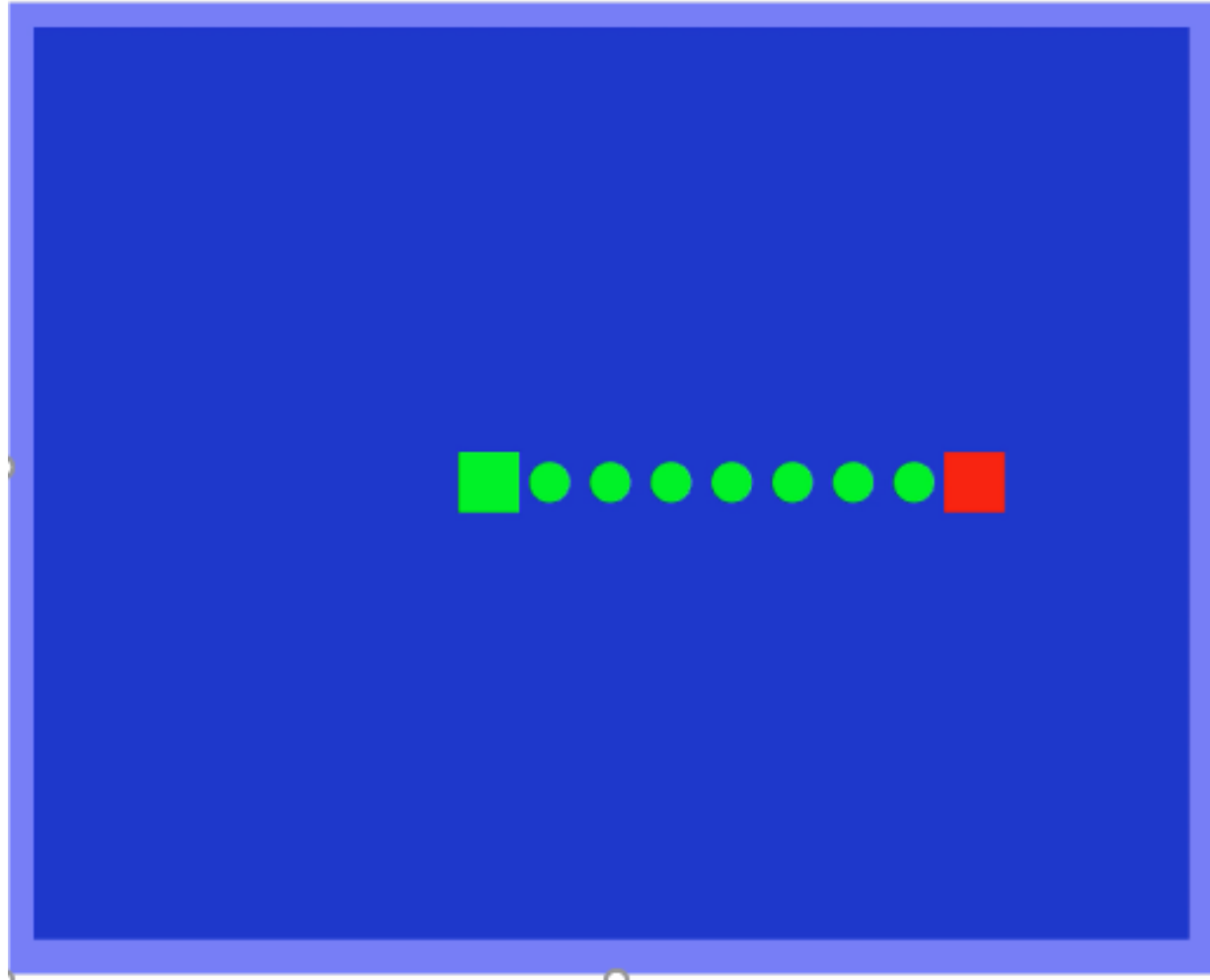
## 从Lasi到Fagaras



# 3. 有信息搜索

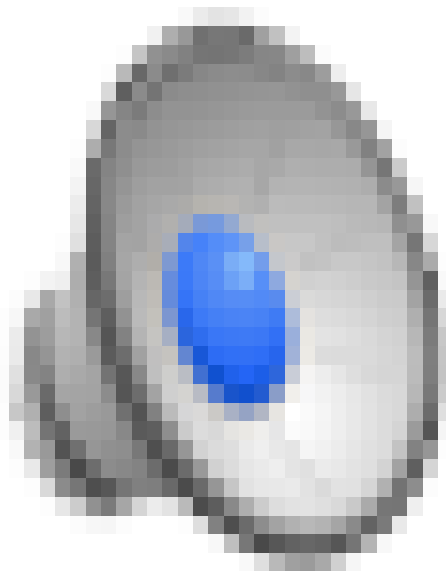


### 3. 有信息搜索

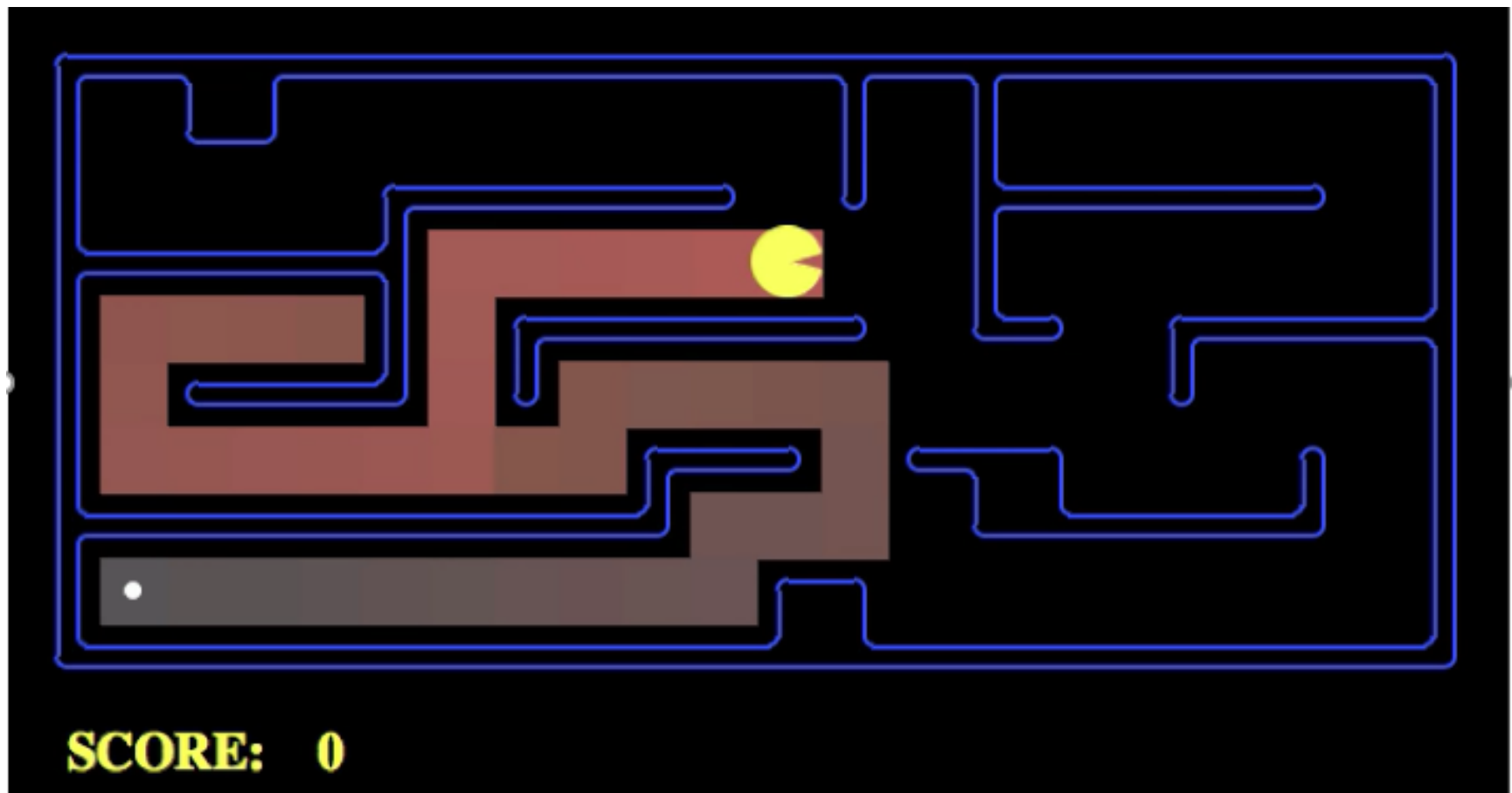




### 3. 有信息搜索



### 3. 有信息搜索



### 3. 有信息搜索

#### 贪婪算法性能分析

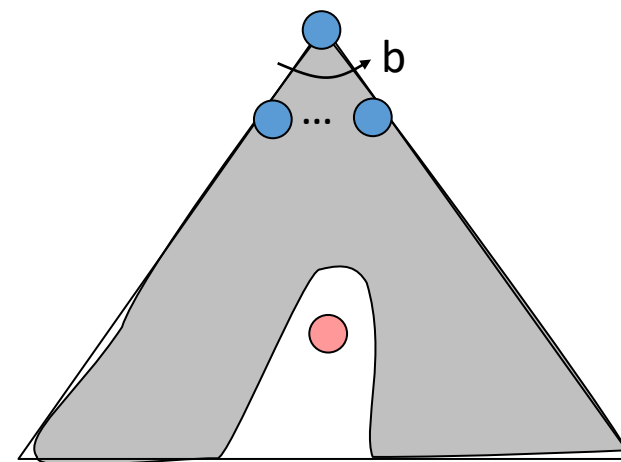
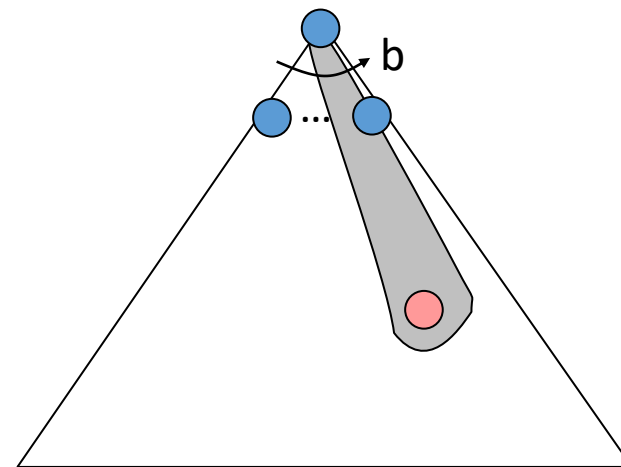
完备性?  $\times$

最优解?  $\times$  局部

时间复杂度: 最坏情况  $O(b^m)$

空间复杂度:  $O(b^m)$  - 需存储所有结点的信息

其中  $b$  代表每个结点的后继结点数,  
 $m$  代表搜索的深度



## 4. A\*搜索

搜索策略:

避免扩展耗费已经很大的路径

估价函数:

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

其中  $g(n)$  初始节点到  $n$  节点的耗费 (实际代价) ;

$h(n)$  是从  $n$  到目标节点最佳路径的估计代价。

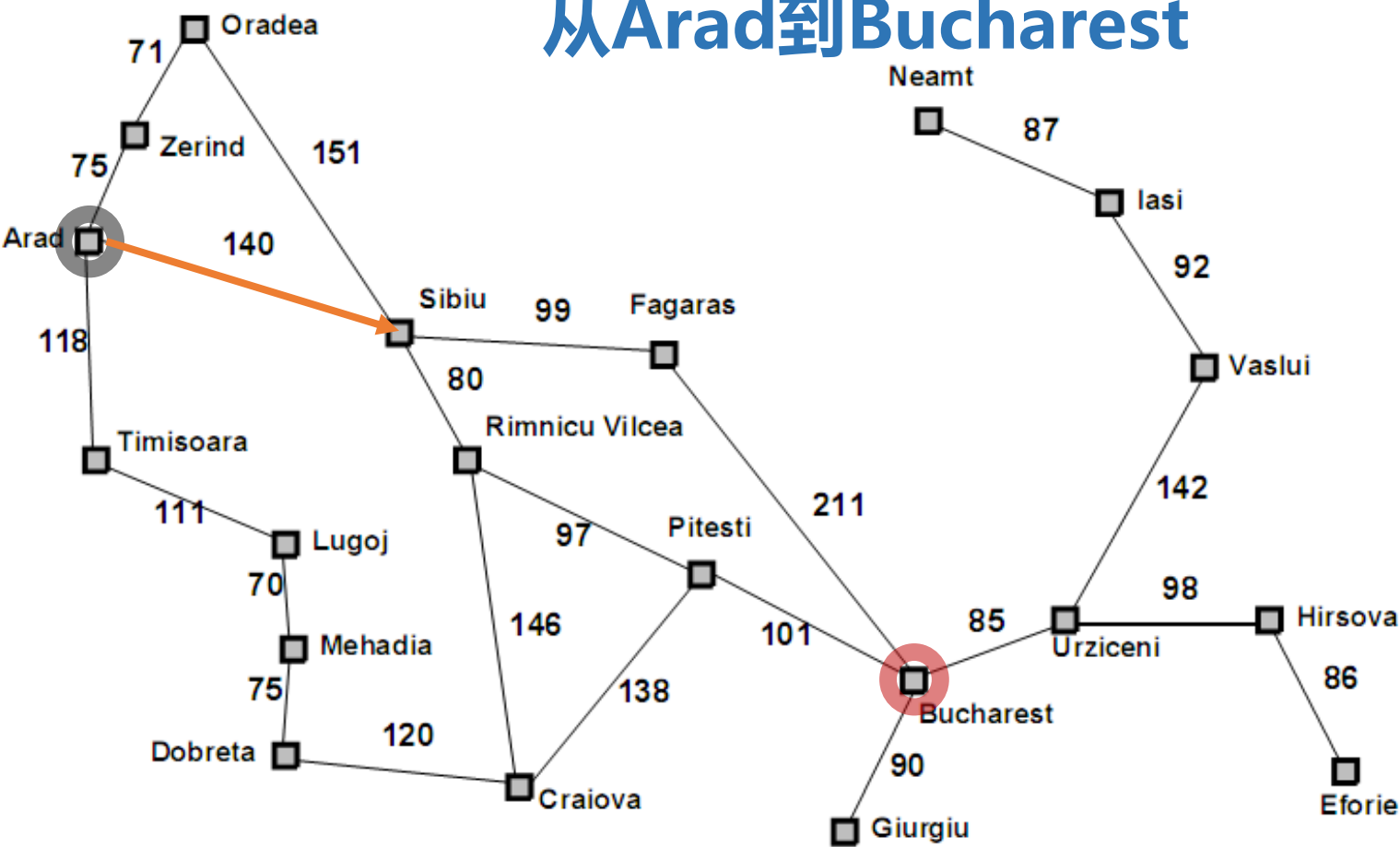


# 4. A\*搜索

从Arad到Bucharest


到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Zerind:  $75 + 374 = 449$

$g(x) + h(x)$

Arad:  $0 + 366 = 366$  

**Sibiu:  $140 + 253 = 393$**

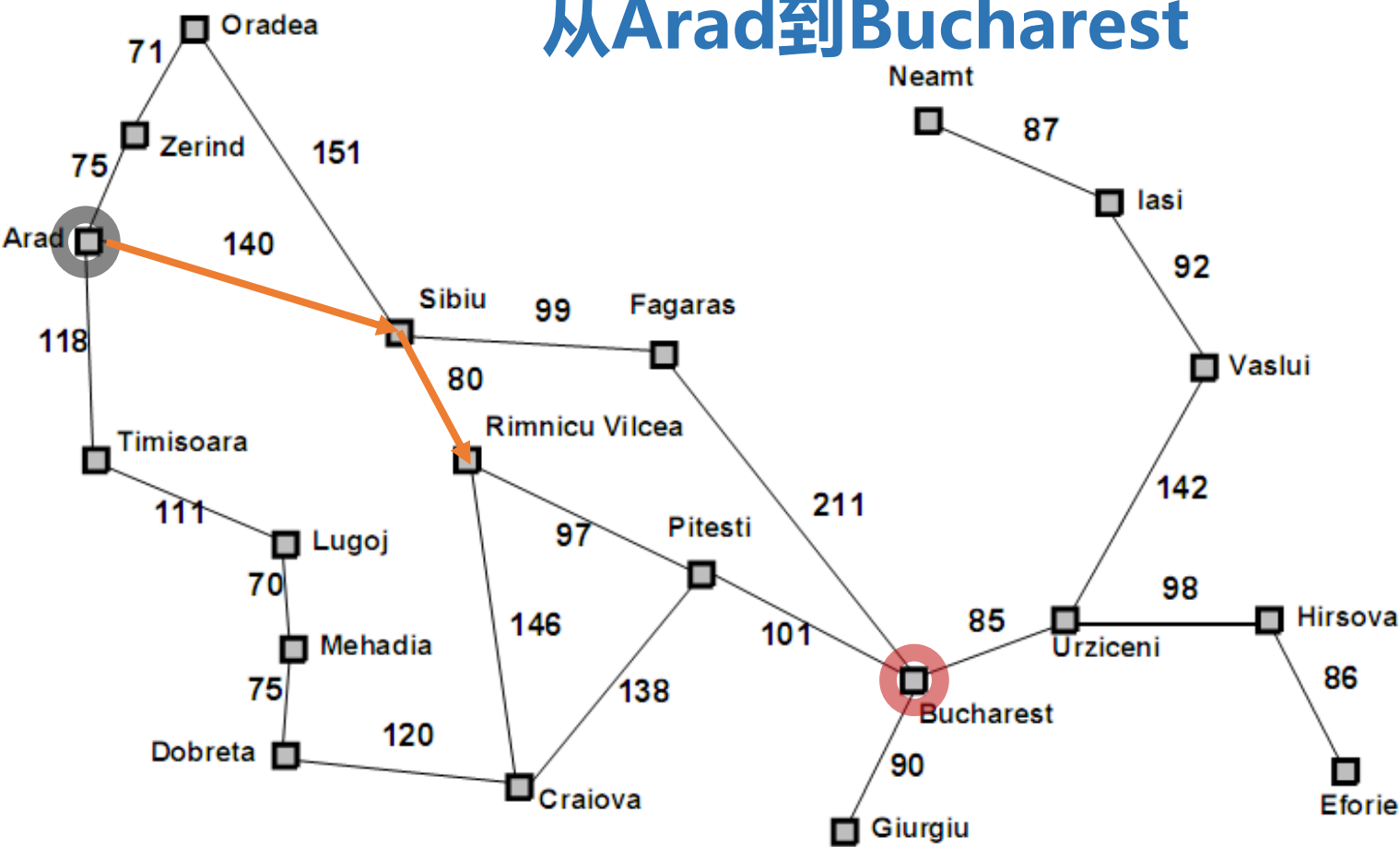
Timisoara:  $118 + 329 = 447$

# 4. A\*搜索

从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Sibiu:  $140 + 253 = 393$  →

Arad:  $280 + 366 = 646$

Oradea:  $291 + 380 = 671$   $g(x) + h(x)$

Fagaras:  $239 + 176 = 415$

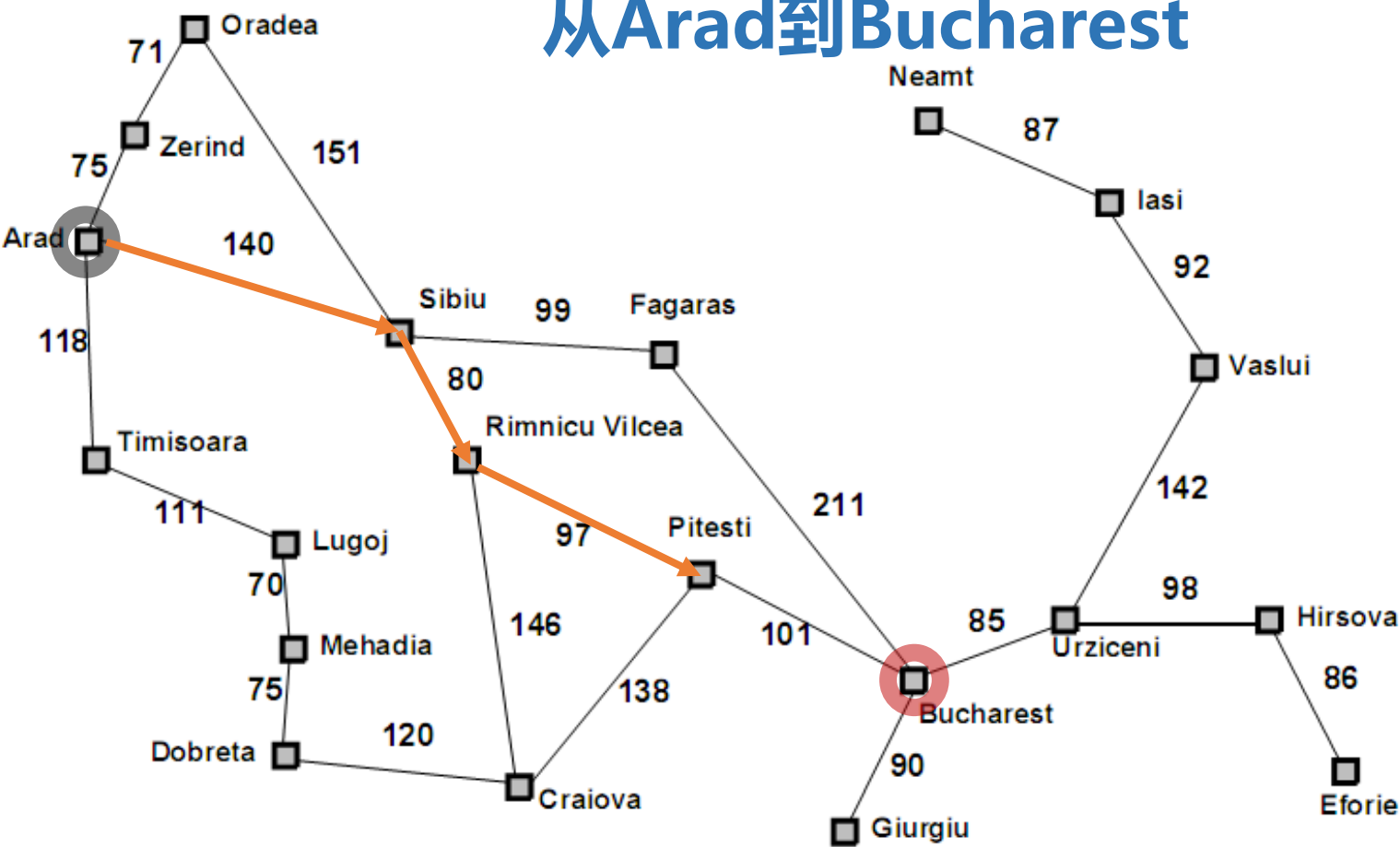
Rimnicu Vilcea:  $220 + 193 = 413$

# 4. A\*搜索

从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Sibiu:  $300 + 253 = 553$

$g(x) + h(x)$

Rimnicu Vilcea:  $220 + 193 = 413$  →

Pitesti:  $317 + 98 = 415$

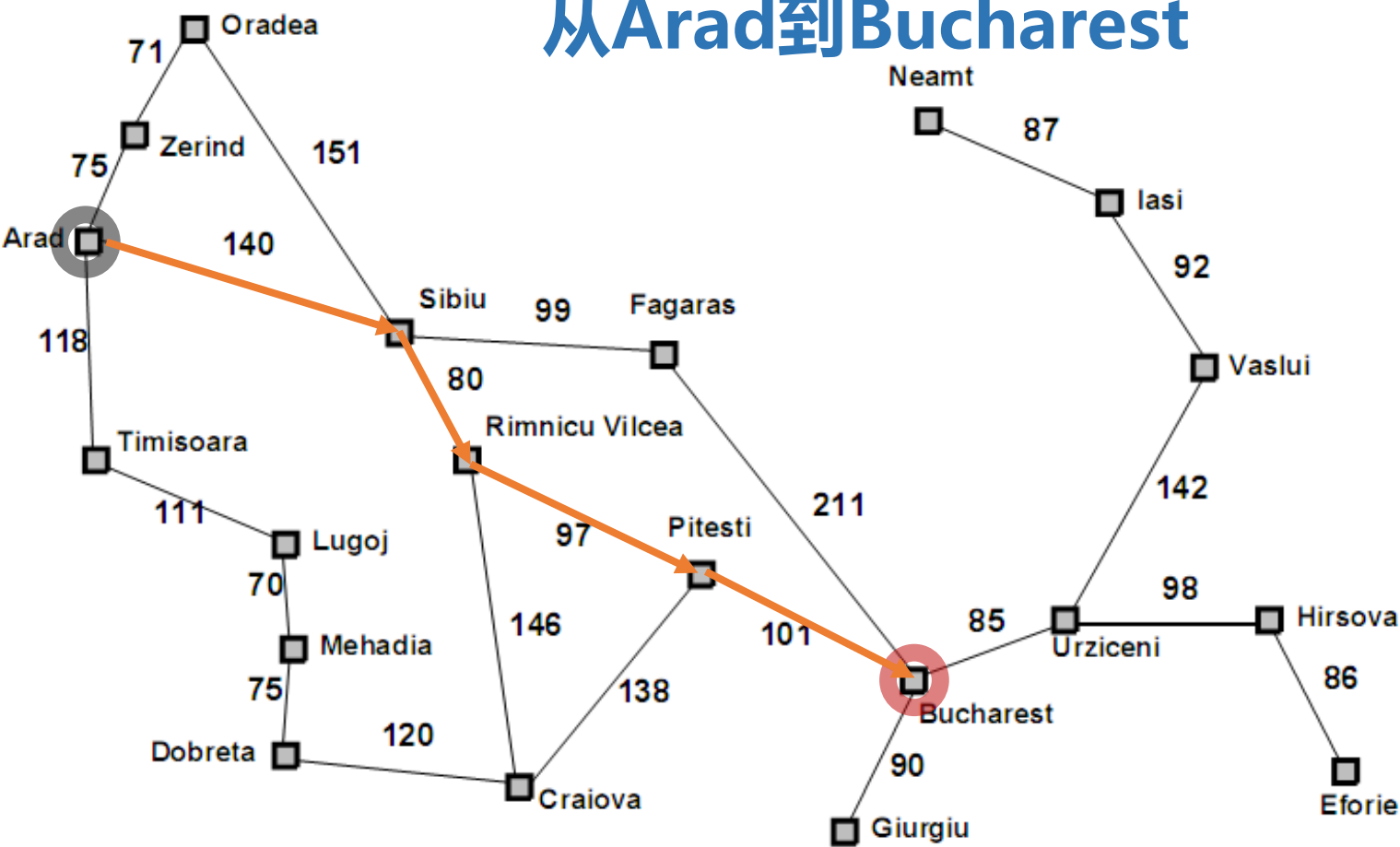
Craiova:  $366 + 160 = 526$

# 4. A\*搜索

从Arad到Bucharest

到目的地直接距离

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Rimnicu Vilcea:  $414 + 193 = 607$

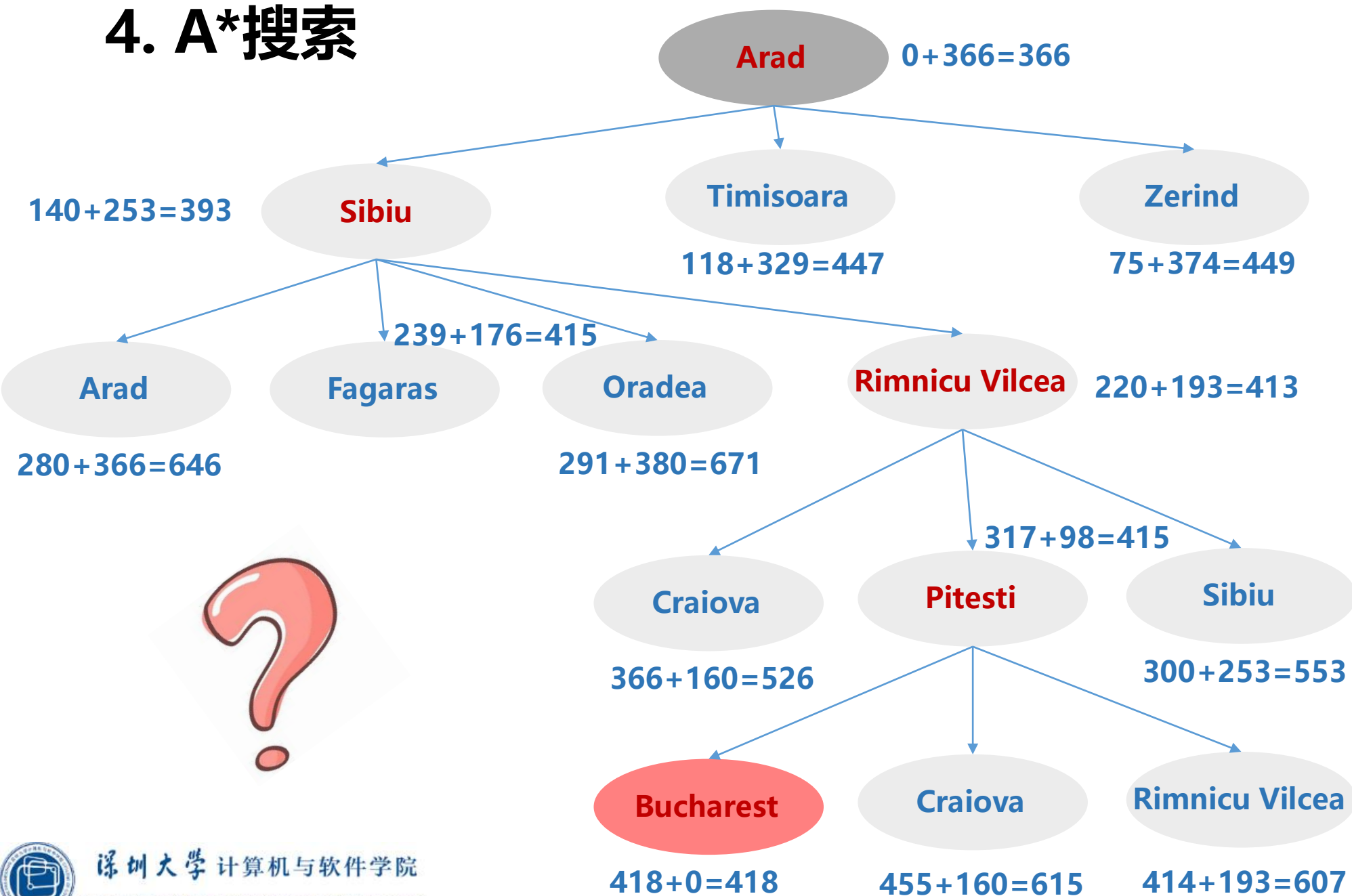
Pitesti:  $317 + 98 = 415$

**Bucharest:  $418 + 0 = 418$**   $g(x) + h(x)$

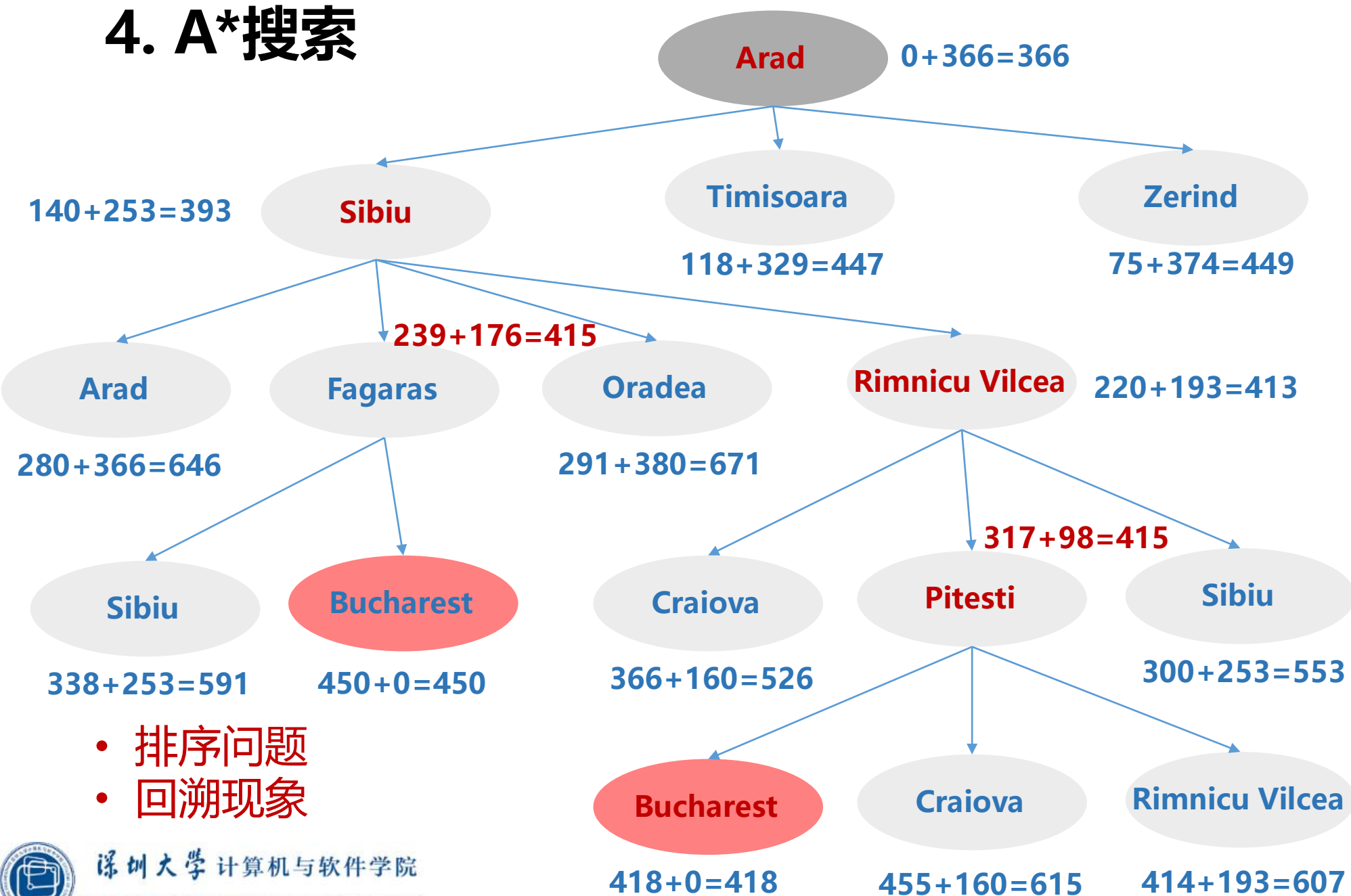
Craiova:  $455 + 160 = 615$



## 4. A\*搜索



# 4. A\*搜索

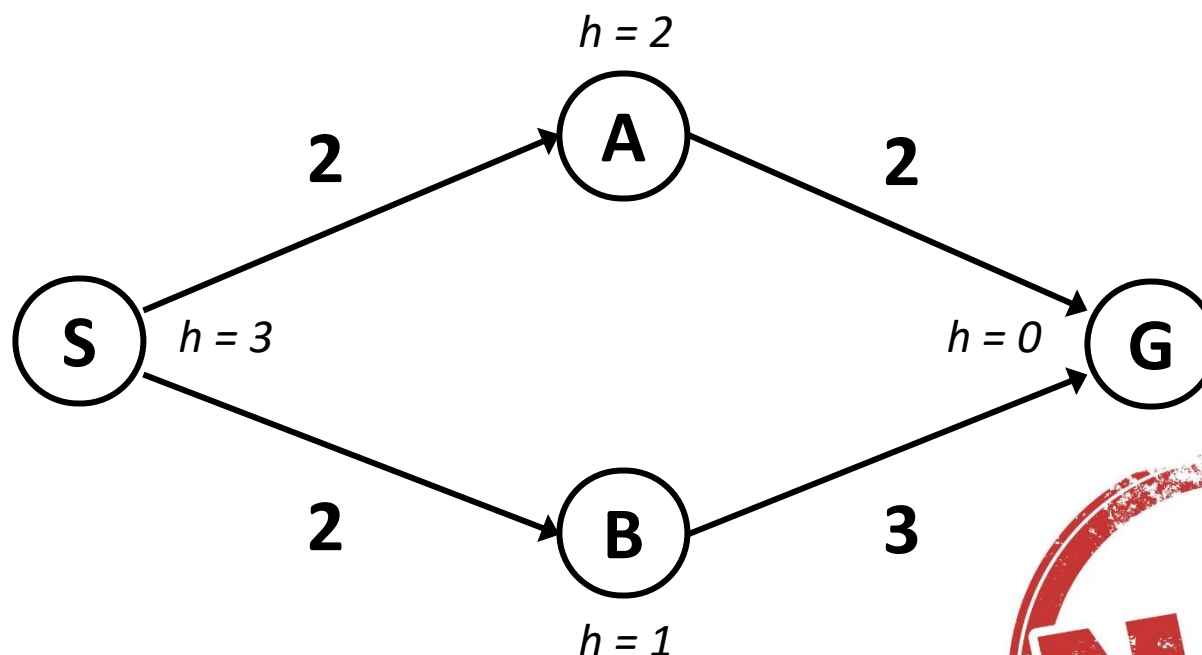


- 排序问题
- 回溯现象

## 4. A\*搜索

### A\*算法停止条件

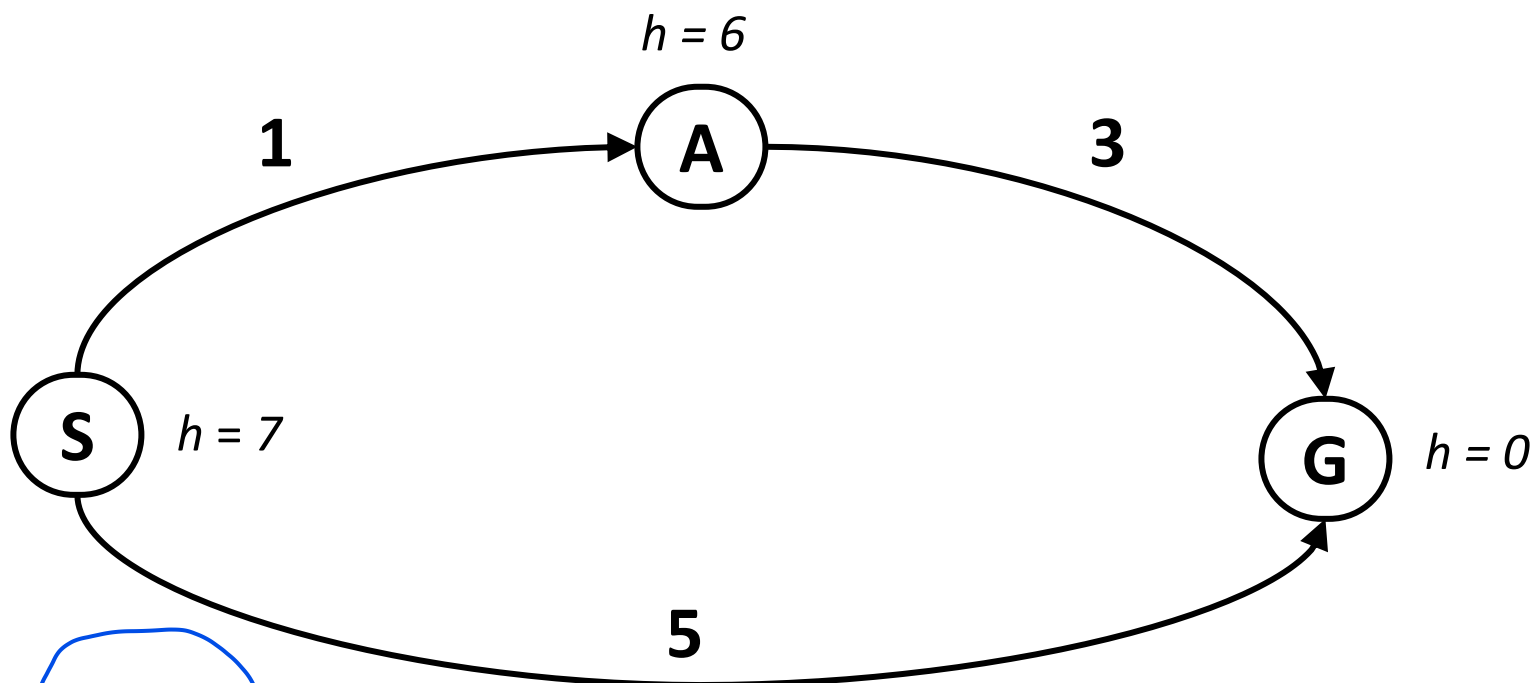
能否在探索到目标后就停止？



**NO!**

## 4. A\*搜索

### A\*算法最优性



- 错过最优解
- 估计目标代价过大
- 估计目标代价应小于实际目标代价

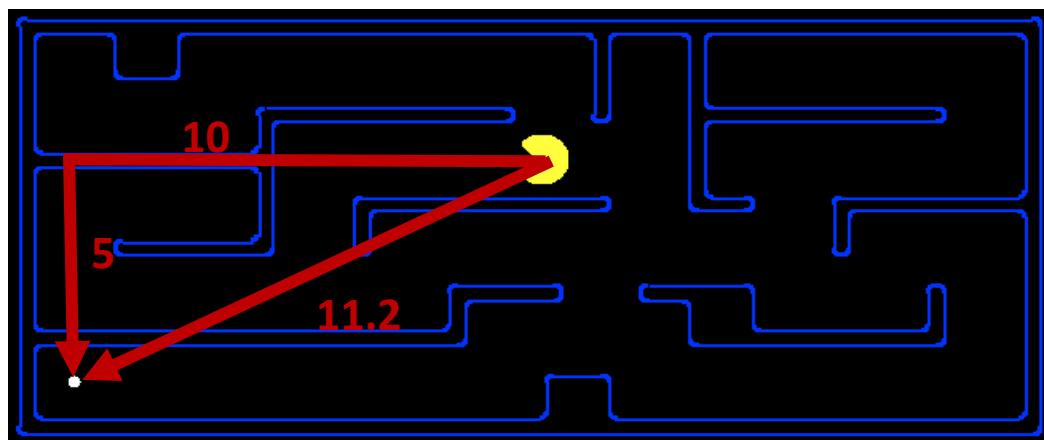
**启发式函数是关键**

## 4. A\*搜索

### 可纳性 (准确性)

- 如果对任意节点 $n$ ,  $h(n) \leq h^*(n)$ 成立, 则启发式函数是可纳的(**admissible**), 其中 $h^*(n)$ 是结点 $n$ 到目标状态的真实代价。
- 可纳启发式函数不会过高估计结点到目标的代价, 所以算法的解是最优的。

**定理:** 如果 $h(n)$ 可纳, 则A\*树搜索算法是最优的



2	8	3
1	6	4
7		5

→

1	2	3
8		4
7	6	5

4

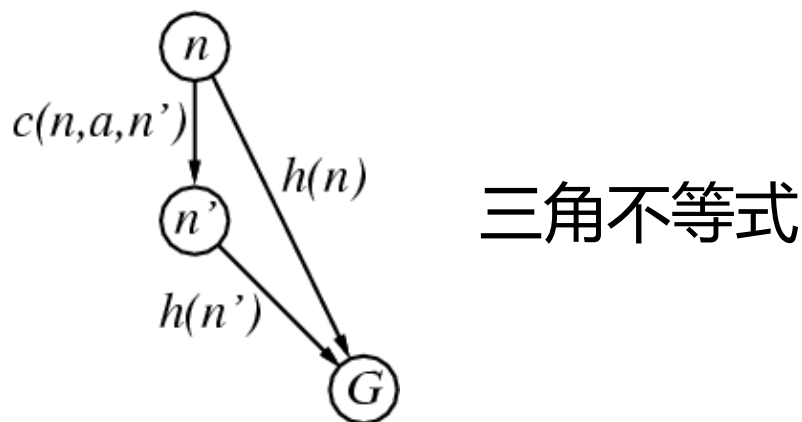


## 4. A\*搜索

### 一致性（单调性） - 仅限于图搜索

- 如果对任意节点 $n$ 和通过操作 $a$ 生成的后续结点 $n'$ ,  
 $h(n) \leq \text{cost}(n, a, n') + h(n')$ 成立, 则启发式函数是一致的。
- 如果启发式函数 $h$ 是一致的, 则评估函数 $f(n)$ 在任意路径上是非递减的。

**定理:** 如果 $h(n)$ 一致, 则A\*图搜索算法是最优的



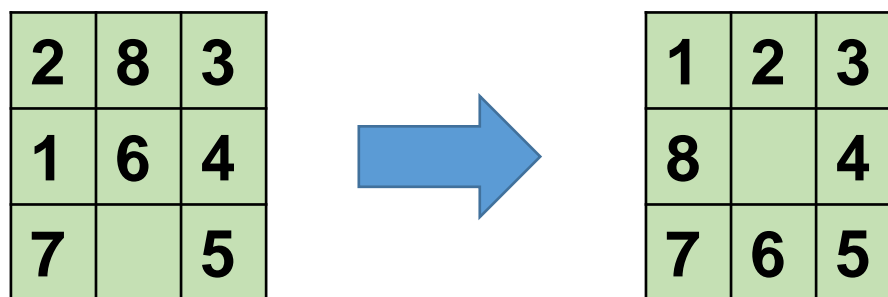
一致的启发式都是可纳的, 一致性的要求比可纳性更严格。

## 4. A\*算法

### 优势性

$$\text{可纳 } h(n) \leq h^*(n)$$

- 对于任意结点 $n$ ，可纳启发式函数 $h_1(n) < h_2(n)$ ，称 $h_2$  优于  $h_1$
- A\*搜索应选择具优势的启发式函数。



$h_1(n)$ 与目标的距离

$h_2(n)$ 不在位棋子数



# 内容回顾

## 贪婪搜索

### 搜索策略:

扩展看起来离目标**最近**的结点

### 估价函数:

$f(n) = h(n)$  (启发式: 估计结点 $n$ 到目标的代价)





# 内容回顾

## A\*搜索策略:

避免扩展耗费已经很大的路径

## 估价函数:

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

其中 $g(n)$  初始节点到 $n$ 节点的耗费（实际代价）；

$h(n)$ 是从 $n$ 到目标节点最佳路径的估计代价。



可纳性

一致性

优势性

## 4. A\*算法

### 启发式函数设计

#### 问题松弛化:

八数码问题的动作描述:

**棋子可以从方格A移动到方格B，如果A与B水平或竖直相邻，而且B为空**

去掉相关条件，形成松弛问题

- 1) A与B相邻，棋子可从A移到B（距离）
- 2) 如果B空，棋子可从A移动B
- 3) 棋子可从A移动到B（不在位）

解路径26步

7	2	4
5		6
8	3	1



	1	2
3	4	5
6	7	8

Absolver自动生成启发式信息

Armand E. Prieditis. Machine Discovery of Effective Admissible Heuristics[J]. Machine Learning, 1993.

## 4. A\*算法

### 启发式函数设计

#### 子问题分解：

原问题分解成多个子问题，考虑子问题的解代价。

#### 模式数据库

存储子问题模式解代价

从目标反向搜索计算模式代价

针对子问题，查找模式数据库

较曼哈顿距离准确

*	2	4
*		*
*	3	1



	1	2
3	4	*
*	*	*

## 4. A\*算法

### 启发式函数设计

#### 从经验中学习:

求解大量的八数码问题，每个最优解供启发式函数学习。

学习算法可预测后续状态的解代价。

给定状态用“特征”进行描述（如 $x_1$ 不在位的棋子数， $x_2$ 现在相邻但在目标状态不相邻的棋子对数），选取100个八数码问题，统计实际解代价。

建立函数 $h(n) = c_1x_1(n) + c_2x_2(n)$

*	2	4
*		*
*	3	1

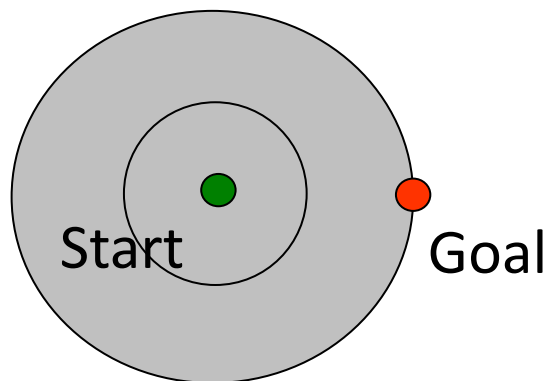
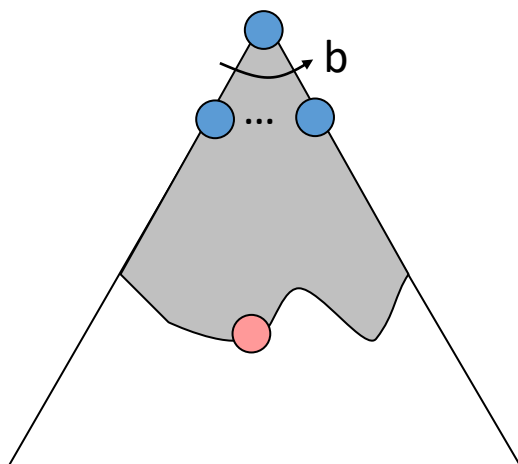


	1	2
3	4	*
*	*	*

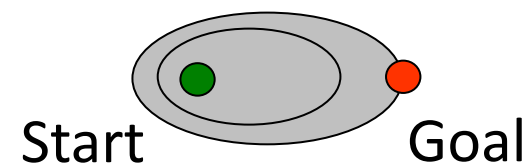
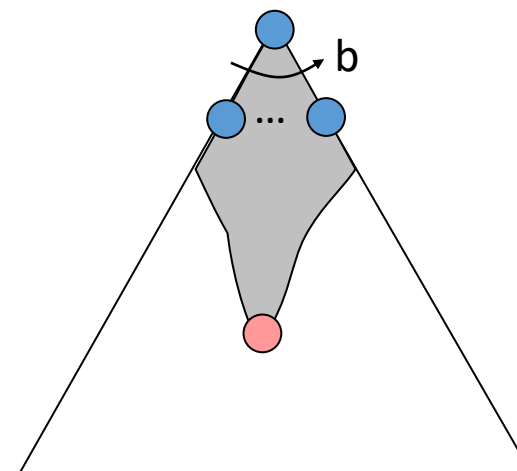
# 4. A\*算法

## A\*算法性能分析

UCS



A\*



## 4. A\*算法

### A\*算法性能分析

完备性?

最优性?

时间复杂度: 最坏情况 $O(b^{\varepsilon d})$

空间复杂度:  $O(b^m)$ — 需存储所有结点的信息

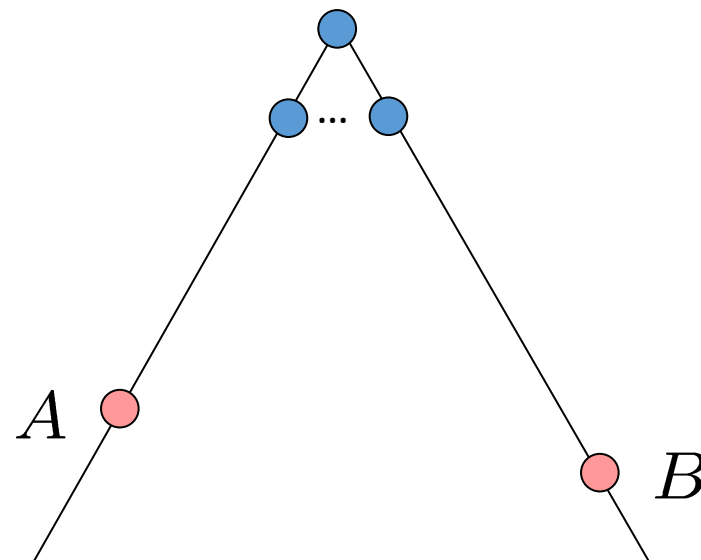
其中 $b$ 代表每个结点的后继结点数,  $d$ 代表解的深度,  
 $\varepsilon = \frac{h^* - h}{h^*}$ 是相对误差, 其中 $h^*$ 为真实代价,  $h$ 为估计  
代价。

## 4. A\*算法

### A\*算法性能分析-最优性

**命题：**假定A是最优目标结点，B是次优，h是可纳启发式函数，则A比B更早被搜索。

**证明：**？



## 4. A\*算法

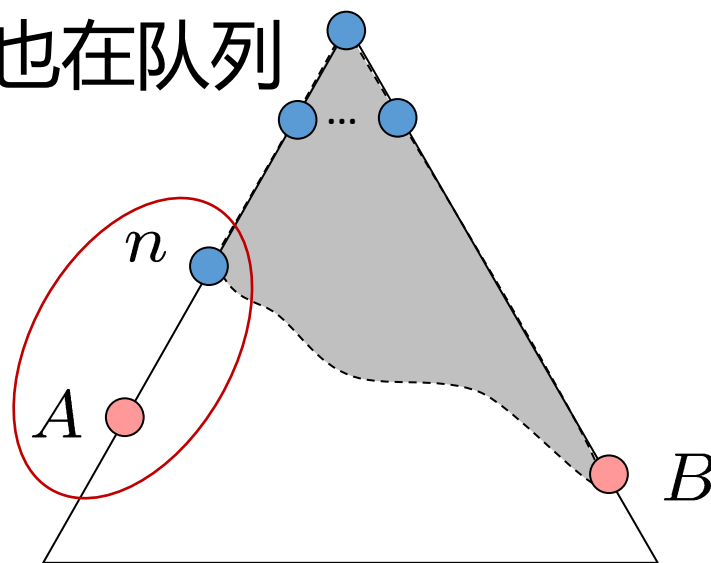
### A\*算法性能分析-最优性

**命题：**假定 $A$ 是最优目标结点， $B$ 是次优， $h$ 是可纳启发式函数，则 $A$ 比 $B$ 更早被搜索。

**证明：**

假设 $B$ 在候选队列， $A$ 的祖先结点 $n$ 也在队列  
只需证明 $n$ 比 $B$ 更早扩展

- $f(n)$ 小于等于 $f(A)$
- $f(A)$ 小于 $f(B)$



$$\begin{array}{ll} g(A) < g(B) & B \text{ 次优} \\ f(A) < f(B) & \text{目标 } h = 0 \end{array}$$



## 4. A\*算法

### A\*算法性能分析-最优性

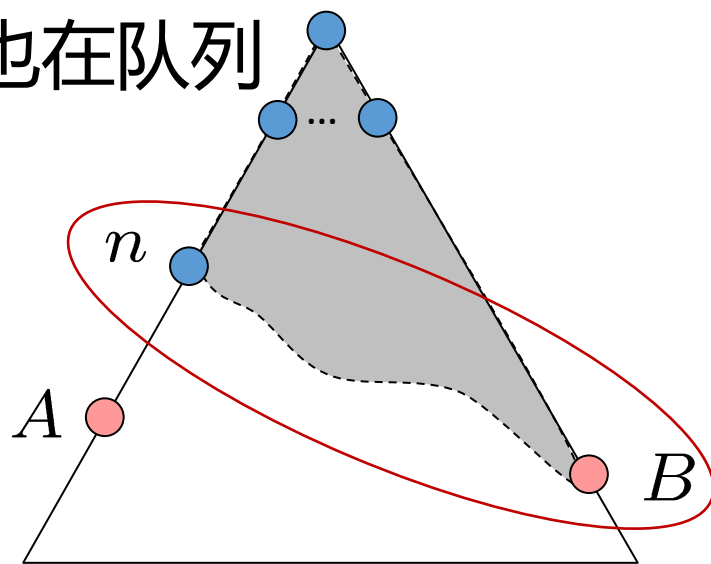
**命题：**假定 $A$ 是最优目标结点， $B$ 是次优， $h$ 是可纳启发式函数，则 $A$ 比 $B$ 更早被搜索。

**证明：**

假设 $B$ 在候选队列， $A$ 的祖先结点 $n$ 也在队列  
只需证明 $n$ 比 $B$ 更早扩展

- $f(n)$ 小于等于 $f(A)$
- $f(A)$ 小于 $f(B)$
- $n$ 比 $B$ 先扩展
- $A$ 比 $B$ 先扩展

命题得证

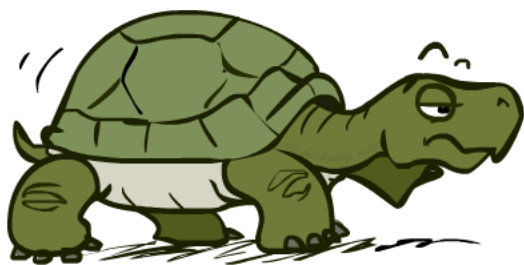


$$f(n) \leq f(A) < f(B)$$

## 4. A\*算法

### 总结:

- A\*算法综合了后向路径代价和前向估计代价;
- 如果启发式函数可纳/一致, A\*是最优的;
- 启发式函数设计是关键, 应选具有优势的启发式函数。



UCS



GBFS

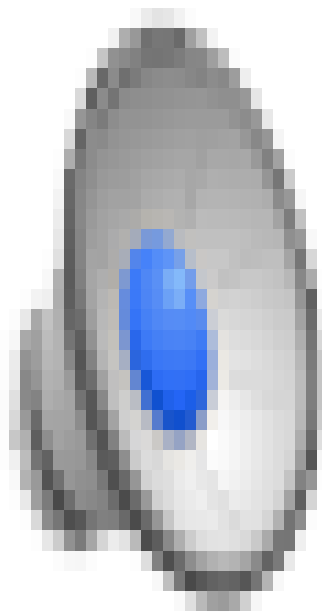


A\*

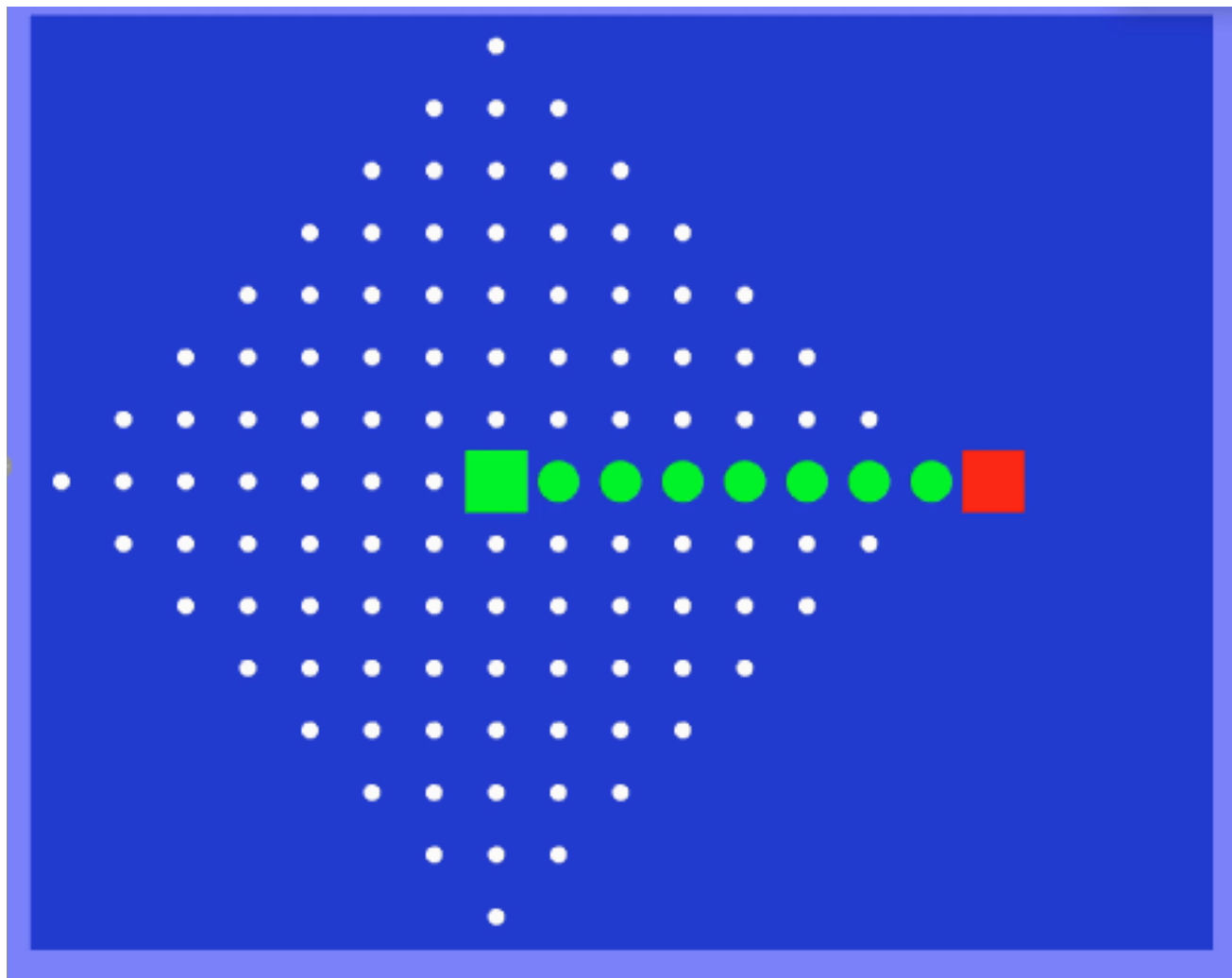
## 5. 基本搜索总结

	后向代价	前向代价	备注
BFS	$g(n)$ : 深度层次	N/A	
DFS	$g(n)$ : 深度层次	N/A	
UCS	$g(n)$ : 真实代价	N/A	<u>退化为BFS</u>
GBFS	N/A	$h(n)$ : 估计代价	<u>退化为DFS</u>
A*	$g(n)$ : 真实代价	$h^*(n)$ : 估计代价	
...			

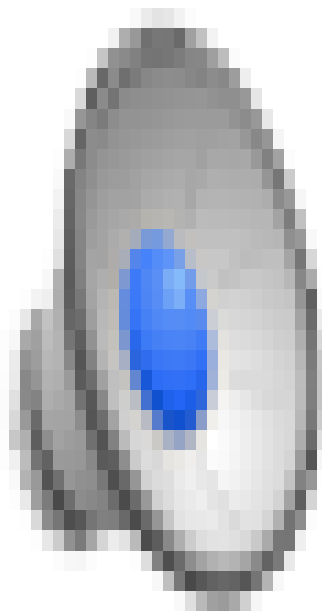
## 5. 基本搜索总结



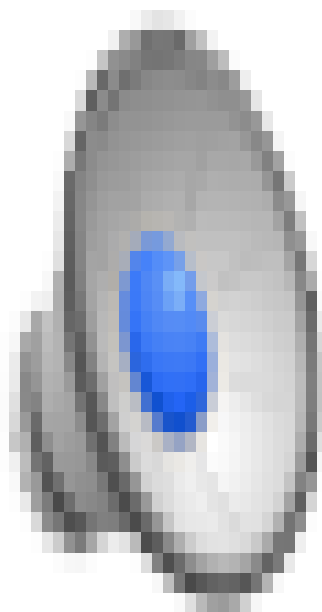
## 5. 基本搜索总结



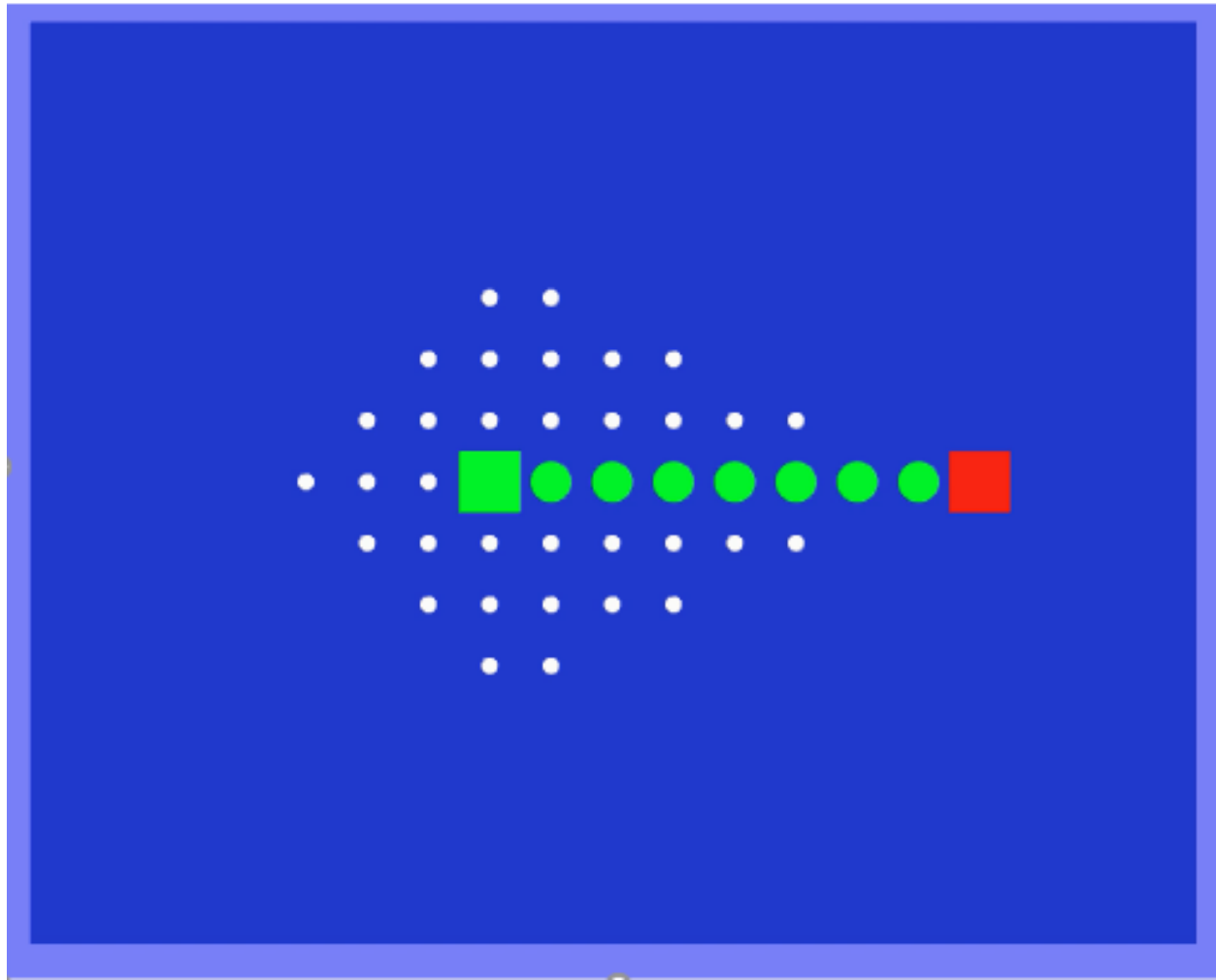
## 5. 基本搜索总结



# 5. 基本搜索总结

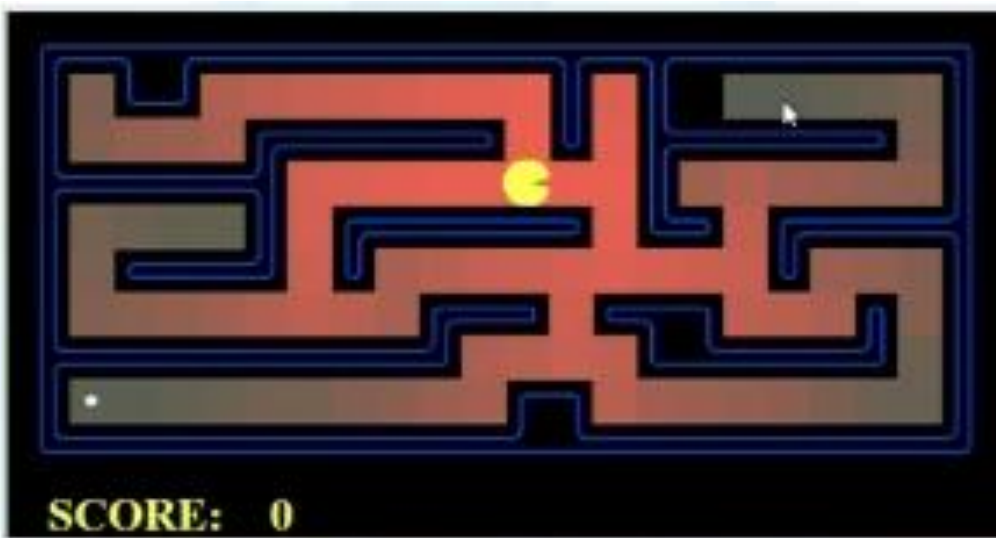


## 5. 基本搜索总结





## 5. 基本搜索总结



## 6. 练习题

### 修道士(Missionaries)和野人(Cannibals)问题(简称M-C问题)。

设在河的一岸有三个野人、三个修道士和一条船，修道士想用这条船把所有的人运到河对岸，但受以下条件的约束：

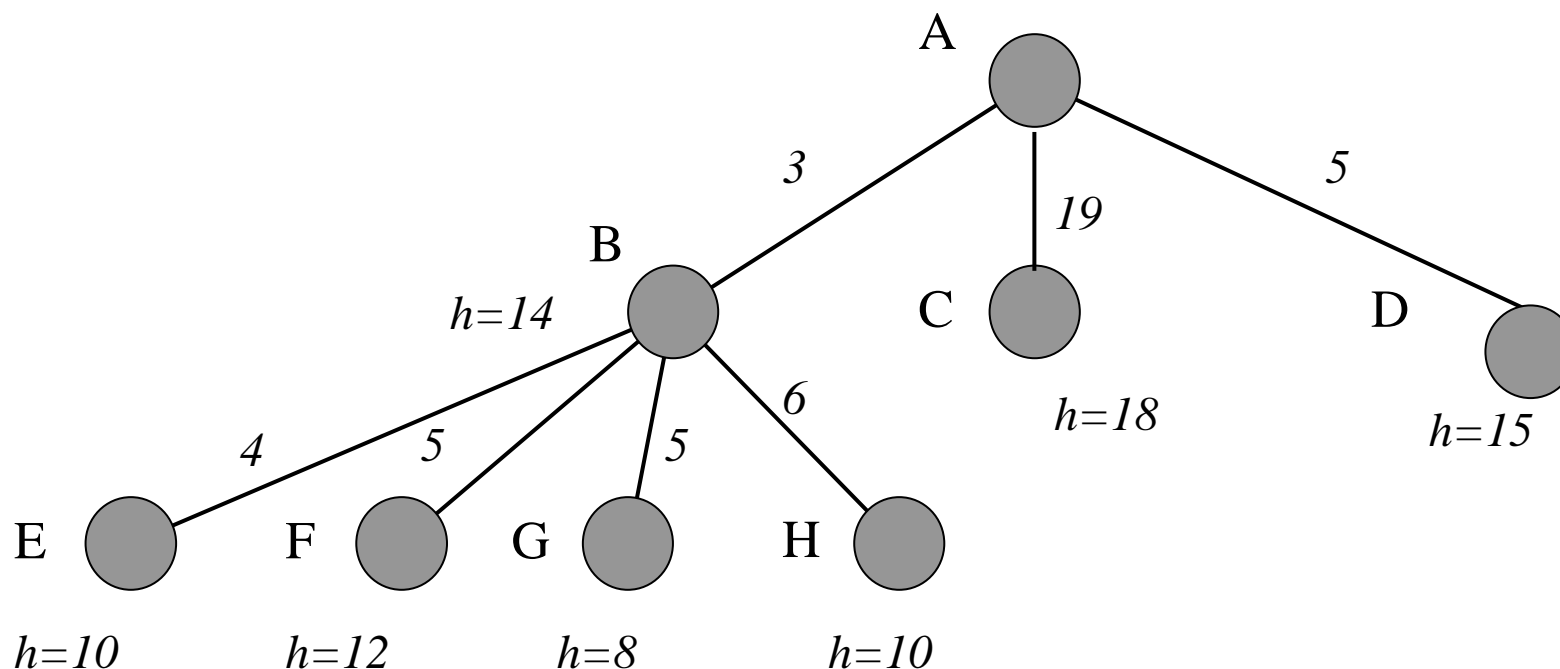
- 一是修道士和野人都会划船，但每次船上至多可载两个人；
- 二是在河的任一岸，如果野人数目超过修道士数，修道士会被野人吃掉。

如果野人会服从任何一次过河安排，请规划一个确保修道士和野人都能过河，且没有修道士被野人吃掉的安全过河计划。



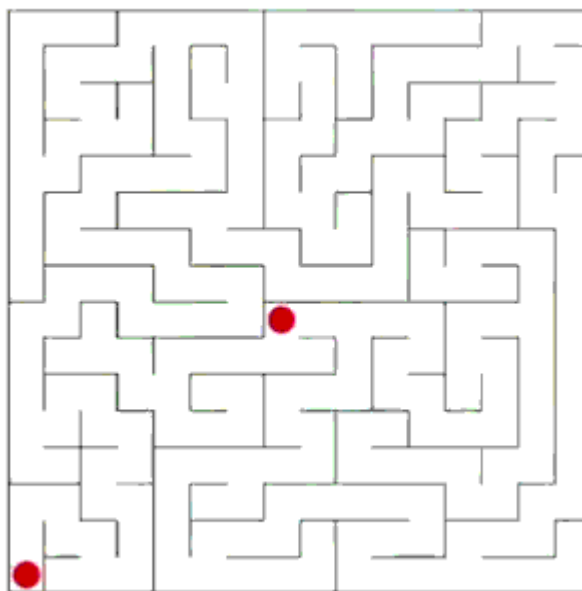
## 6. 练习题

计算宽度优先、深度优先、等代价、贪婪和A\*算法搜索过程。



## 6. 实验题

实现宽度优先和深度优先迷宫求解算法（建议Python）

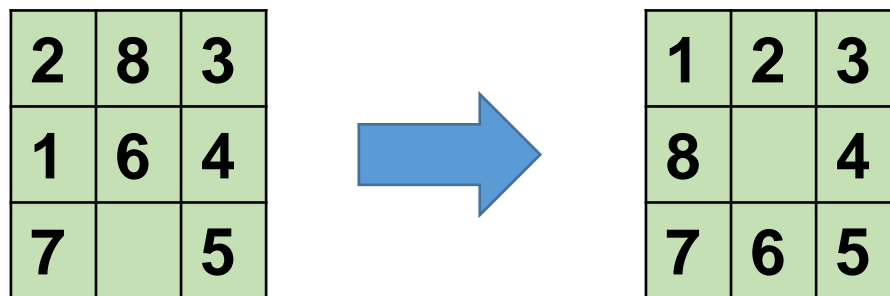


迷宫搜索

[https://blog.csdn.net/qq\\_40276310/article/details/80668401](https://blog.csdn.net/qq_40276310/article/details/80668401)

## 6. 实验题

针对以下8数码问题尝试设计两种或以上的启发式函数，并比较它们的优势性。



# 谢谢聆听 欢迎提问



深圳大学 计算机与软件学院

College of Computer Science and Software Engineering of Shenzhen University