



# 人工智能 I





# 本节课安排

## □ 无信息搜索

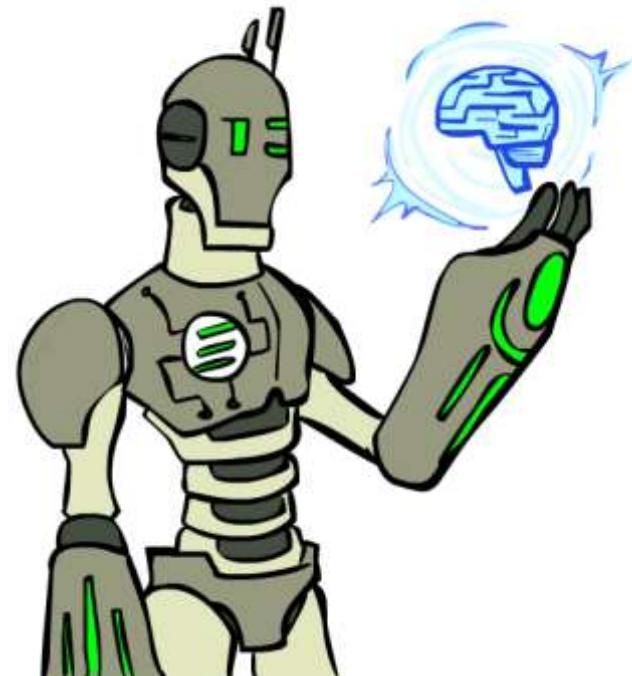
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)





# 本周四实验课安排

周次	4
日期	2025/10/16
星期	星期四
时间	19:00-21:25
机房地点	五五楼435-软实七
实验任务	无信息搜索



# 上节课回顾

## □ 无信息搜索

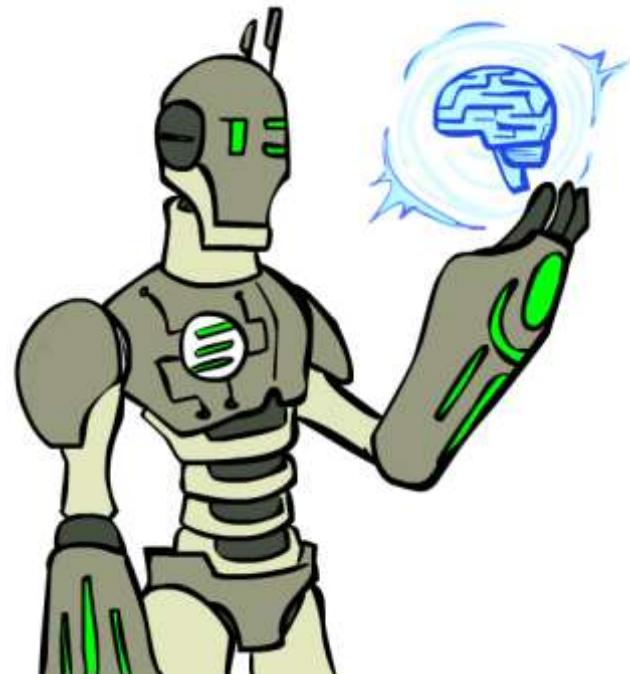
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

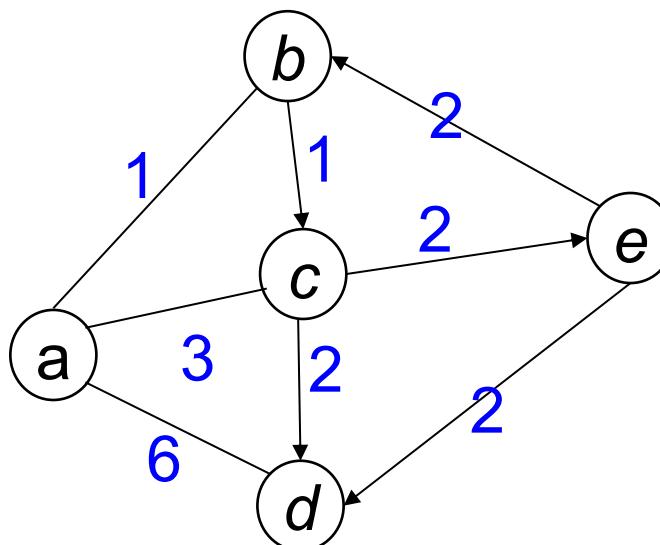
- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)



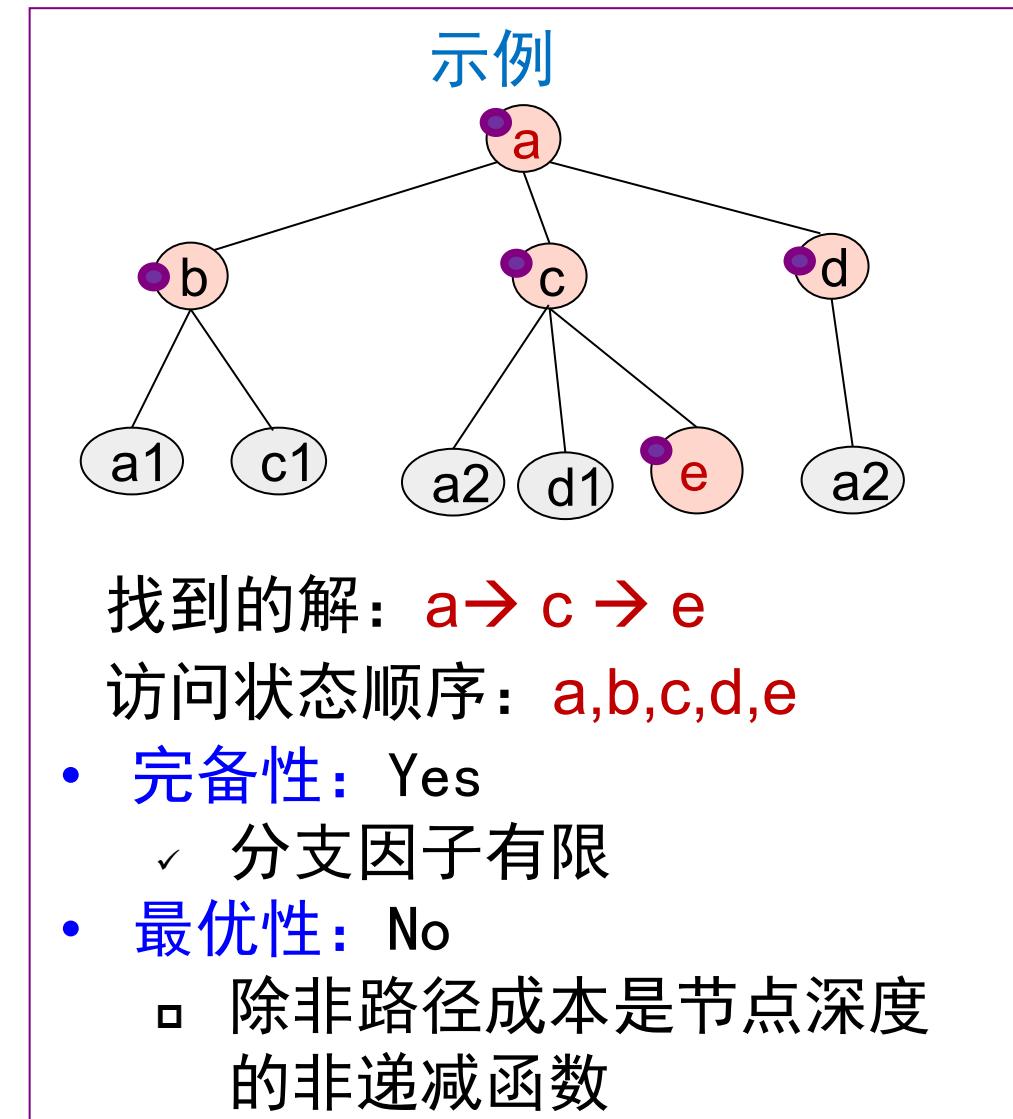


# 广度优先搜索（图搜索）

- 策略：优先探索最浅的节点
- 实现：Frontier为queue (FIFO)



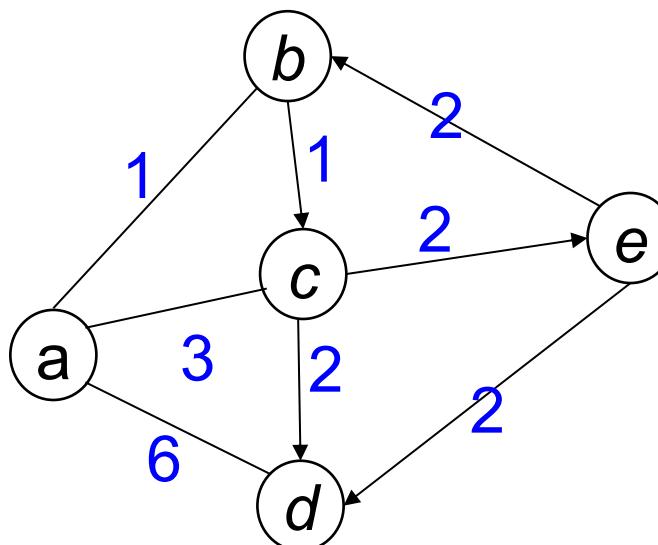
Initial state = a  
Goal state = e





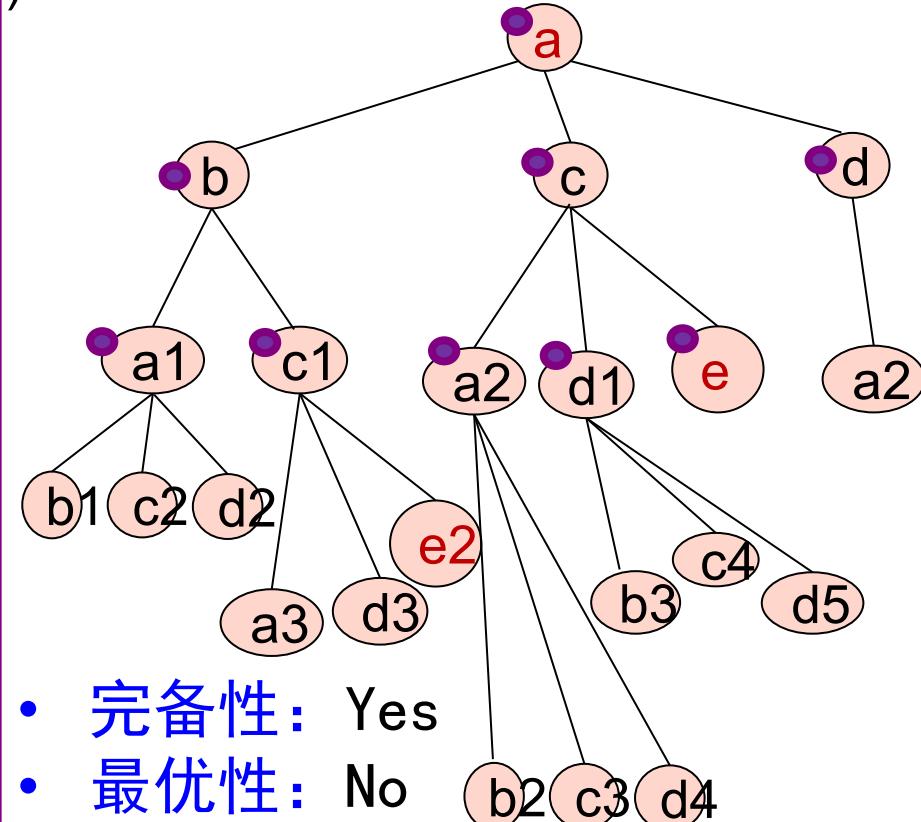
# 广度优先搜索（树搜索）

- 策略：优先探索最浅的节点
- 实现：Frontier为queue (FIFO)



Initial state = a  
Goal state = e

示例



- 完备性：Yes
- 最优性：No

找到的解： $a \rightarrow c \rightarrow e$

访问状态顺序： $a, b, c, d, a, c, a, d, e$



# 上节课回顾

## □ 无信息搜索

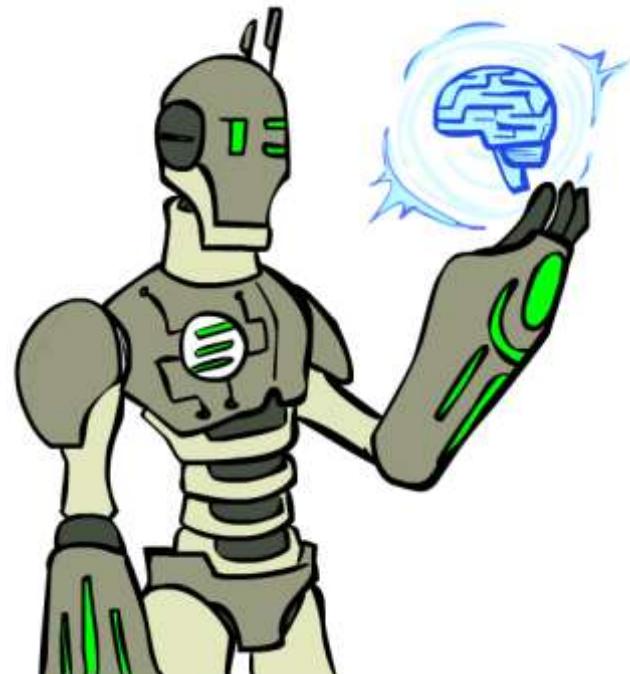
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

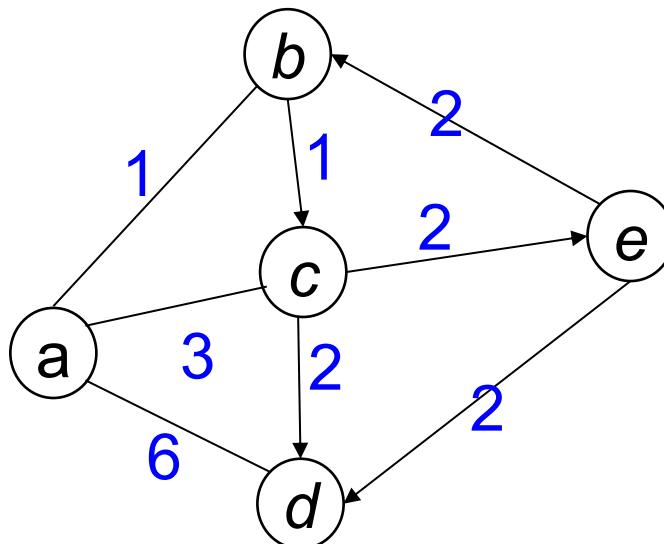
- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)



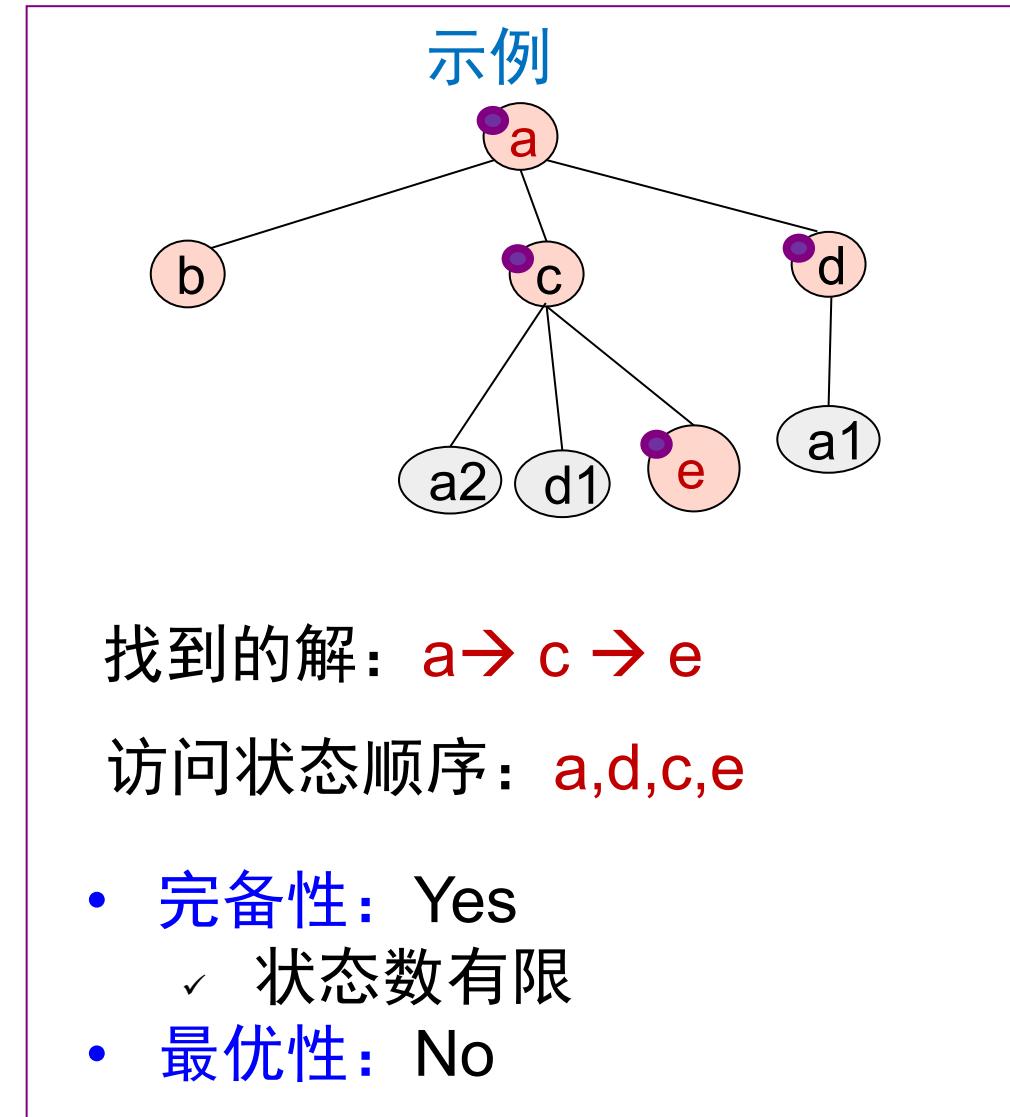


# 深度优先搜索（图搜索）

- 策略：优先探索最深的节点
- 实现：Frontier为stack (**LIFO**)



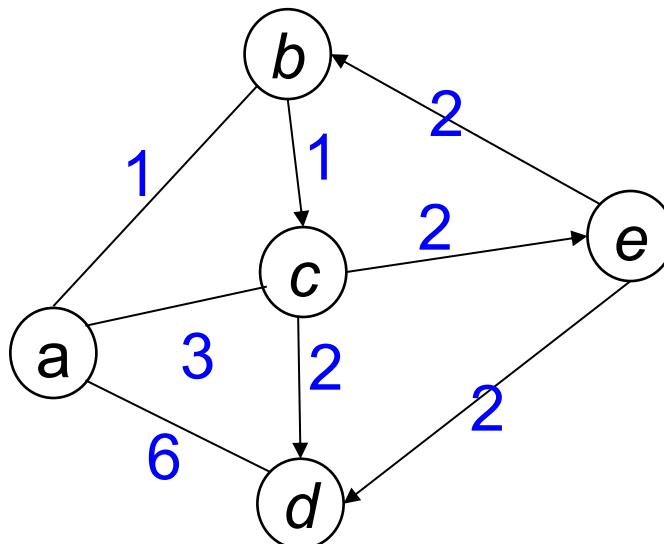
Initial state = a  
Goal state = e



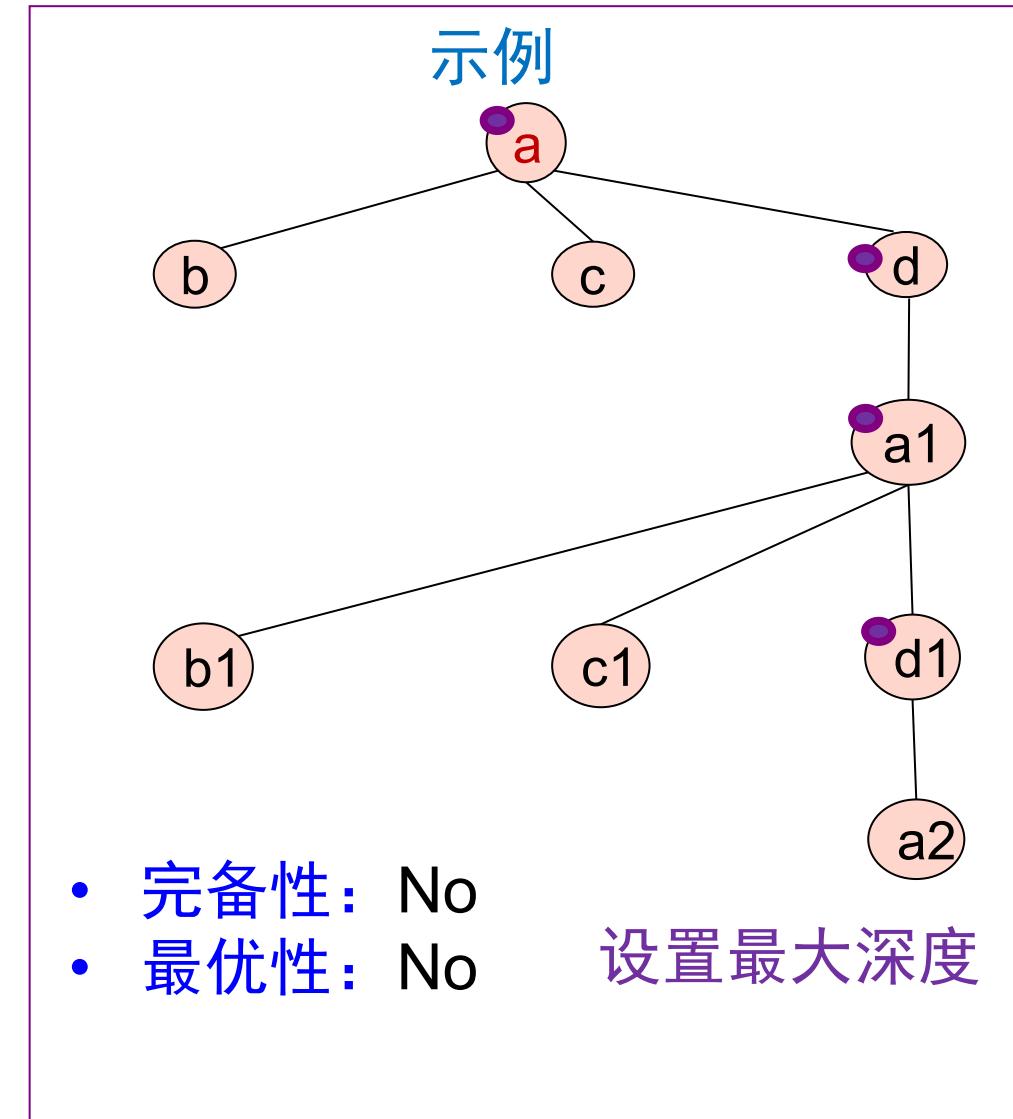


# 深度优先搜索（树搜索）

- 策略：优先探索最深的节点
- 实现：Frontier为stack (**LIFO**)



Initial state = a  
Goal state = e





# 上节课回顾

## □ 无信息搜索

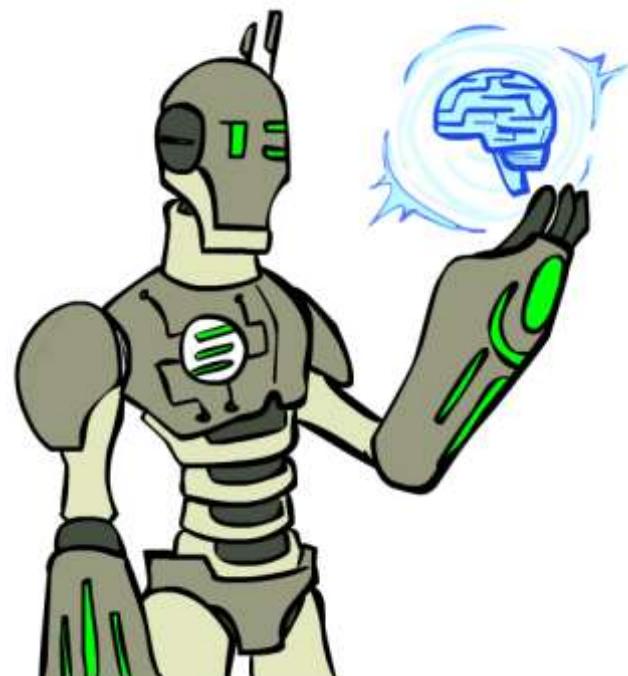
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)





# 代价一致搜索

## □ 代价一致 (Uniform-Cost )

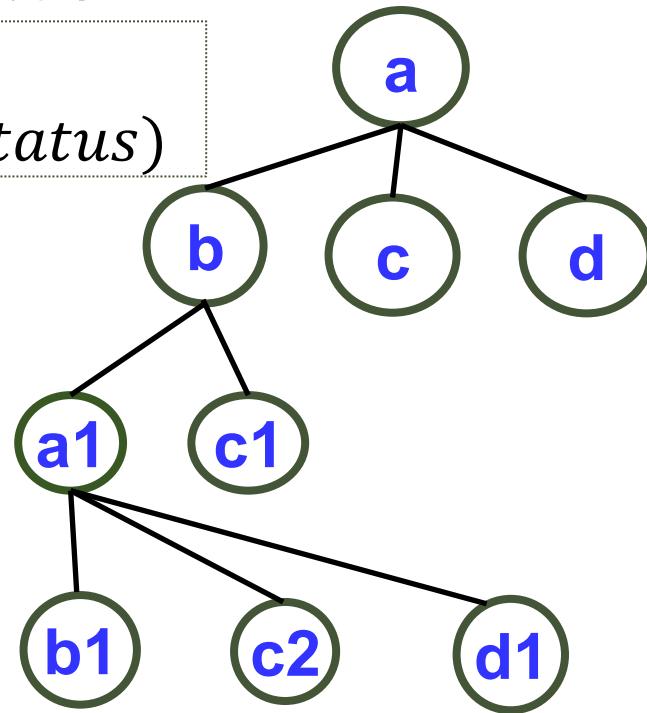
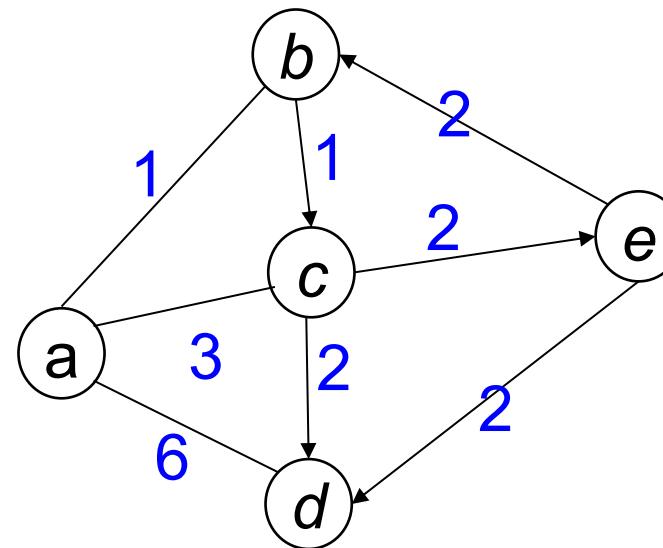
- 边缘 (Frontier) 为 priority queue
- 优先探索步骤成本最低的节点
  - 搜索树中节点 $n$ 的路径成本



$g(n)$ : 从根节点到节点 $n$ 的步骤成本

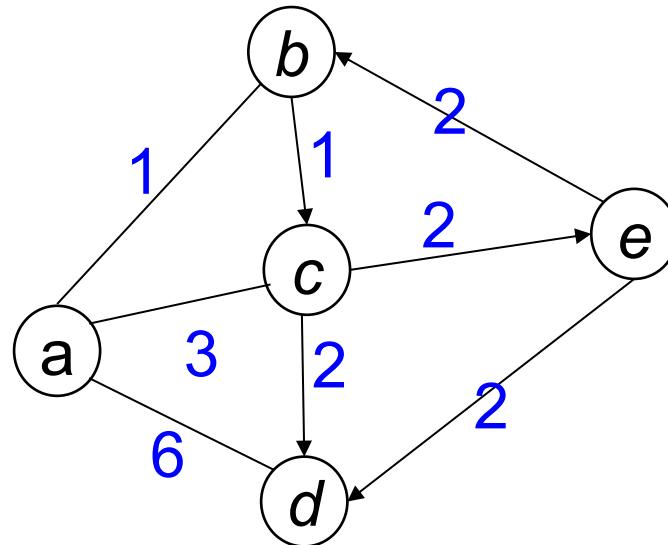
$g(a) = 0$   
 $g(b) = 1$   
 $g(c) = 3$   
 $g(d) = 6$   
 $g(a1) = 2$   
 $g(c1) = 2$   
 $g(b1) = 3$   
 $g(c2) = 5$   
 $g(d1) = 8$

$g(n) = g(n.parent)$   
 $+ c(n.parent.status \rightarrow n.status)$

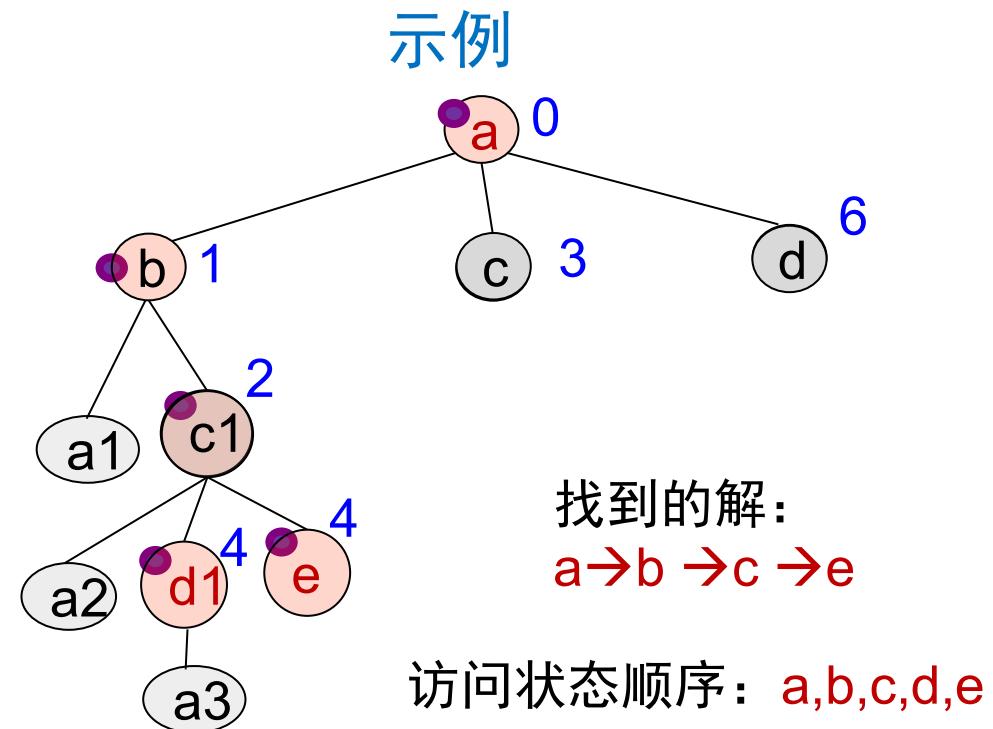




# 代价一致搜索（图搜索）



Initial state = a  
Goal state = e

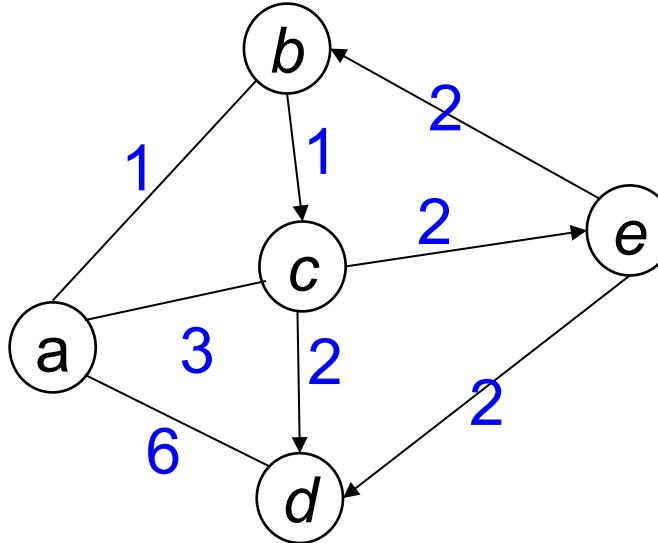


```
if child.State is not in explored or frontier then
    Insert(child, frontier);
elseif child.State is in frontier with higher path-
    cost then replace that frontier node with child;
```

- 完备性： Yes
  - 状态数有限或（分子因  
子有限且步骤代价大于0）
- 最优性： Yes



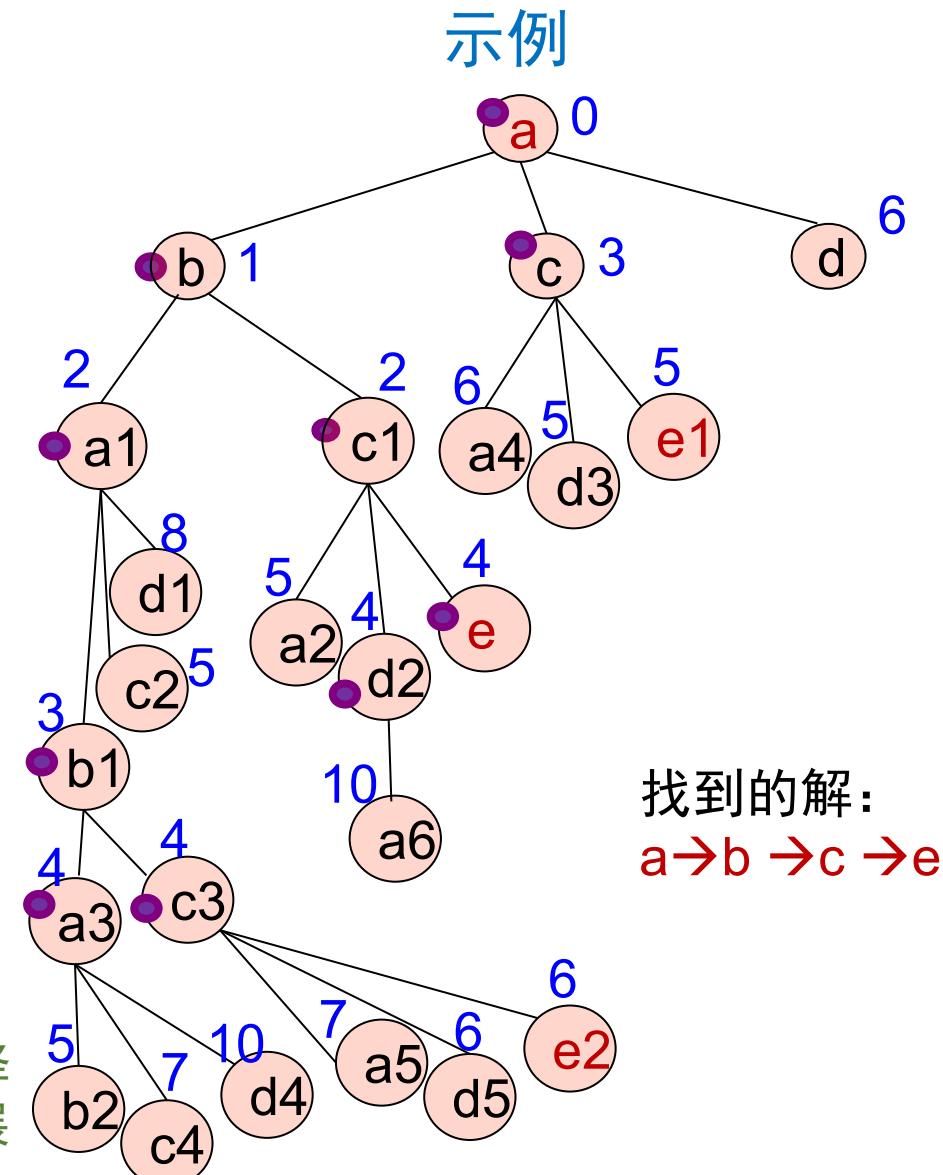
# 代价一致搜索 (树搜索)



Initial state = a  
Goal state = e

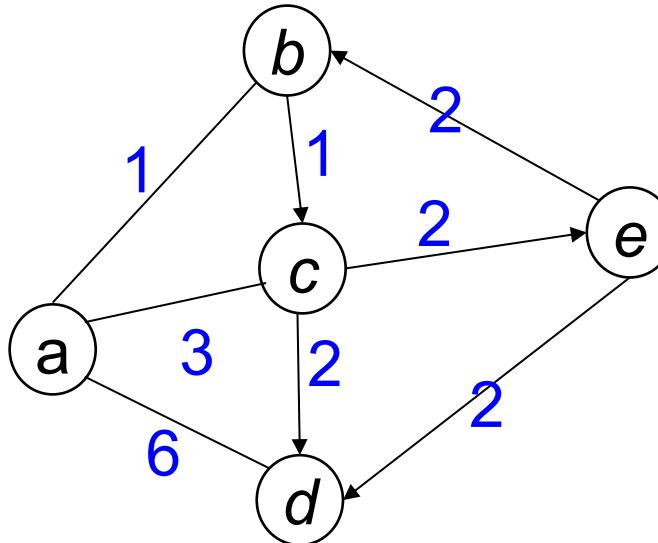
- 完备性: Yes
  - 分支因子有限且步骤成本大于0
- 最优性: Yes

当多个节点代价相等时，可以随机选择扩展的节点，或者根据某个规定来扩展



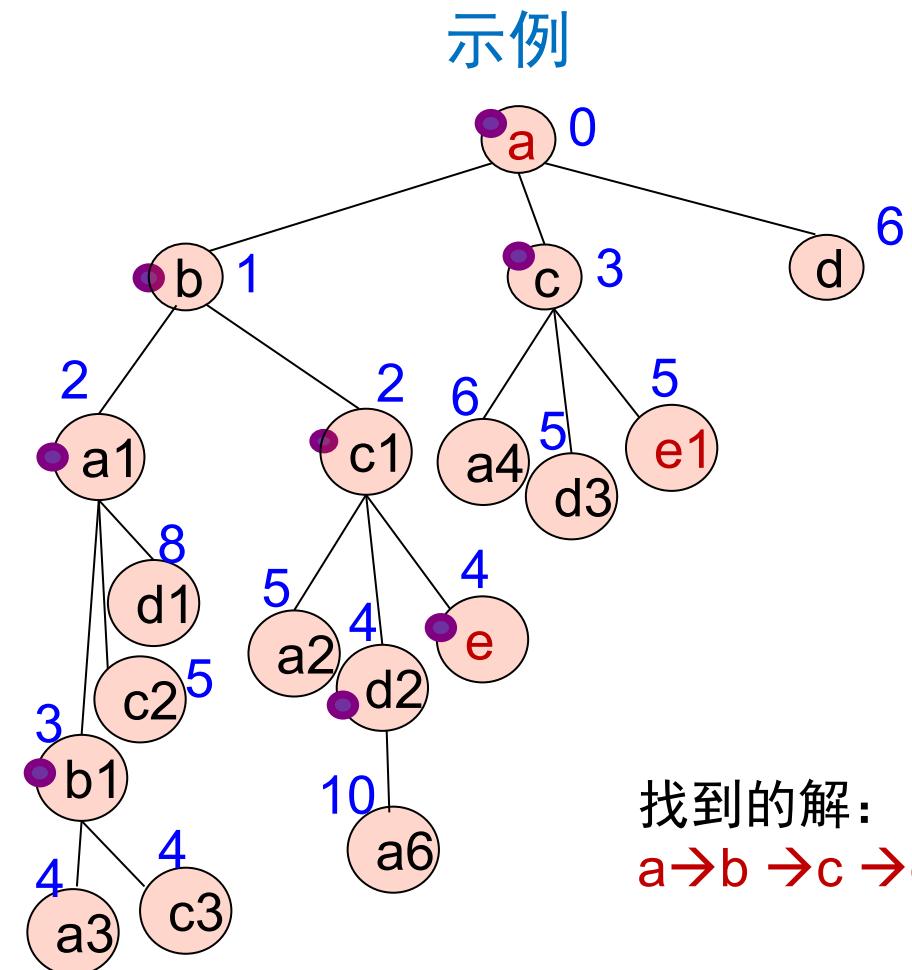


# 代价一致搜索 (树搜索)



Initial state = a  
Goal state = e

- 完备性: Yes
  - 分支因子有限且步骤成本大于0
- 最优性: Yes



找到的解:  
 $a \rightarrow b \rightarrow c \rightarrow e$

当多个节点代价相等时，常用的是FIFO原则，  
也就是扩展先加入队列的节点



# BFS vs DFS vs UCS

## 树搜索版本

	完备性	最优性	时间复杂度	空间复杂度
广度优先	Yes	No	$\mathcal{O}(b^d)$	$\mathcal{O}(b^d)$
深度优先	No	No	$\mathcal{O}(b^m)$	$\mathcal{O}(bm)$
代价一致	Yes	Yes	$\mathcal{O}(b^{1+C^*/\epsilon})$	$\mathcal{O}(b^{1+C^*/\epsilon})$

## 图搜索版本

	完备性	最优性	时间复杂度	空间复杂度
广度优先	Yes	No	$\min(s, \mathcal{O}(b^d))$	$\min(s, \mathcal{O}(b^d))$
深度优先	Yes	No	$\min(s, \mathcal{O}(b^m))$	$\min(s, \mathcal{O}(b^m))$
代价一致	Yes	Yes	$\min(s, \mathcal{O}(b^{1+C^*/\epsilon}))$	$\min(s, \mathcal{O}(b^{1+C^*/\epsilon}))$

如何降低时间/空间复杂度？



# 本节课安排

## □ 无信息搜索

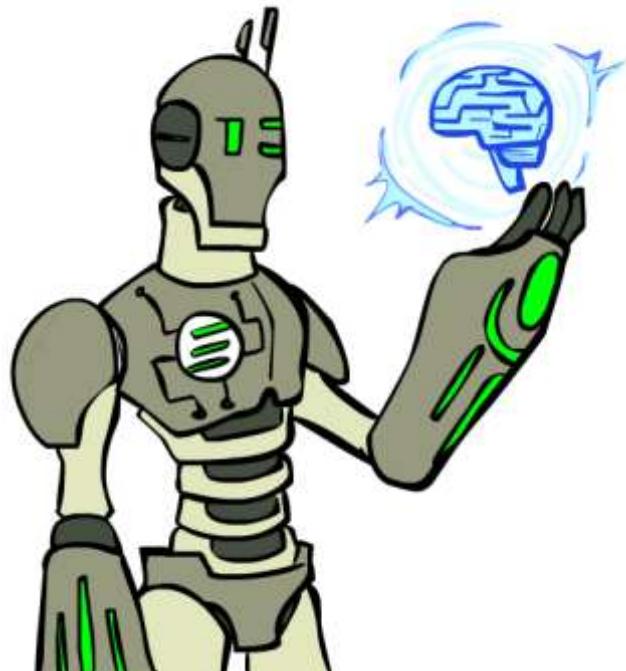
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)





# 有信息搜索

---

## □ 有信息（启发式）搜索：

- ✓ 使用问题本身的定义之外的**特定知识**进行问题求解
- ✓ 比无信息搜索更有效



# 有信息搜索的基本框架

## □ 评估函数 (Evaluation Function)

- $f(n)$ : 评估节点 $n$ 的优劣程度；在搜索中， $f(n)$ 是某种代价，是效用 (utility) 的减函数

- utility =  $\frac{1}{f(n)}$

## □ 优先选择 “好” 的节点进行探索

## □ 算法框架和代价一致搜索(UCS)完全相同

- UCS:  $f(n) = g(n)$
- BFS:  $f(n) = \text{节点 } n \text{ 的深度}$
- DFS:  $f(n) = 1/\text{节点 } n \text{ 的深度}$

## □ 关键: 如何设计评估函数?

- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)



# 本节课安排

## □ 无信息搜索

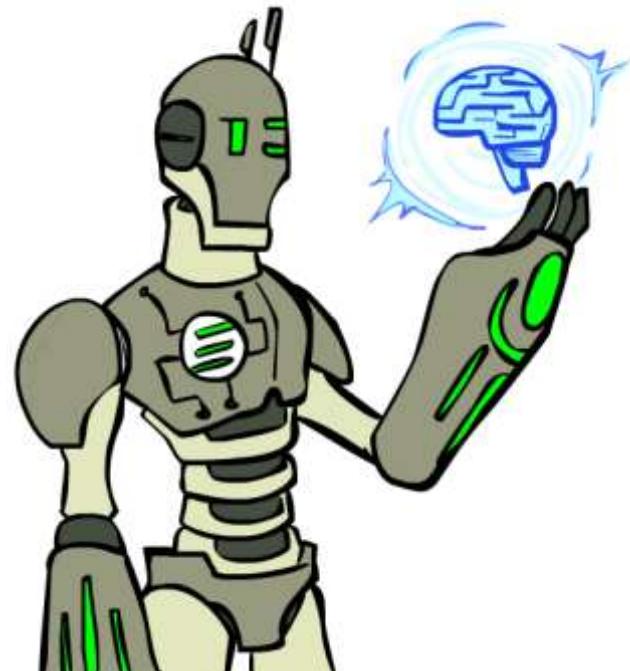
### **Uninformed Search**

- 广度优先搜索 (BFS)
- 深度优先搜索 (DFS)
- 代价一致搜索 (UCS)

## □ 有信息搜索

### **Informed Search**

- 贪婪搜索(Greedy Search)
- A\*搜索 (A\* Search)

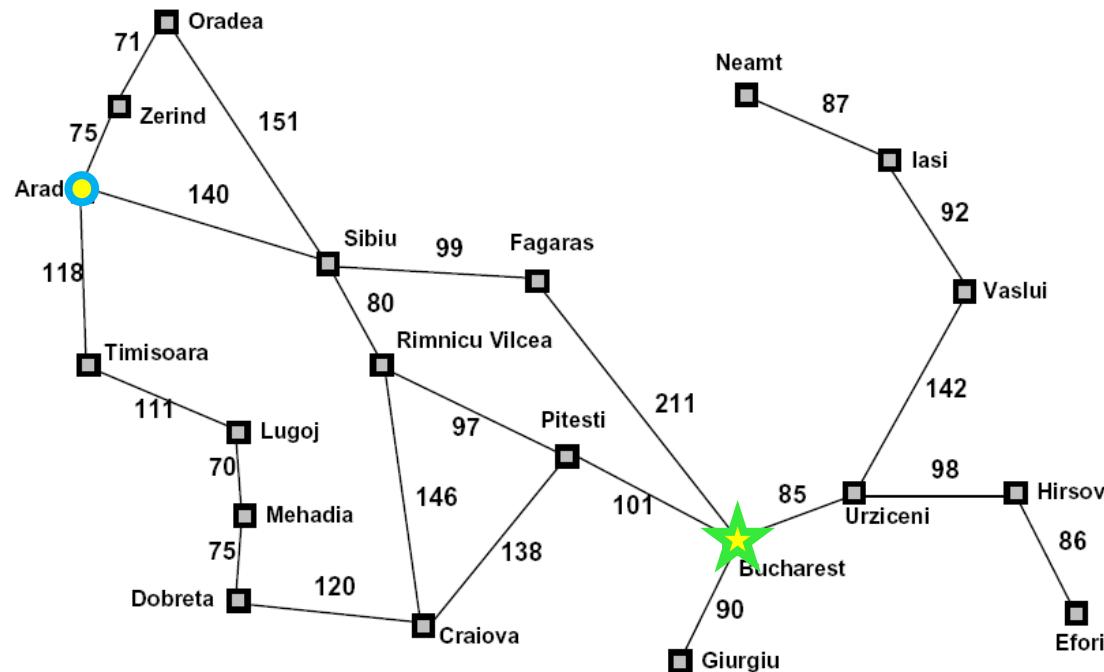




# 启发函数

# 口 启发函数 (Heuristic Function)

- $h(n)$ : 估计从当前节点  $n$  到目标状态的最小路径成本



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

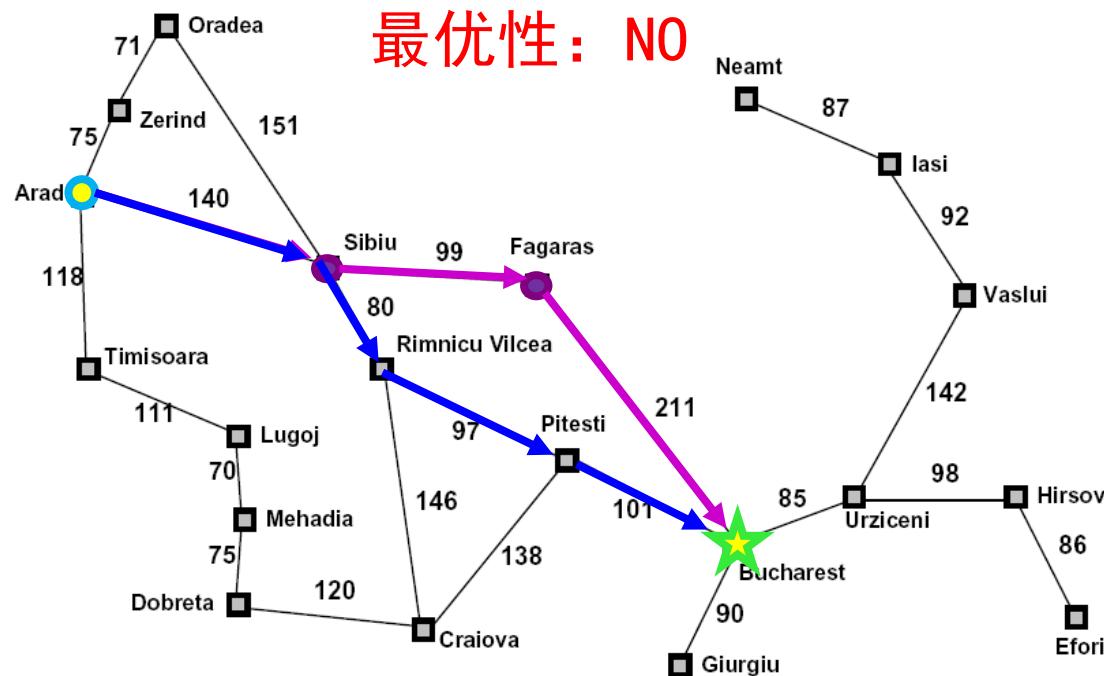
$h(n)$ 为节点 $n$ 到目标的直线距离



# 贪婪搜索

## 口 启发函数 (Heuristic Function)

- $h(n)$ : 估计从当前节点 $n$ 到目标状态的最小路径成本



Straight-line distance to Bucharest	
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

“贪婪” 每一步都要试图找到离目标直线距离最近的节点

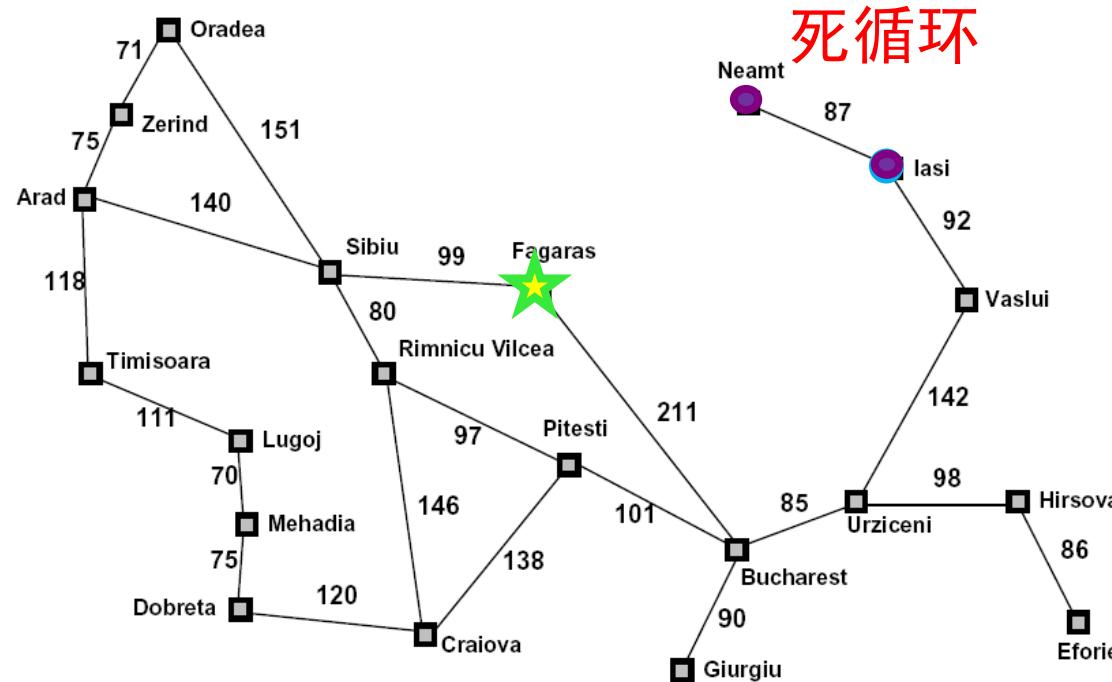
$h(n)$ 为节点 $n$ 到目标的直线距离



# 贪婪搜索

## 口 启发函数 (Heuristic Function)

- $h(n)$ : 估计从当前节点 $n$ 到目标状态的最小路径成本

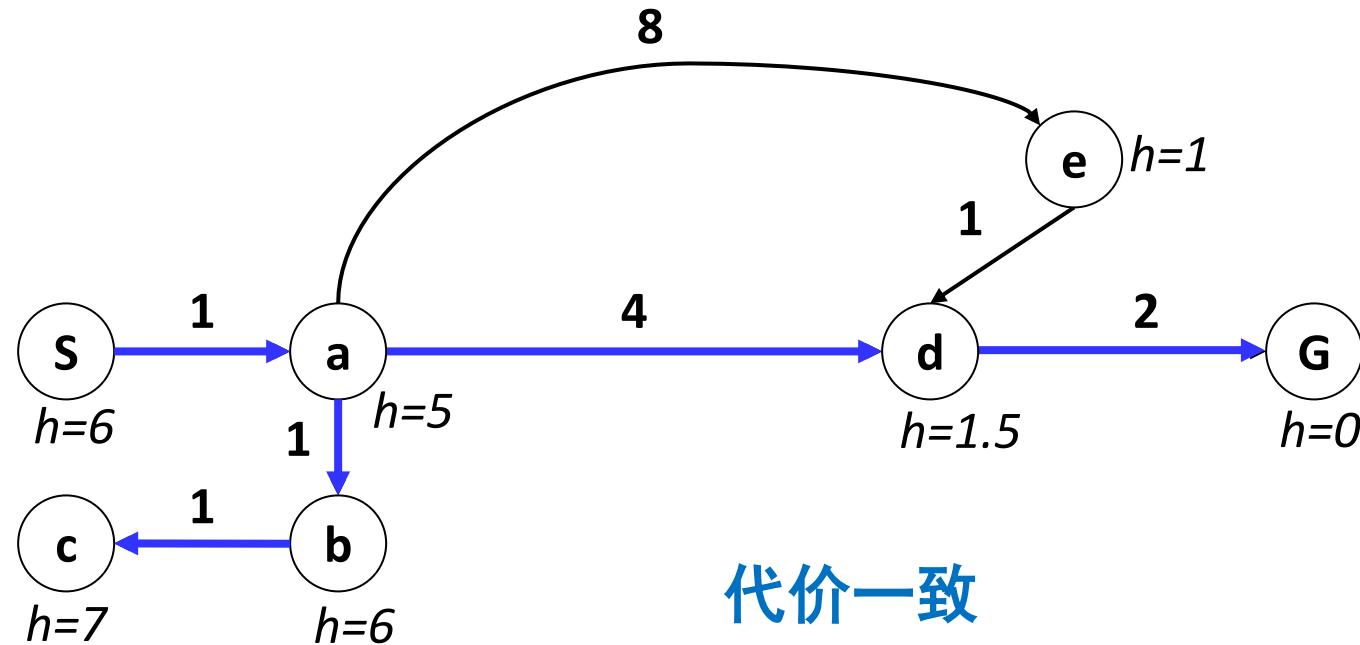


完备性: NO



# A\*搜索

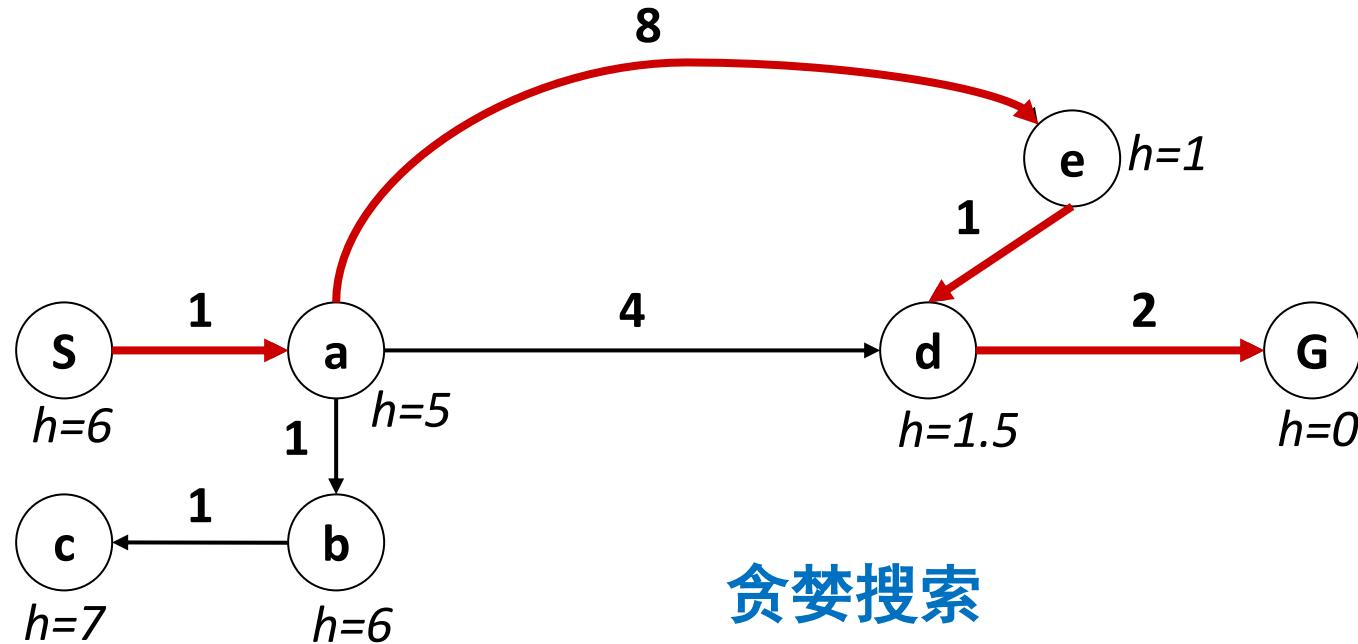
- 最佳优先搜索的最广为人知的形式：**A\*搜索**
- **A\*搜索：**  $f(n) = g(n) + h(n)$
- 只要  $h(n)$  满足一定条件，A\*是完备的、最优的；而且拥有“最佳效率”（optimally efficient），即：某种意义上，计算开销最小





# A\*搜索

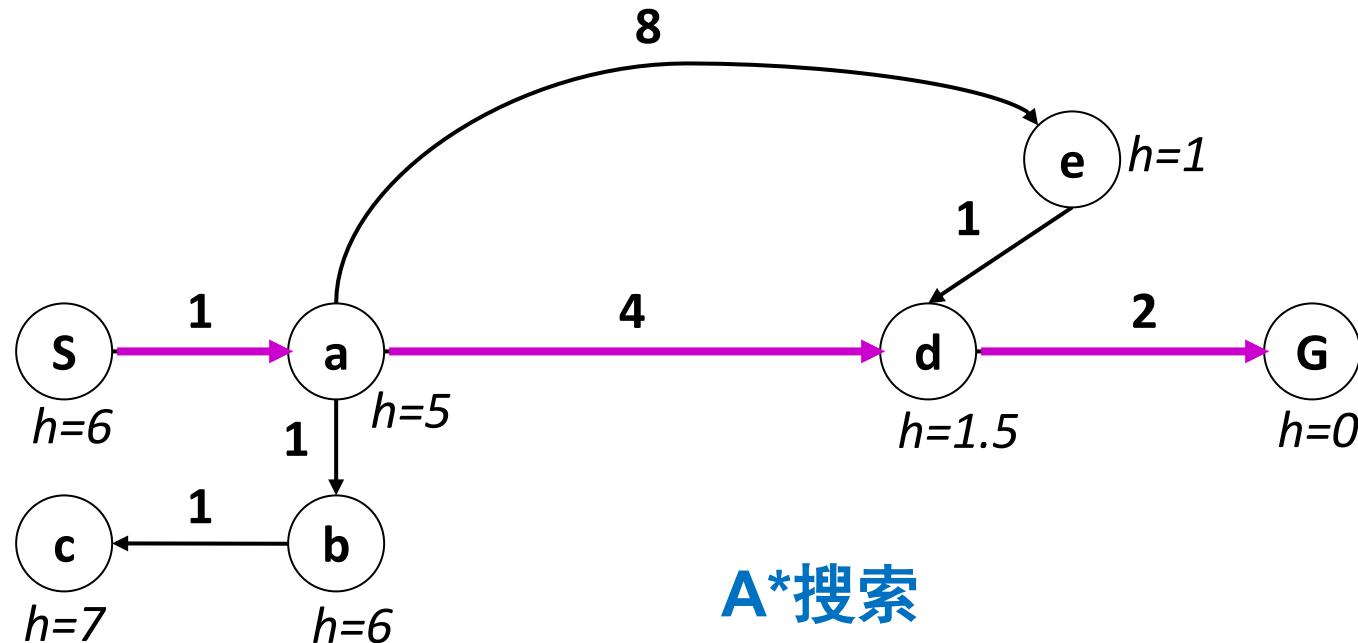
- 最佳优先搜索的最广为人知的形式： **A\*搜索**
- **A\*搜索**:  $f(n) = g(n) + h(n)$
- 只要  $h(n)$  满足一定条件， A\* 是完备的、最优的；而且拥有“最佳效率”（optimally efficient），即：某种意义上，计算开销最小





# A\*搜索

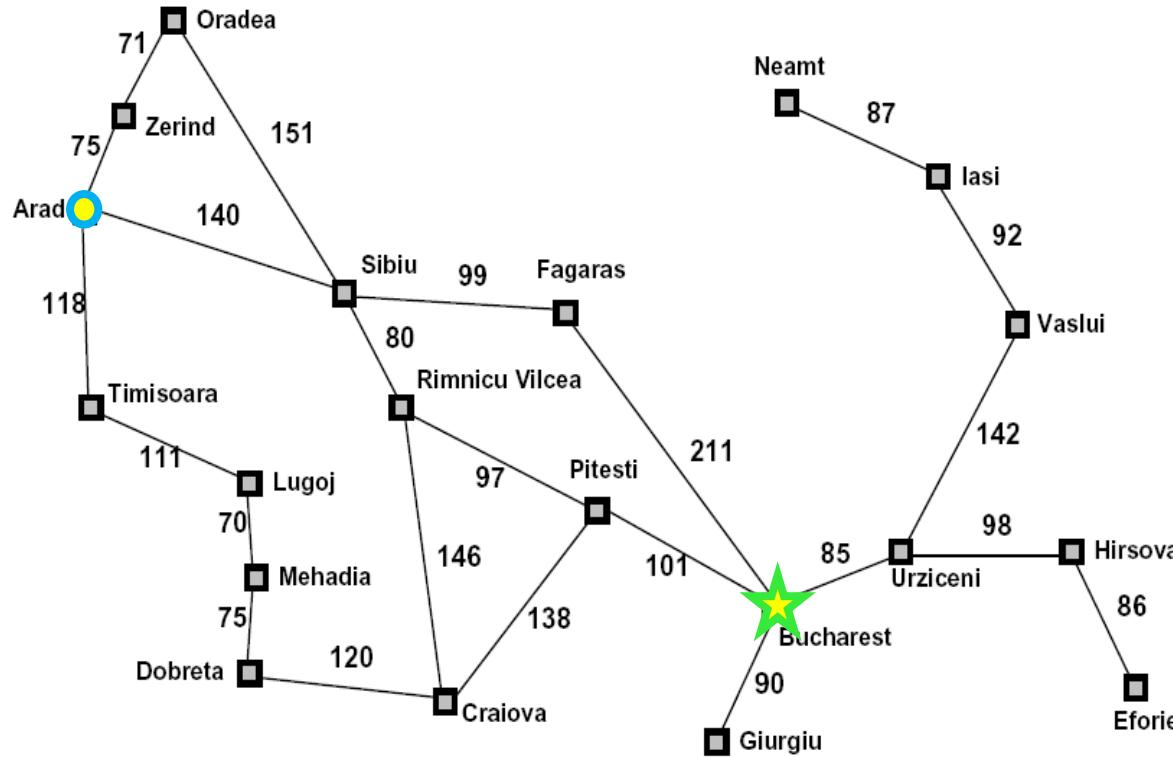
- 最佳优先搜索的最广为人知的形式： **A\*搜索**
- **A\*搜索**:  $f(n) = g(n) + h(n)$
- 只要  $h(n)$  满足一定条件， A\* 是完备的、最优的；而且拥有“最佳效率”（optimally efficient），即：某种意义上，计算开销最小





# 相容性 (Admissible)

- 启发函数  $h(n)$  是相容的 (admissible) , 当且仅当:  
$$0 \leq h(n) \leq h^*(n)$$
  - $h^*(n)$ : 从节点n到达目标的真实最小成本



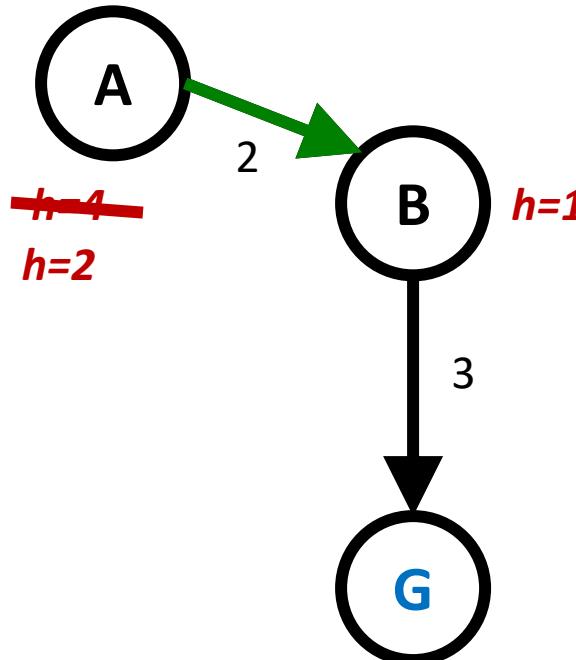
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



# 一致性 (Consistency)

- 启发函数  $h(n)$  是一致的 (consistent) , 当且仅当:  
$$h(n) \leq c(n, a, n') + h(n')$$

- $a$ : 节点 $n$ 状态下的任一动作
- $n'$ : 节点 $n$ 的子节点,  $n'.state = \text{Result}(n, a)$



**性质:** 如果 $h$ 是一致性的，则 $h$ 一定也是相容的；但反之不成立。



# A\*算法的性质

## □ 最优性:

- 树搜索版本:

- $h(n)$ 是相容的 (admissible)

UCS的最优性

$$h(n) = 0$$

- 图搜索版本:

- $h(n)$ 是一致的 (consistent)

## □ 完备性:

- 同代价一致搜索 (UCS)

$$\epsilon = (h^* - h)/h^*$$

$$\mathcal{O}(b^{\epsilon d})$$

$$\mathcal{O}(1) \text{ if } \epsilon = 0$$

## □ 优势:

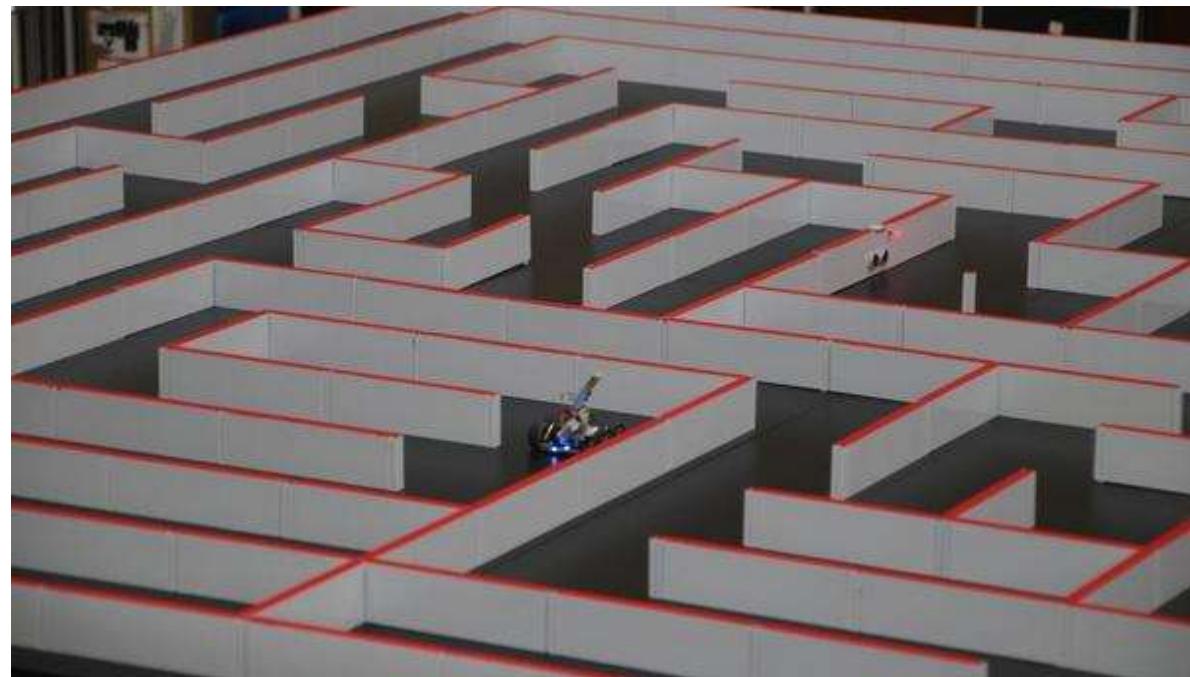
- 对于指定的 $h$ , 计算开销最小 (效率最优)

One final observation is that among optimal algorithms of this type—algorithms that extend search paths from the root and use the same heuristic information—A\* is **optimally efficient** for any given consistent heuristic. That is, no other optimal algorithm is guaranteed to expand fewer nodes than A\* (except possibly through tie-breaking among nodes with  $f(n) = C^*$ ). This is because any algorithm that *does not* expand all nodes with  $f(n) < C^*$  runs the risk of missing the optimal solution.



# A\*算法的应用

- 机器人
- 路径/路由问题
- 资源规划问题
- 语言分析
- 机器翻译
- 语音识别
- 视频游戏
- ...

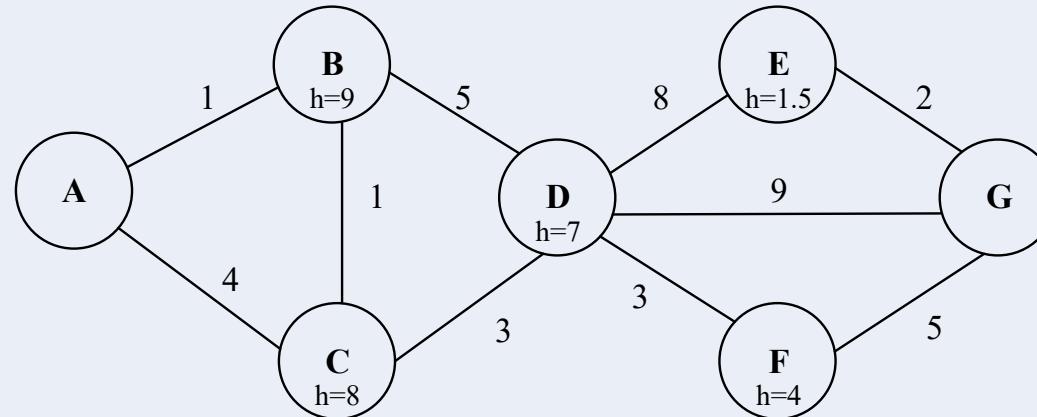




# 练习

— (本题共16分) 以下问题涉及确定性问题搜索知识，回答问题。

考虑下图所示的状态空间结构图。其中，A为起始节点；G为目标节点；相邻节点边上的数值为路径耗散值，节点内部的数值 $h$ 为当前节点到目标节点的启发代价。



1. (选择、填空题, 本小题共6分)

- a) 图上给出的启发代价 (是或否) 相容 (admissible) ? (请在是或否上打钩)  
b) 本课程介绍了\_\_\_\_\_ (B1) 、\_\_\_\_\_ (B2) 、\_\_\_\_\_ (B3) 三种无信息搜索方法；以及\_\_\_\_\_ (B4) 、\_\_\_\_\_ (B5) 两种有信息搜索方法。

2. (填空题, 本小题共10分)

注：采用图搜索版本。同一层节点入栈或入队列按照字母序排列，即A-Z顺序。

采用(B1)搜索方法，得到的解为\_\_\_\_\_，访问状态顺序为\_\_\_\_\_。

采用(B2)搜索方法，得到的解为\_\_\_\_\_，访问状态顺序为\_\_\_\_\_。

采用(B3)搜索方法，得到的解为\_\_\_\_\_，访问状态顺序为\_\_\_\_\_。

采用(B4)搜索方法，得到的解为\_\_\_\_\_，访问状态顺序为\_\_\_\_\_。

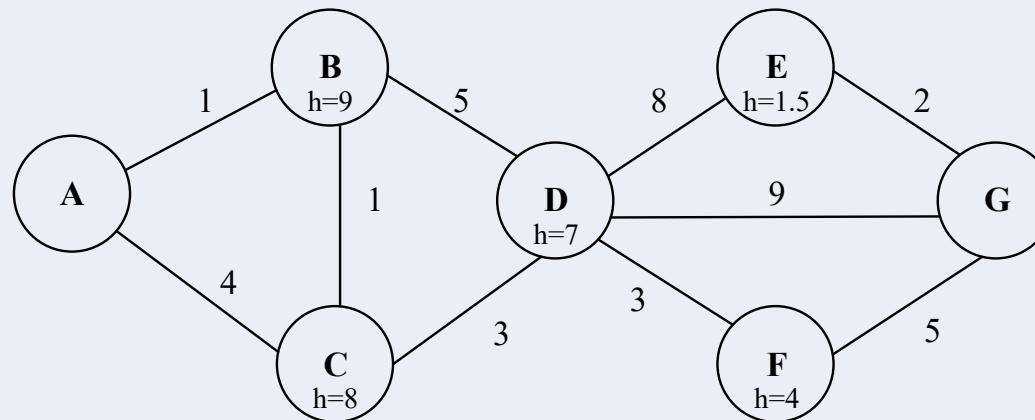
采用(B5)搜索方法，得到的解为\_\_\_\_\_，访问状态顺序为\_\_\_\_\_。



# 答案

— — (本题共16分) 以下问题涉及确定性问题搜索知识，回答问题。

考虑下图所示的状态空间结构图。其中，A为起始节点；G为目标节点；相邻节点边上的数值为路径耗散值，节点内部的数值 $h$ 为当前节点到目标节点的启发代价。



1. (选择、填空题, 本小题共6分)

a) 图上给出的启发代价 (是或否) 相容 (admissible) ? (请在是或否上打钩)

b) 本课程介绍了 BFS (B1) 、 DFS (B2) 、 UCS (B3) 三种无信息 (图) 搜索方法；以及 Greedy (B4) 、  $A^*$  (B5) 两种有信息 (图) 搜索方法。

2. (填空题, 本小题共10分)

注：采用图搜索版本。同一层节点入栈或入队列按照字母序排列，即A-Z顺序。

采用(B1)搜索方法，得到的解为 ABDG ， 访问状态顺序为 ABCDEFG 。

采用(B2)搜索方法，得到的解为 ACDG ， 访问状态顺序为 ACDG 。

采用(B3)搜索方法，得到的解为 ABCDGF ， 访问状态顺序为 ABCDFEG 。

采用(B4)搜索方法，得到的解为 ACDG ， 访问状态顺序为 ACDG 。

采用(B5)搜索方法，得到的解为 ABCDGF ， 访问状态顺序为 ABCDGF 。