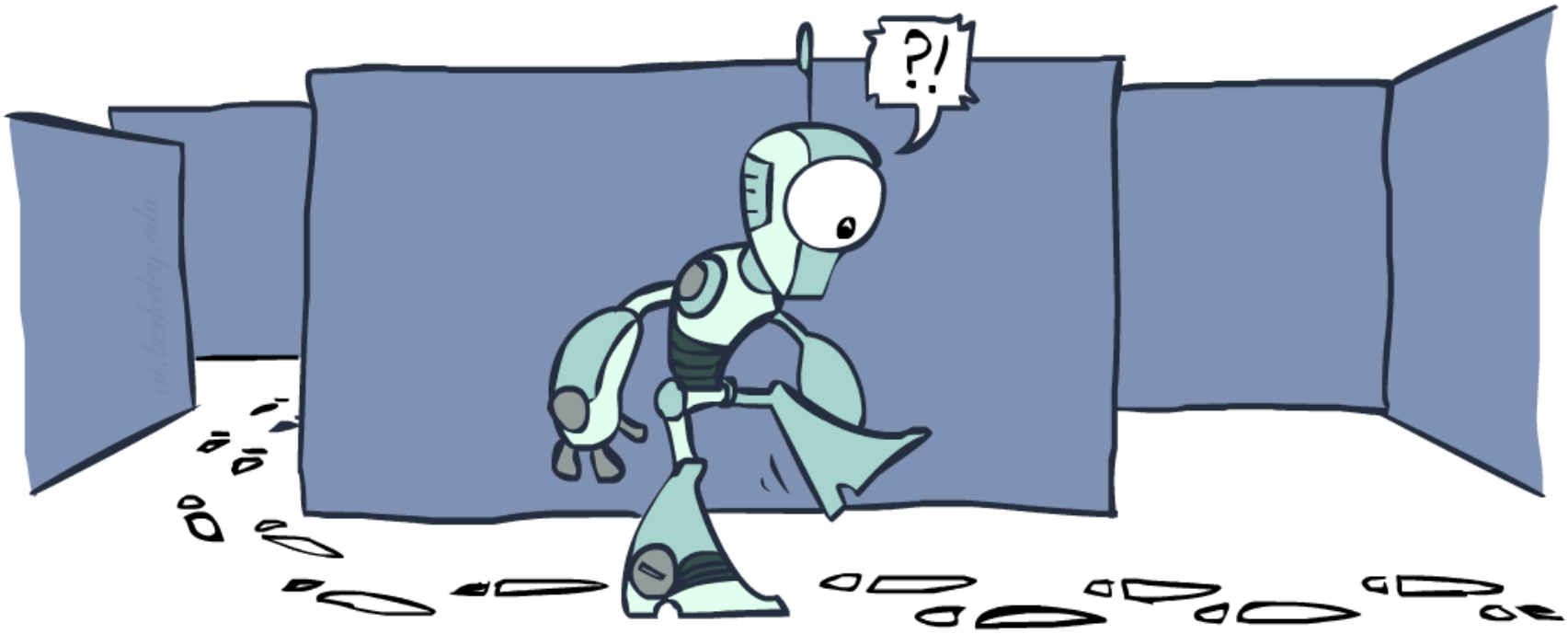
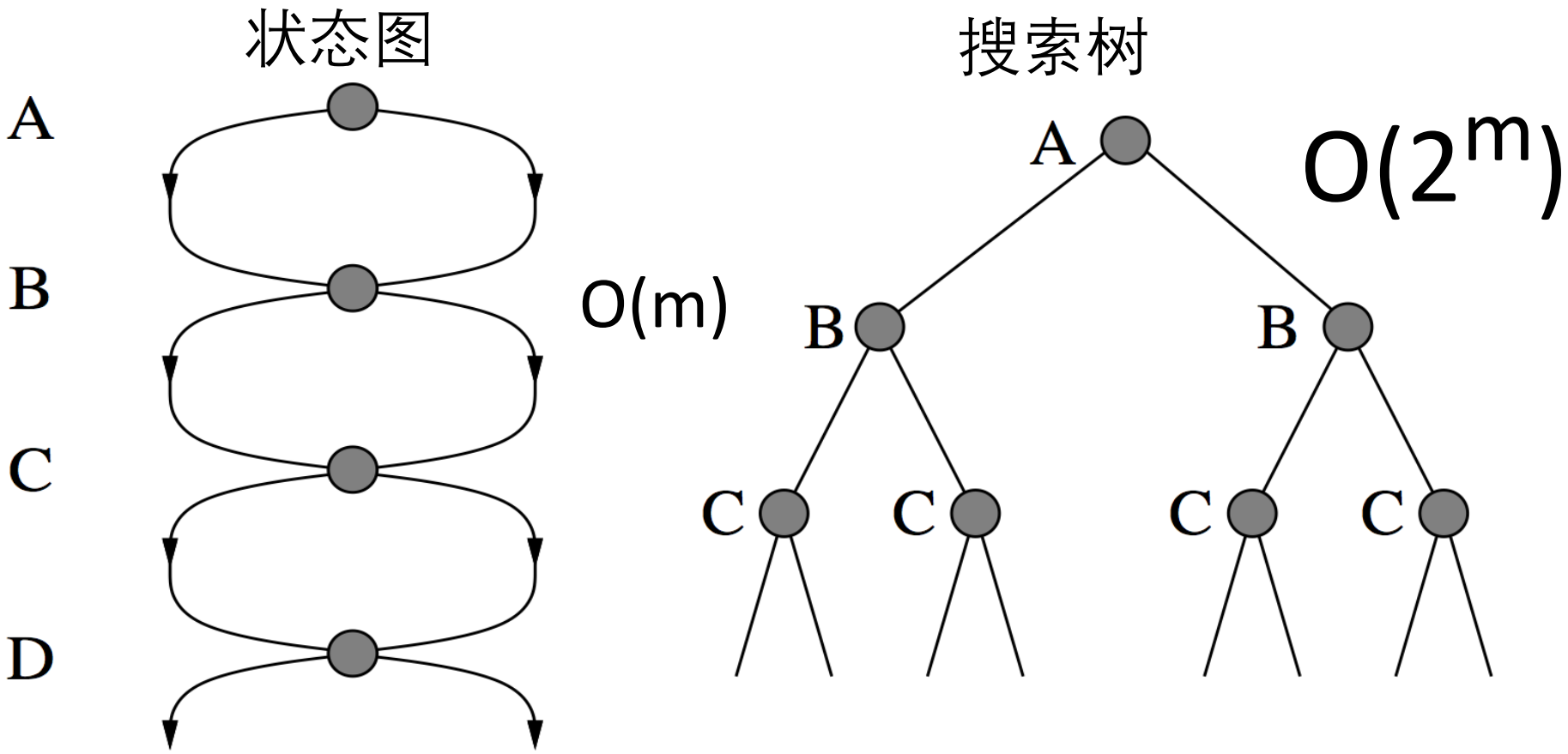
树搜索 VS 图搜索



树搜索的思考

- 没有检测重复出现的状态，导致产生节点成树深度的指数增长



通用树搜索算法

```
function 树搜索(问题) returns 一个 解, 或 失败
    把问题的初始状态放入 搜索前沿
     loop do
          if 搜索前沿 为空 then return 失败
          选择一个叶 节点 并把它从 搜索前沿 中移除
          if 这个 节点 包含一个目标状态 then return 相应的解（路径）
          扩展这个选择的 节点, 把扩展结果的 节点 加入 搜索前沿
```

通用图搜索

每个状态在构建的搜索树上最多只出现一次

Function 图搜索(问题) **returns** 一个 解, 或是失败

把问题的初始状态放入 搜索前沿

初始化已探索节点集为空

loop do

if 探索前沿 为空 then return 失败

选择一个叶 节点 并把它从 搜索前沿 中移除

if 这个节点包含一个目标状态 **then return** 相关路径

把这个节点加入已探索节点集

扩展这个选择的 节点, 把扩展的 子节点 加入 搜索前沿

但是只有当该子节点不在搜索前沿或已探索节点集里

树搜索 VS 图搜索

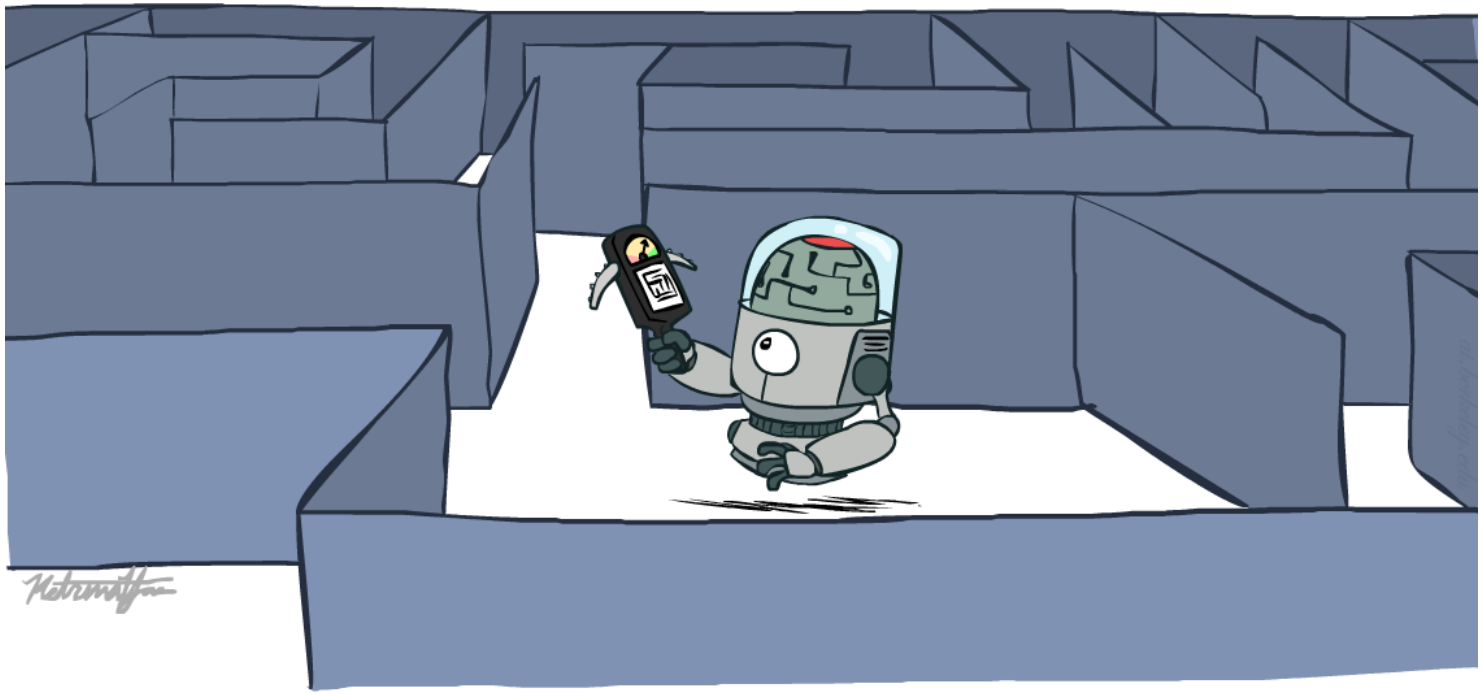
- 图搜索
  - 避免无限循环：在一个有限的状态空间里，m 是有限的
  - 消除了成指数增长的重复路径
  - 要求内存空间不断增长!

搜索算法出错了吗?


MAPQUEST
Brier Ave
McCready Ave
Hidalgo Ave
Silver Ridge Ave
Lake View Ave
Silver Lake Blvd
Teviot St
Bancroft Ave
Earl St
Loma Vista Pl
END
Cove Ave
Moore St
Oak Glen
Apex Ave
Glendale Blvd
Baxter St
Ewing St
2005 MapQuest.com, Inc.

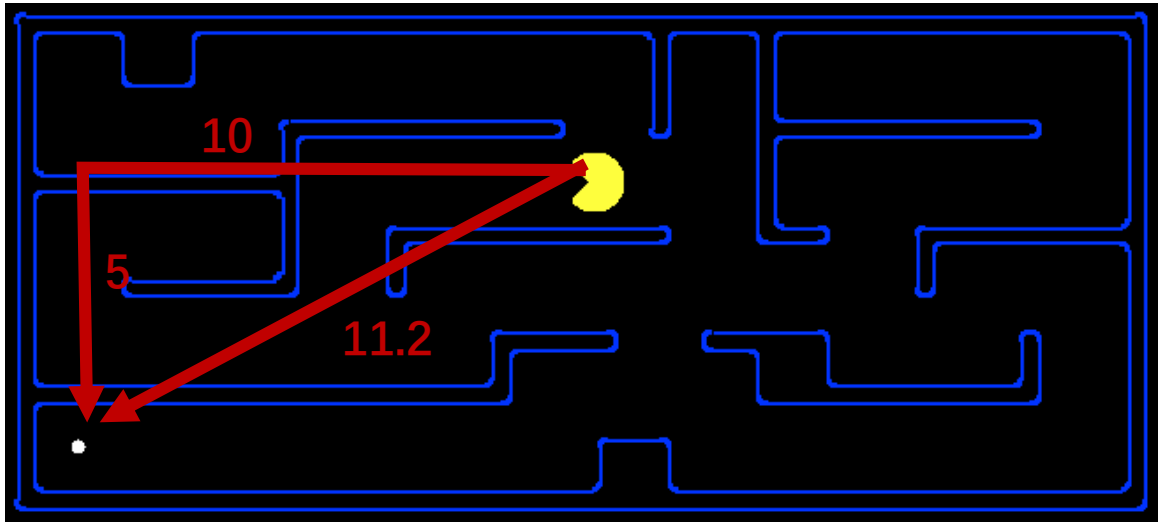

NORTH
SOUTH
WEST
EAST
ARCTIC OCEAN
ATLANTIC OCEAN
ICELAND
MapPoint
End
Start
Stockholm
Helsinki
Helsingfors
Tver
RUSSIA
Riga
Smolensk
Vilnius
Växjö
Arhus
Aarhus
Edinburgh
Copenhagen
Białystok
BELARUS
Kiev
IRE.
U.K.
Amsterdam
Berlin
POLAND
Wrocław
UKRAINE
London
GERMANY
Brussels
CZECH REP.
Chisinau
Paris
Bern
Berne
Vienna
HUNGARY
ROMANIA
Zagreb
FRANCE
ITALY
Bucharest
©2004 Microsoft Corp ©2003 NAVTEQ
km
mi
500
1000
200
400
600
Legend
Zoom on map cli
Start: Haugesund, Rogaland, Norway
End: Trondheim, Sør-Trøndelag, Norway
Total Distance: 2713.2 Kilometers
Estimated Total Time: 47 hours, 31 minutes
nrk.no/allitidmoro

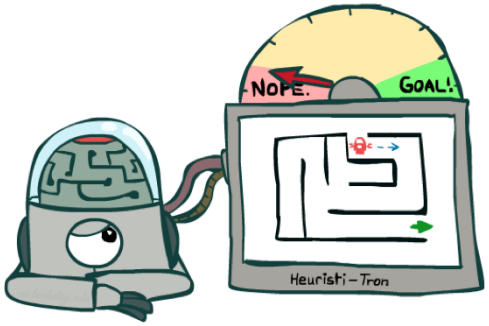
人工智能导论：启发式搜索

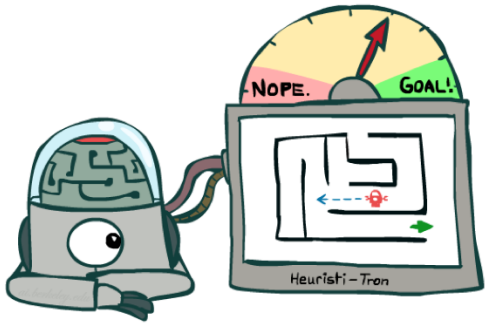


搜索启发法（Heuristics）

- Heuristic :
  - 一个函数估计一个状态有多接近一个目标
  - 为一个特定搜索问题设计的
  - 例如：曼哈顿距离，欧几里德距离

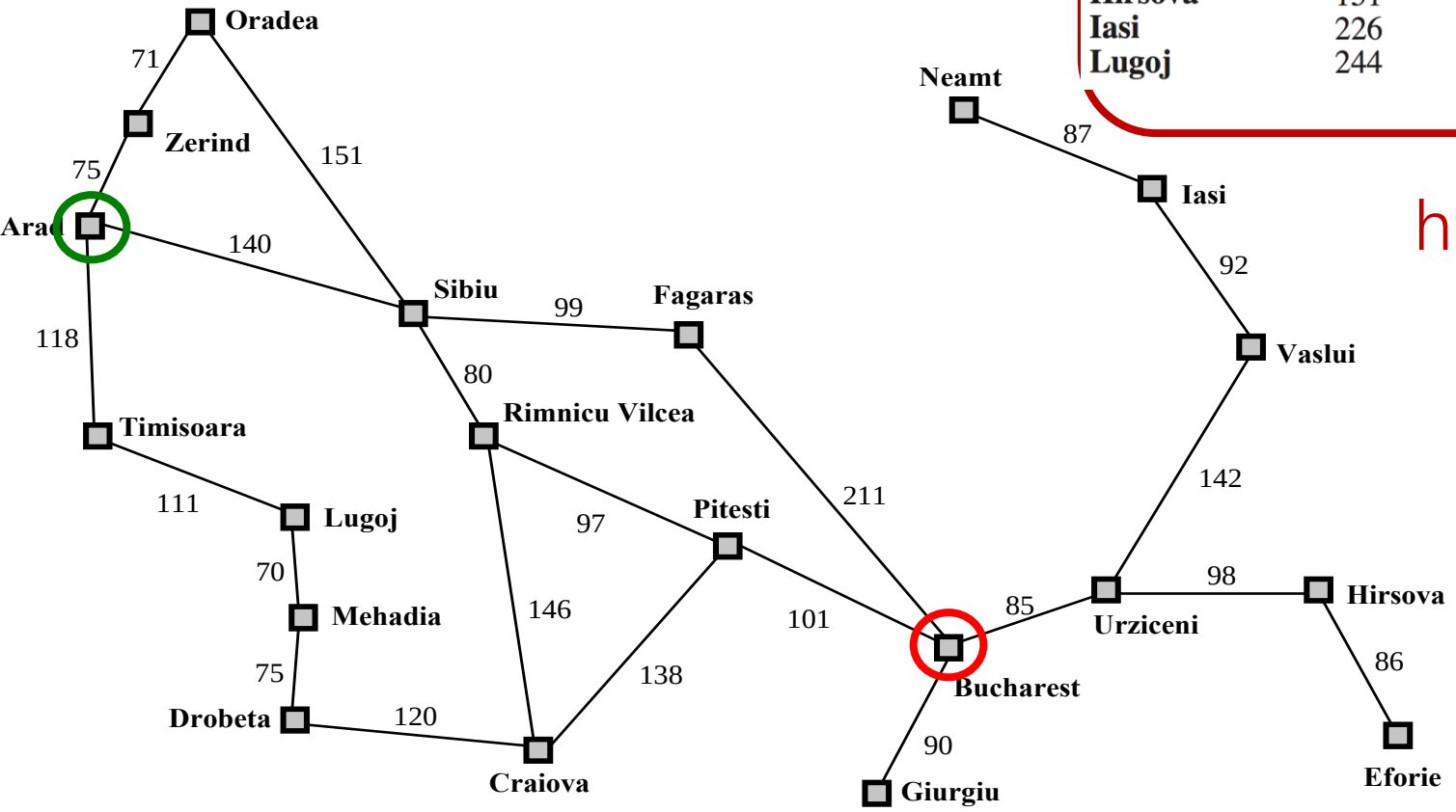






举例：到布加勒斯特的距离（罗马尼亚旅行问题）

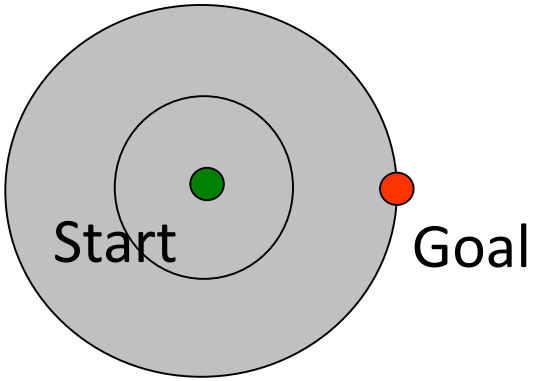
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



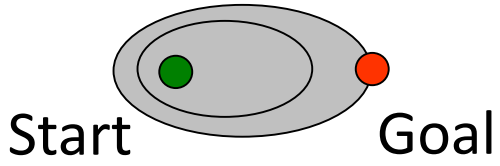
h(x)

启发式的影响

指引搜索过程朝向目标，而不是朝向各个方向里的所有空间



基础搜索（无信息启发）

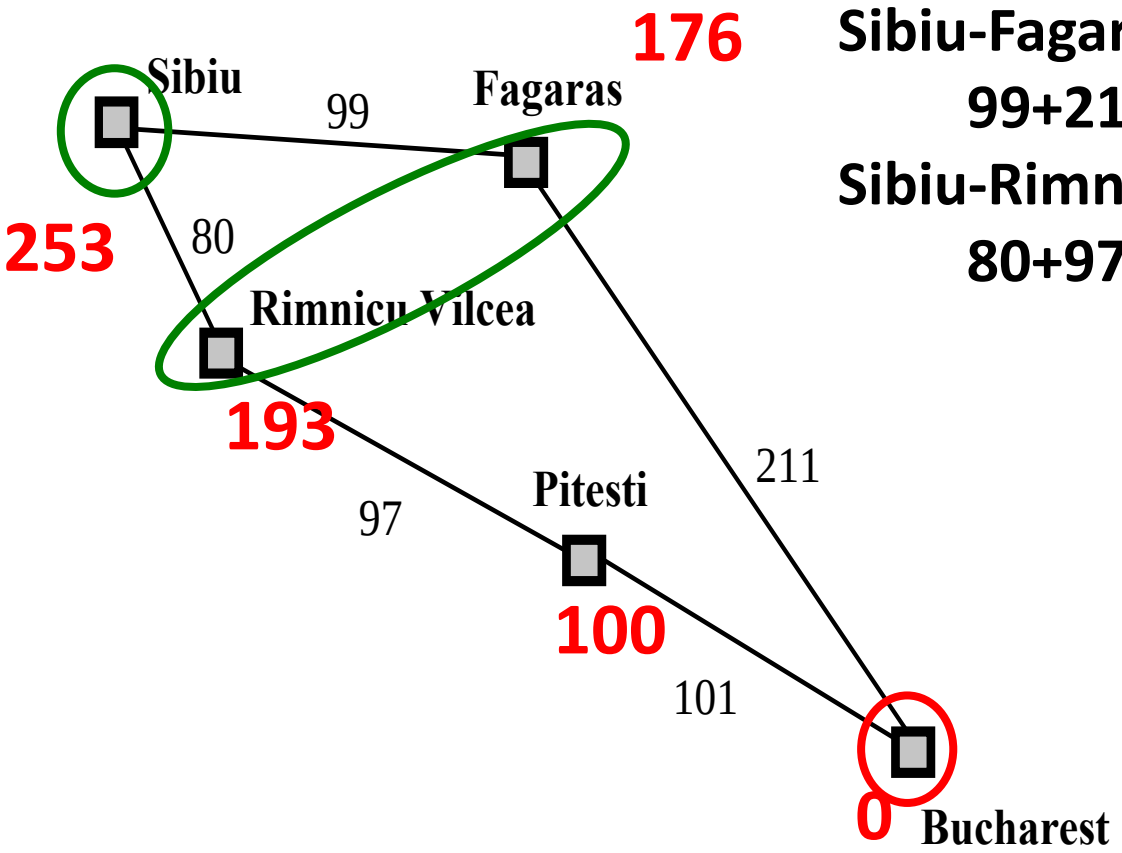


信息启发的

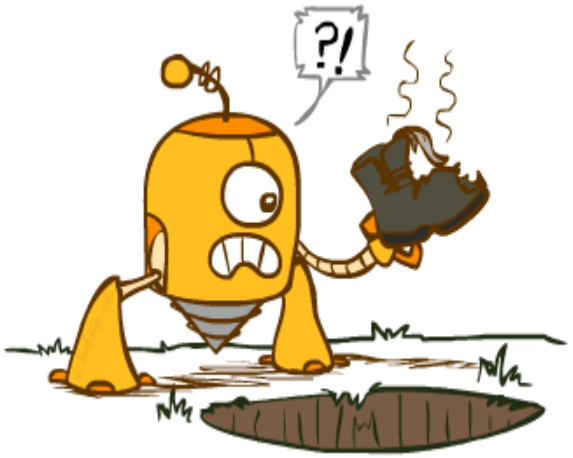
贪婪搜索

贪婪搜索

- 扩展那个离目标似乎最近的节点 （搜索前沿节点按h值排序）
- 会出现什么问题?

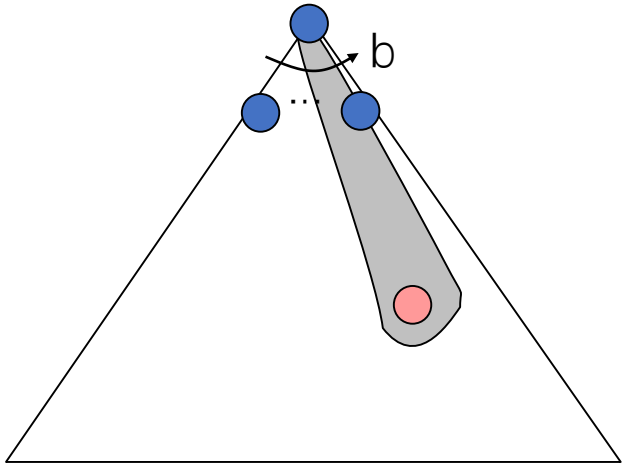


Sibiu-Fagaras-Bucharest =
99+211 = 310
Sibiu-Rimnicu Vilcea-Pitesti-Bucharest =
80+97+101=278

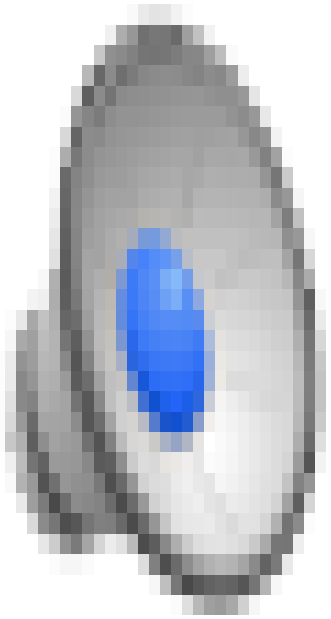


贪婪搜索

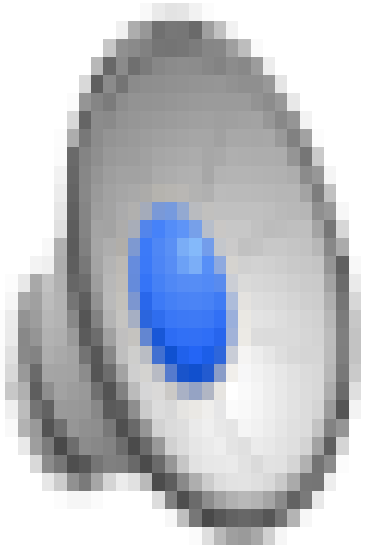
- 策略：扩展一个似乎离目标节点最近的节点，根据 h 值
- 问题1： 没有考虑行动（步骤）成本，导致选择的路径可能长而蜿蜒
- 问题2：依赖于 h 值，可是它也有可能完全是错的



视频展示：贪婪算法（空环境里）

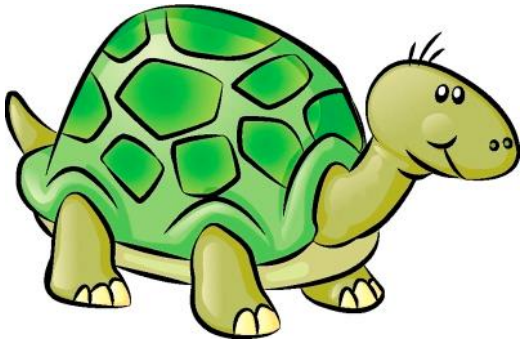


视频展示：贪婪算法（Pacman 小迷宫）



A* 搜索

A* 搜索



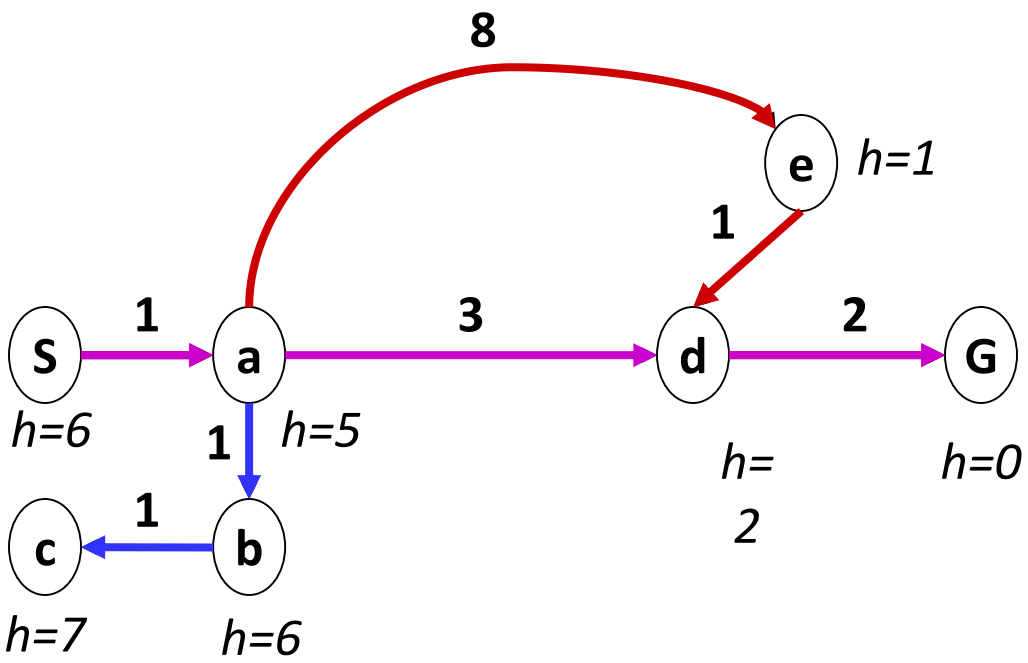
基于成本的统一搜索法（UCS）

A*

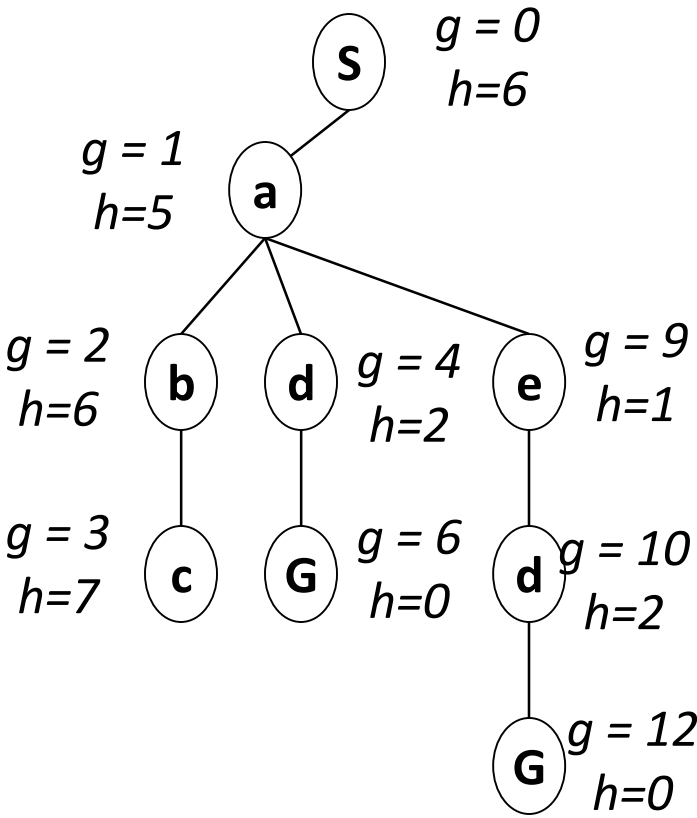
贪婪法（Greedy）

结合统一搜索和贪婪搜索

- 统一：把路径成本排序，即来程的成本 g(n)
- 贪婪：排序按照与目标的临近性，即前程成本 h(n)

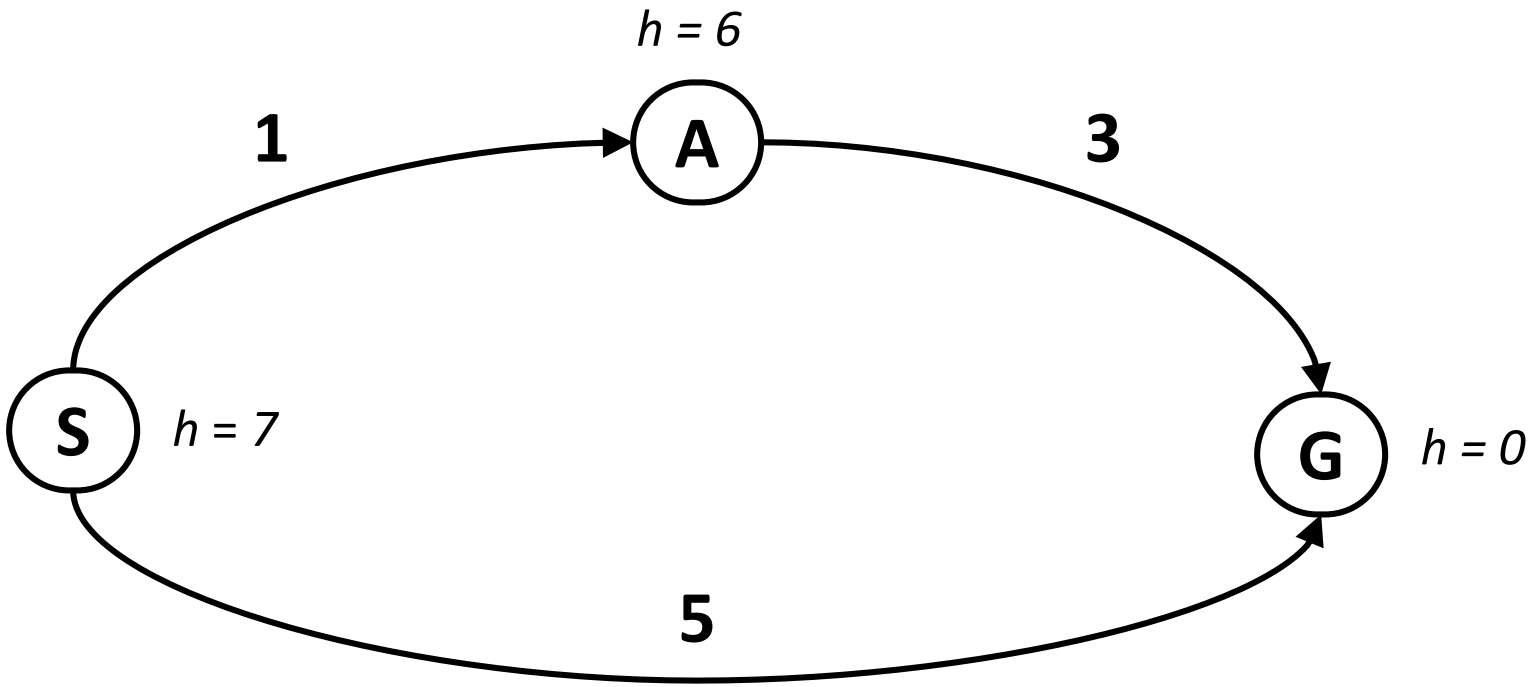


A* 搜索 : f(n) = g(n) + h(n)



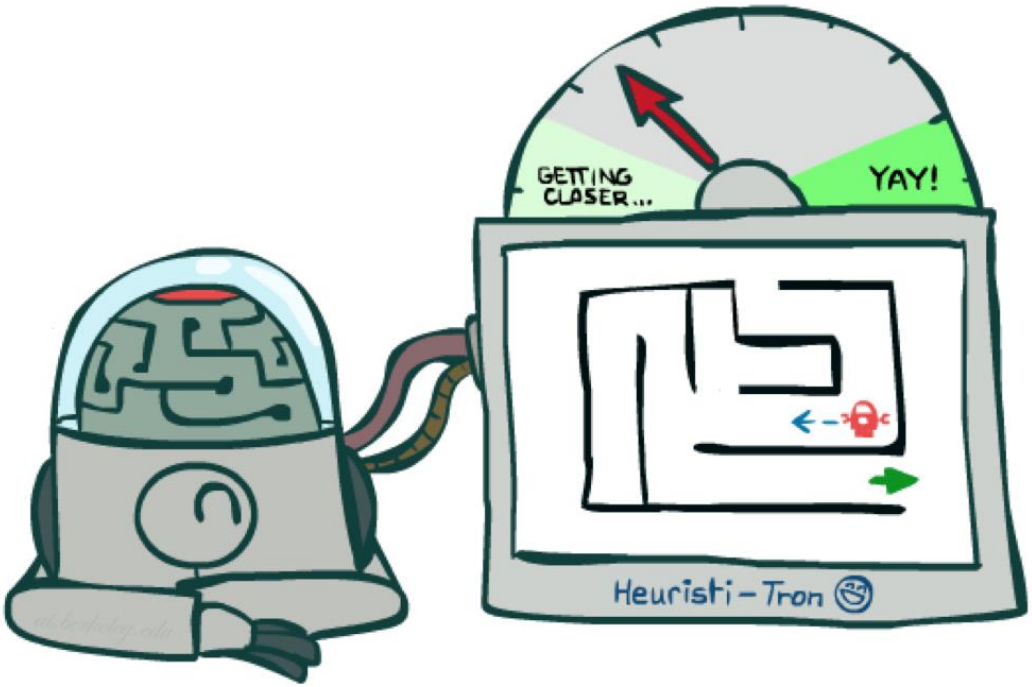
Example: Teg Grenager

A* 是最优的吗？



- 哪个地方错了？
- ***实际到目标成本 < 估计的到目标成本***
- 我们需要估计值小于实际成本

可接纳的启发式函数



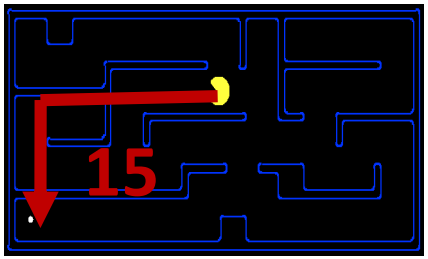
可接纳的启发式函数（Admissible Heuristics）

- h 是 *可接纳的* (乐观的，想的比实际好)：

$$0 \leq h(n) \leq h^*(n)$$

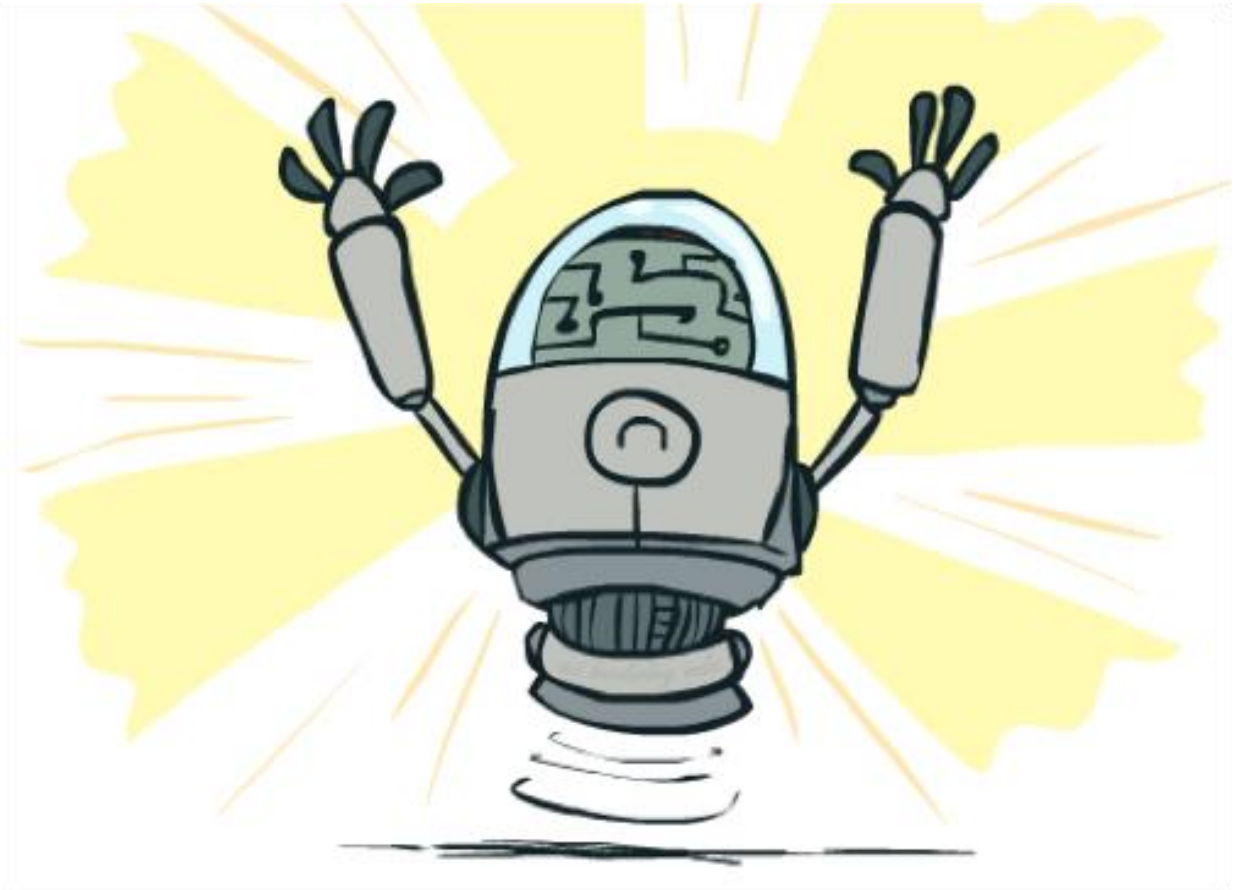
$h^*(n)$ 是到一个最近的目标节点的真成本值

- 举例：



- 在实际应用A*算法时，一个主要任务就是设计可接纳的启发式函数。

A* 树搜索的最优性



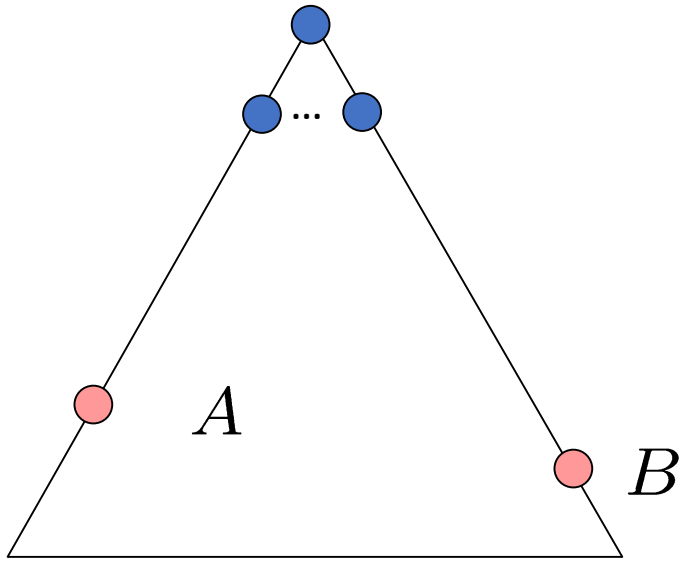
A* 树搜索的最优性

假定:

- A：一个最优目标节点
- B：一个次优目标节点
- h 是可接纳的

如果最优性成立，那么:

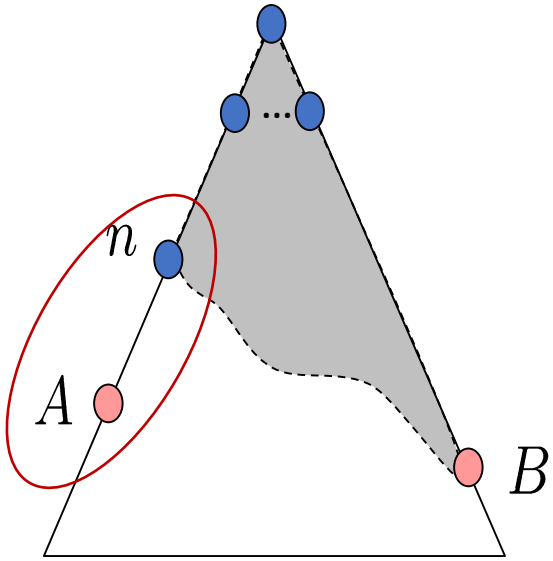
- A 将会在B之前被选择先扩展



A* 树搜索的最优性

证明:

- 假设B在搜索前沿里
- A的某个祖先节点 n 也在搜索前沿里
- 引理: *n* 会先于 B 被选择扩展
    1. f(n) 小于或等于 f(A)



$f(n) = g(n) + h(n)$ f 的定义

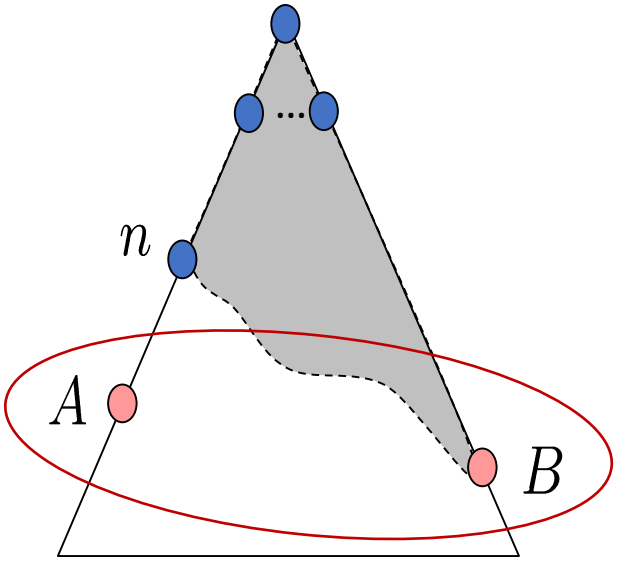
$f(n) \leq g(A)$ h 的可接纳性

$g(A) = f(A)$ h = 0 当在目标节点

A* 树搜索的最优性

证明:

- 假设B在搜索前沿里
- A的某个祖先节点 n 也在搜索前沿里
- 宣称: *n* 会先于 B 被选择扩展
  1. f(n) 小于或等于 f(A)
  2. f(A) 小于 f(B)



$g(A) < g(B)$

$f(A) < f(B)$

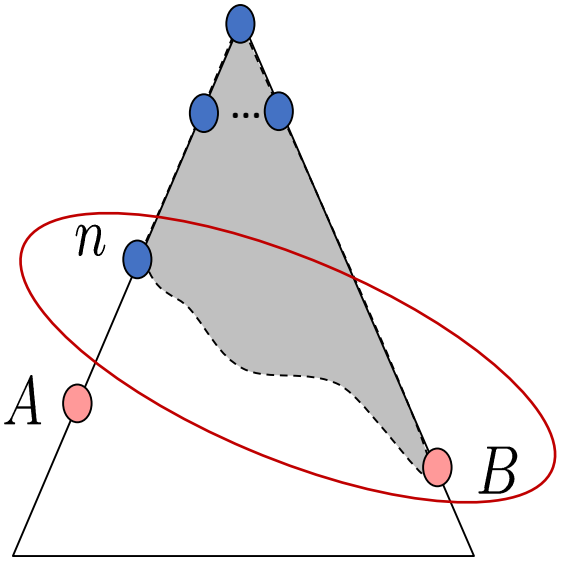
B 次优解

h = 0 在目标节点

A* 树搜索的最优性

证明:

- 假设B在搜索前沿里
- A的某个祖先节点 n 也在搜索前沿里
- 宣称: *n* 会先于 B 被选择扩展
  1. f(n) 小于或等于 f(A)
  2. f(A) 小于 f(B)
  3. *n* 先于 B 被扩展
- 所有 A的祖先节点都会在 B 之前被扩展
- A 先于 B 被扩展
- A* 搜索是最优的

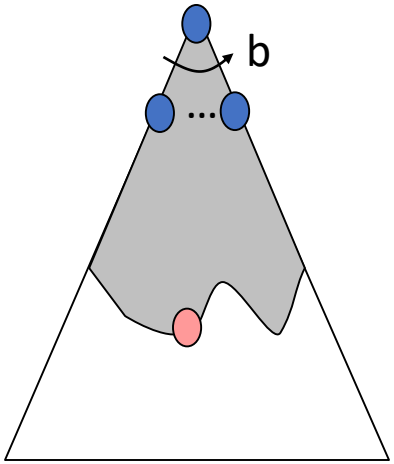


$$f(n) \leq f(A) < f(B)$$

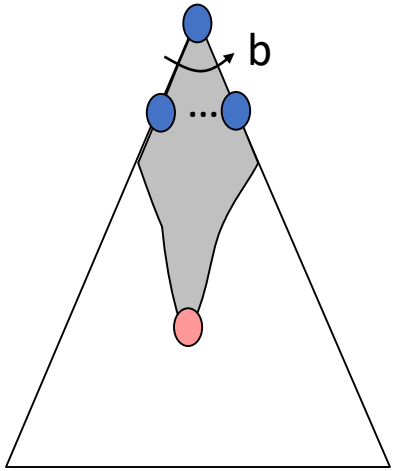
A* 搜索的属性

A*搜索的属性

成本统一搜索

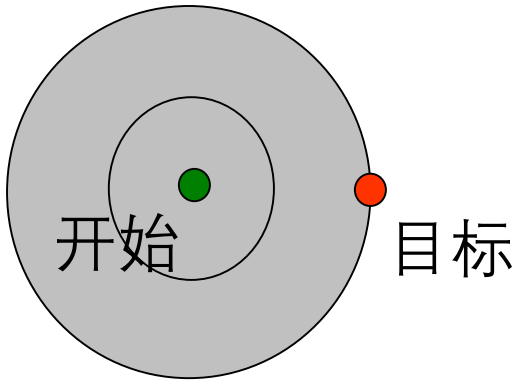


A*搜索

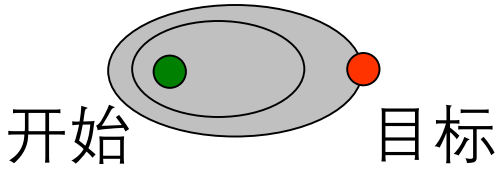


成本统一搜索 vs A* 搜索轮廓

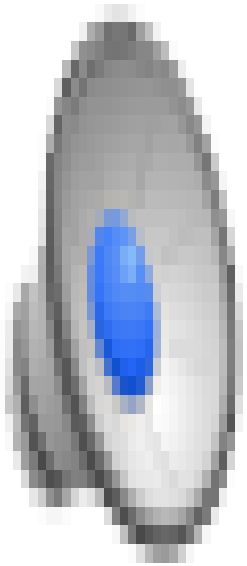
- 基于成本的统一搜索在各个方向上均匀探索



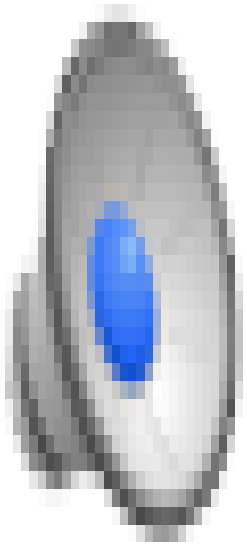
- A* 在朝向目标的方向上进行探索，同时保证解的最优性



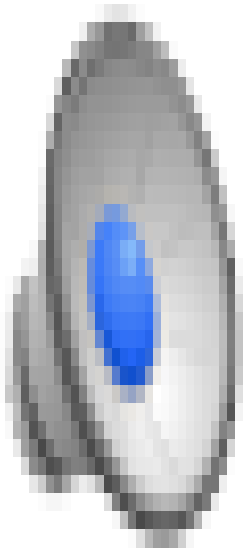
视频展示：搜索轮廓 (空迷宫) – 基于成本的统一搜索



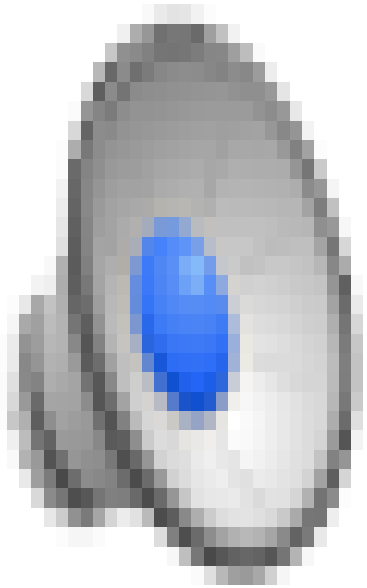
视频展示：搜索轮廓 (空迷宫) – 贪婪搜索



视频展示：搜索轮廓 (空迷宫) – A*



视频展示：搜索轮廓 (Pacman 迷宫) – A*



比较

贪婪搜索

A*

基于成本的统一搜索

A* 应用

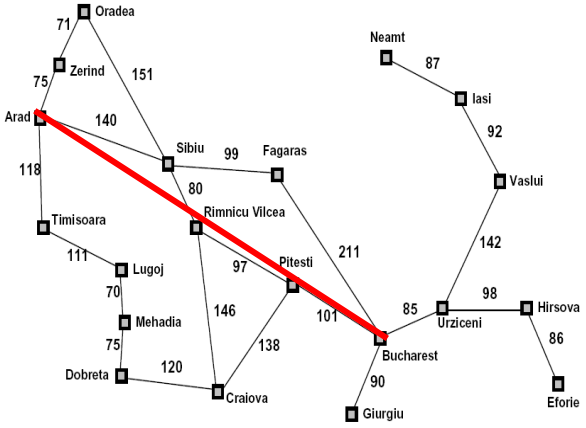
- 视频游戏
- 路径搜索问题
- 资源规划问题
- 机器人移动规划
- 语言分析
- 机器翻译
- 语音识别
- …

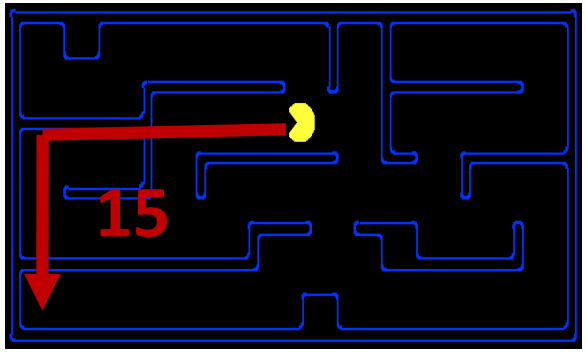
创建启发式函数

创建可接纳的启发式函数

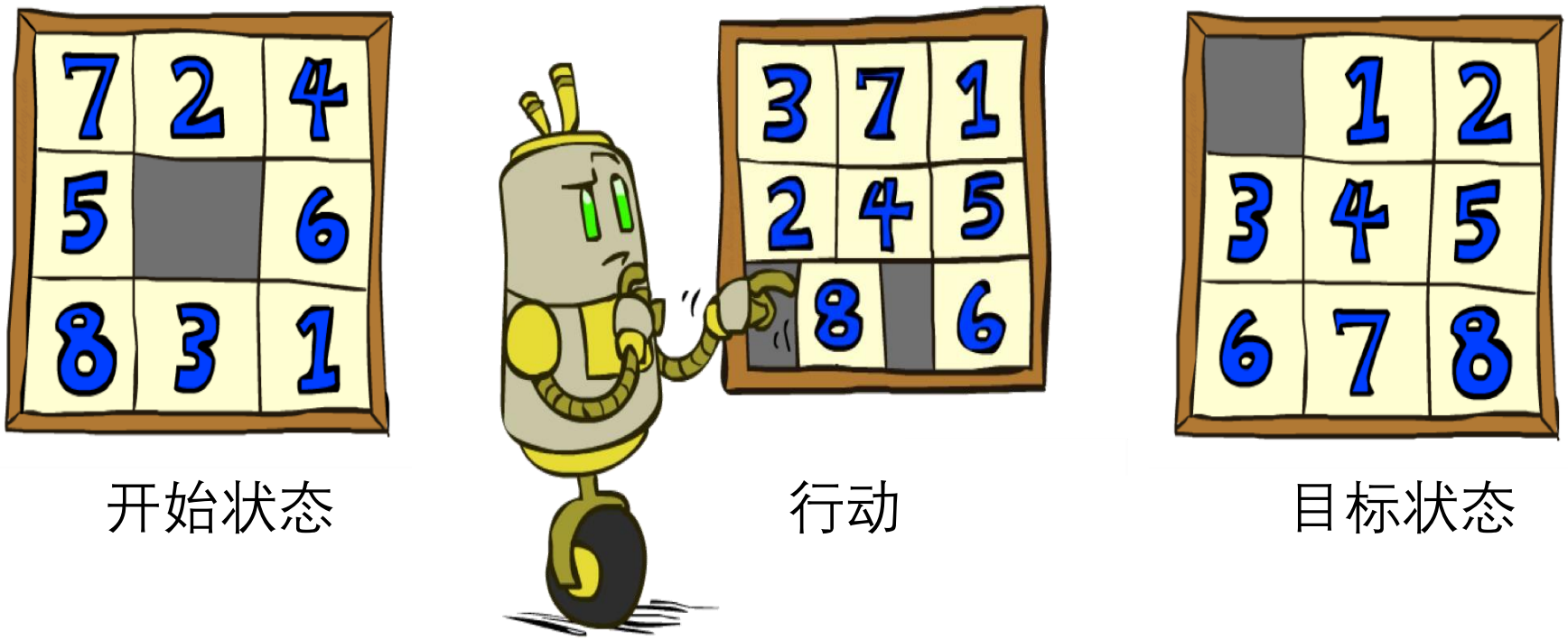
- 在求解很难的搜索问题时，大部分的工作是找到可接纳的启发式函数。
- 可接纳的启发式函数信息，通常是对应的***松弛问题(relaxed problems)的解***，*解除对行动的限制。*

366





举例: 8 数字谜题

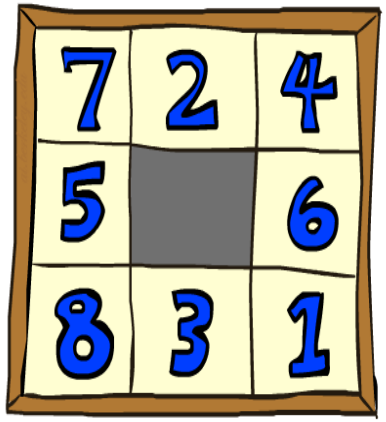


开始状态　　行动　　目标状态

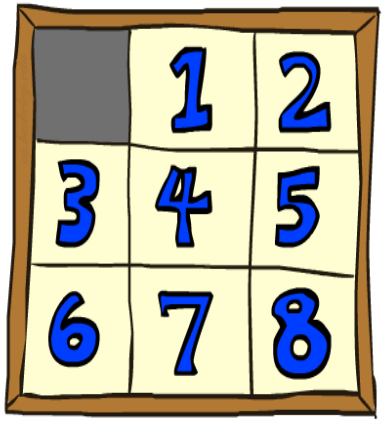
- 状态？总共有多少种状态？
- 行动？开始状态有多少种可选行动
- 行动成本？

8 数字谜题

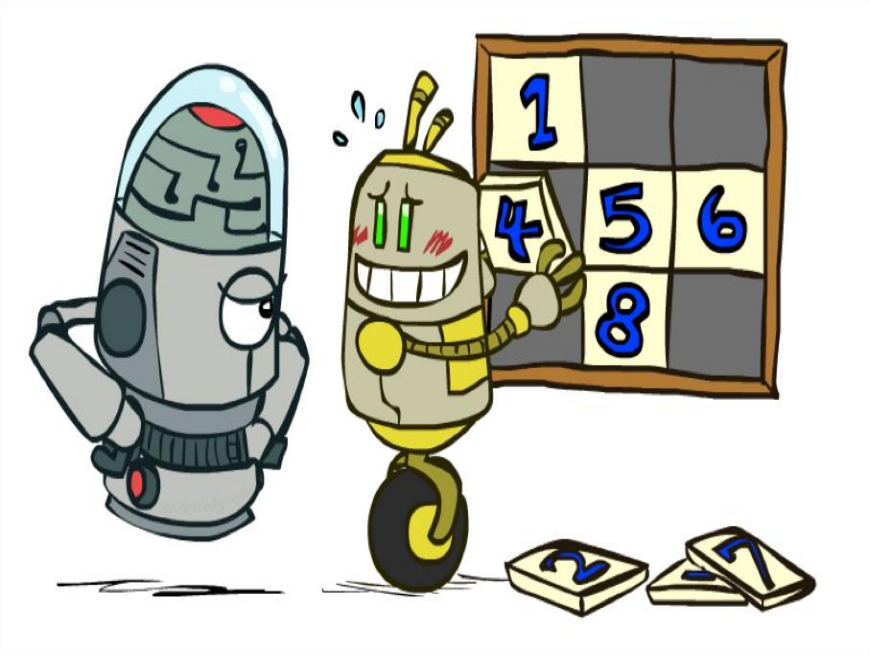
- 启发式: 错位方块的数量
- 为什么是可接纳性的?
- h(开始) = 8
- 松弛问题的启发信息（假设可直接跳到位）



开始



目标



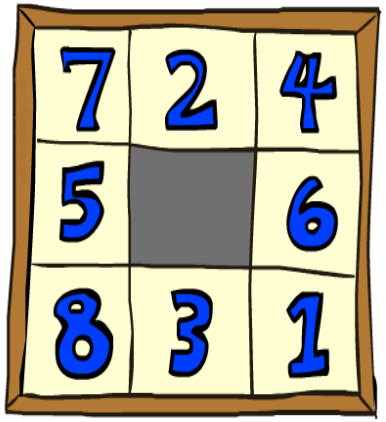
	平均节点扩展数，当最优解的深度是:		
	...4 步	...8	...12
UCS	112	6,300	3.6×10^6
A*TILES	13	39	227

Statistics from Andrew Moore

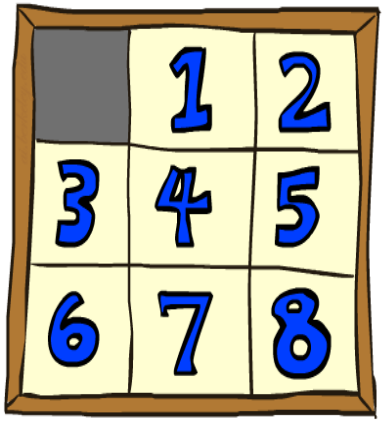
8 数字谜题，继续

- 如果把条件再松弛一下，任何方块可以在任何时候朝任何方向滑动，无论那里有没有其他方块
- 即曼哈顿距离
- 可接纳性的?
- H(开始) =

3 + 1 + 2 + … = 18



开始



目标

	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
A*TILES	13	39	227
A*MANHATTAN	12	25	73

组合启发式（函数）信息

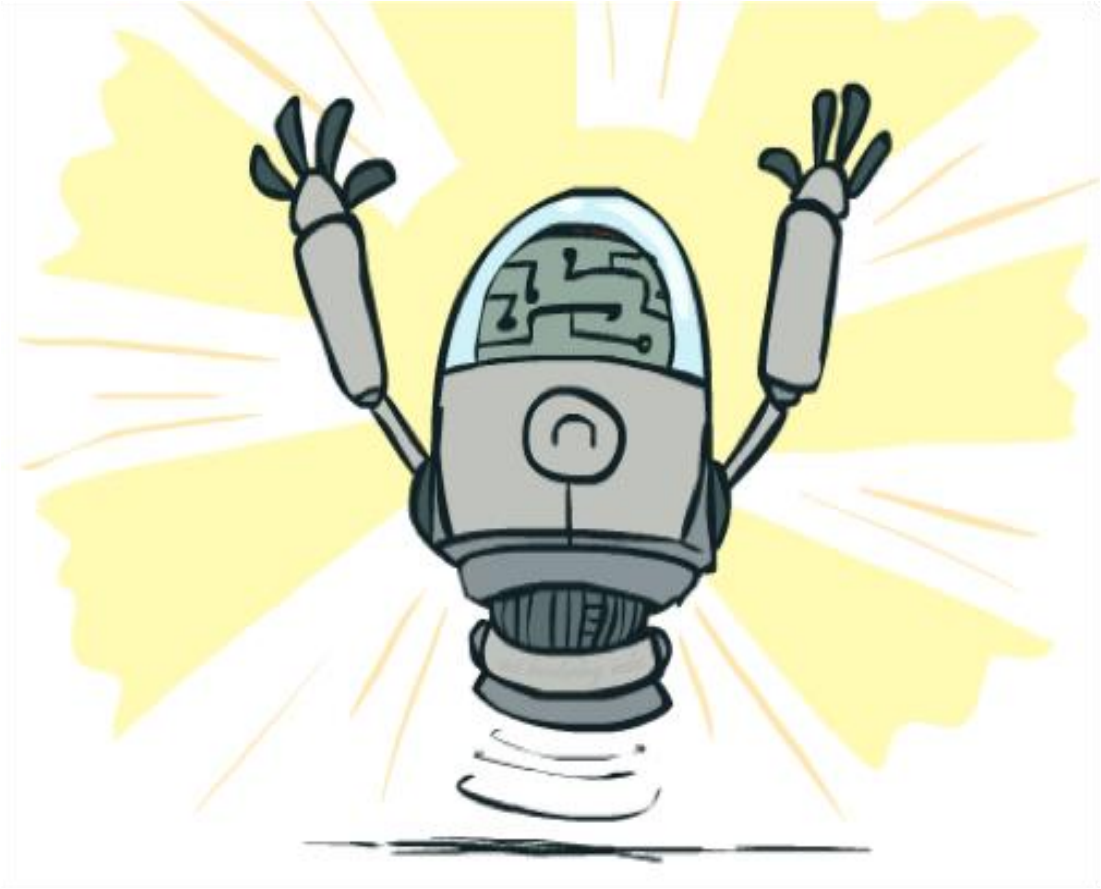
- 支配优势: $h_a \geqslant h_c$ 如果满足:

$$\forall n : h_a(n) \geq h_c(n)$$

  - 一般讲, 越大越好，只要保持可接纳性。
  - H 是 0的话，比较糟，(A* 变成什么，如果 h=0?)。
  - 和真实目标成本相同最好, 但很难达到!

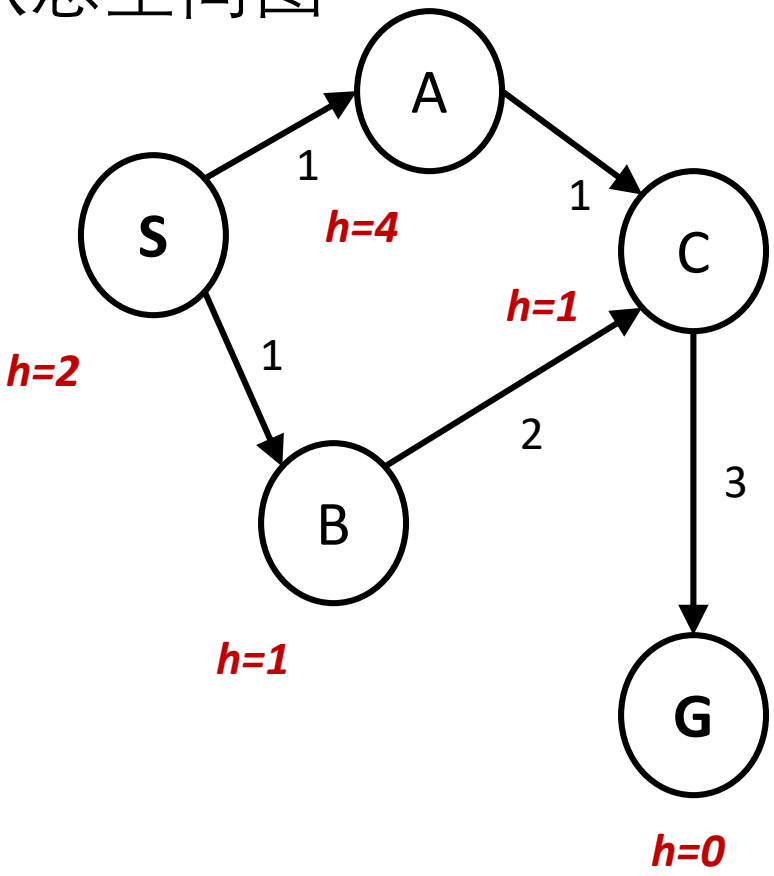
- 如果两个启发式函数，都不支配对方怎么办?
  - 形成一个新的，通过最大式组合: $h(n) = max(h_a(n), h_b(n))$
  - 这个既是可接纳的，也是对之前任一个都有支配优势的，启发式函数。

A* 图搜索算法的 优化性

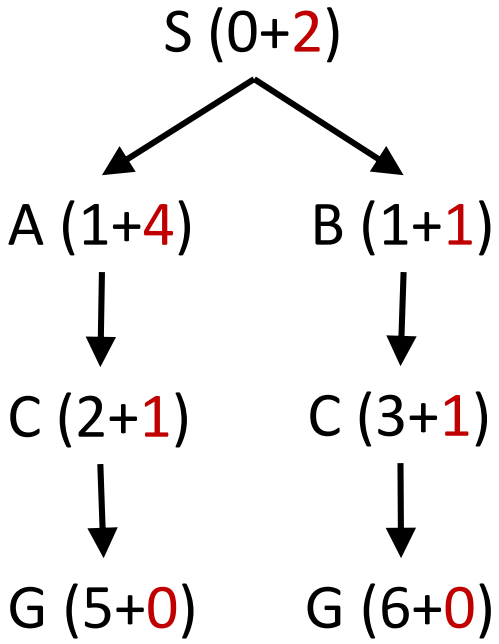


A*图搜索走错了吗?

状态空间图

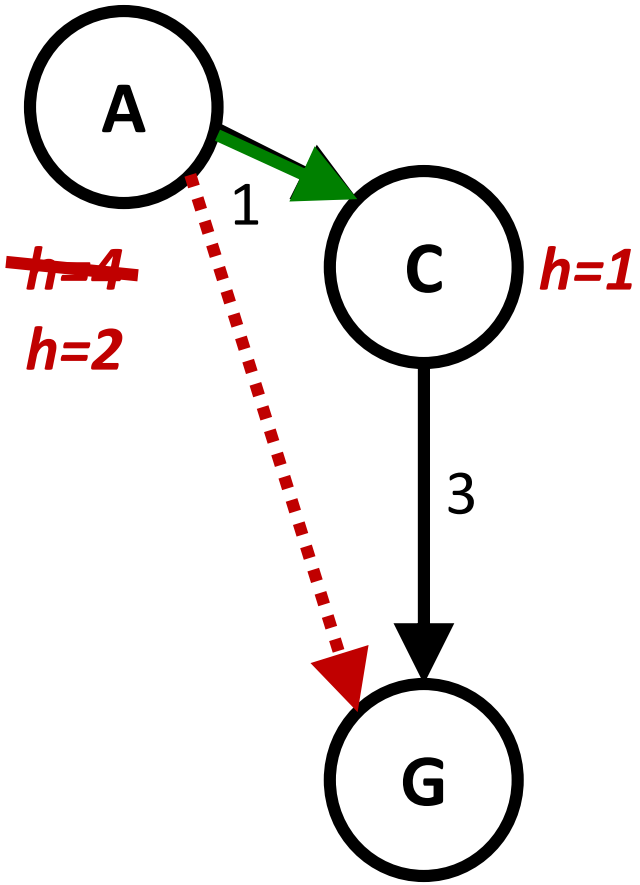


搜索树



C – G 不会被扩展，因为C 已被访问过，因此错过了最优路径。

启发性函数的一致性



- 主体思想: 估计成本 ≤ 实际成本
  - 可接纳性: 启发函数估计成本 ≤ 实际路径成本

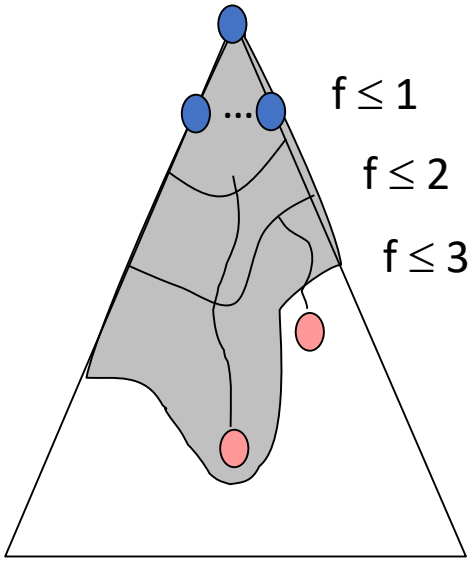
    h(A) ≤ 从 A 到 G 的实际路径成本
  - 一致性: 启发函数估计的步骤成本（弧成本） ≤ 实际步骤成本

    h(A) – h(C) ≤ cost(A 到 C)

- 一致性的结果:
  - f 值 在同一路径搜索中不会减少（递增的）
    - h(A) ≤ cost(A 到 C) + h(C)
  - A* 图搜索是最优的（找到的解是最优的）

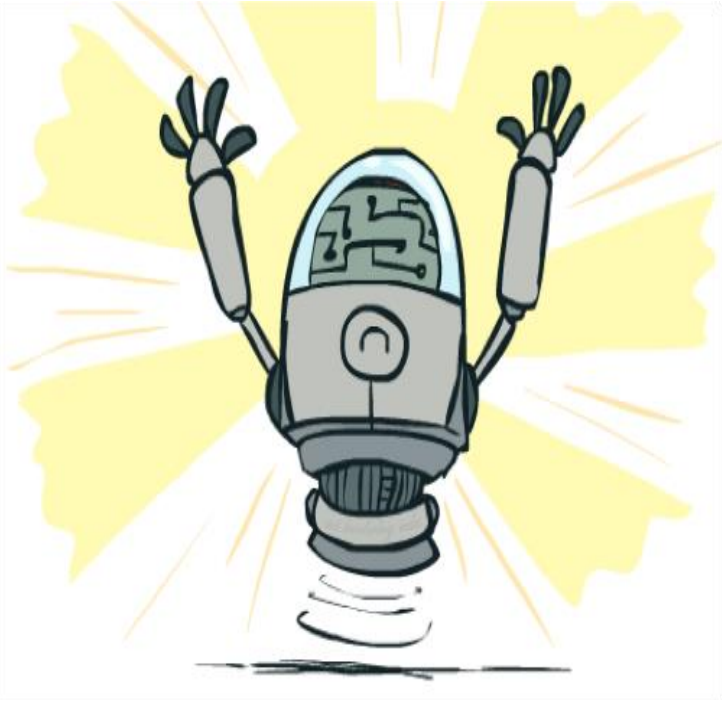
A* 图搜索策略的最优性（Optimality）

- 如果 A* 和一个一致性的启发式函数，那么：
  - 事实 1: A* 扩展节点过程中，f 值递增 (f-轮廓)
  - 事实 2: 对于每个状态 s, 到达 s 的最优路径上的节点先于到达 s的次优路径节点被扩展
  - 结果: A* 图搜索是最优的



最优性

- 树搜索:
  - 如果启发式函数是满足可接纳性的， A* 具有最优性。
  - 基于成本的统一搜索(UCS) 是一个特例 (h = 0)。
- 图搜索:
  - 如果启发式函数是满足一致性的， A* 最优。
  - UCS 最优 (h = 0 也是一致性的)
- 一致性蕴涵了可接纳性
- 通常, 从条件松弛问题中得到的启发式信息趋向于满足一致性条件。



A*: 总结

A*: 总结

- A* 既使用了来程（已走过的）路径成本，又利用了估计的前程（将要走的）路径成本
- A* 是最优的，如果伴随使用可接纳性的或一致性的启发式函数
- 启发式函数的设计是关键: 通常运用条件松弛问题的解