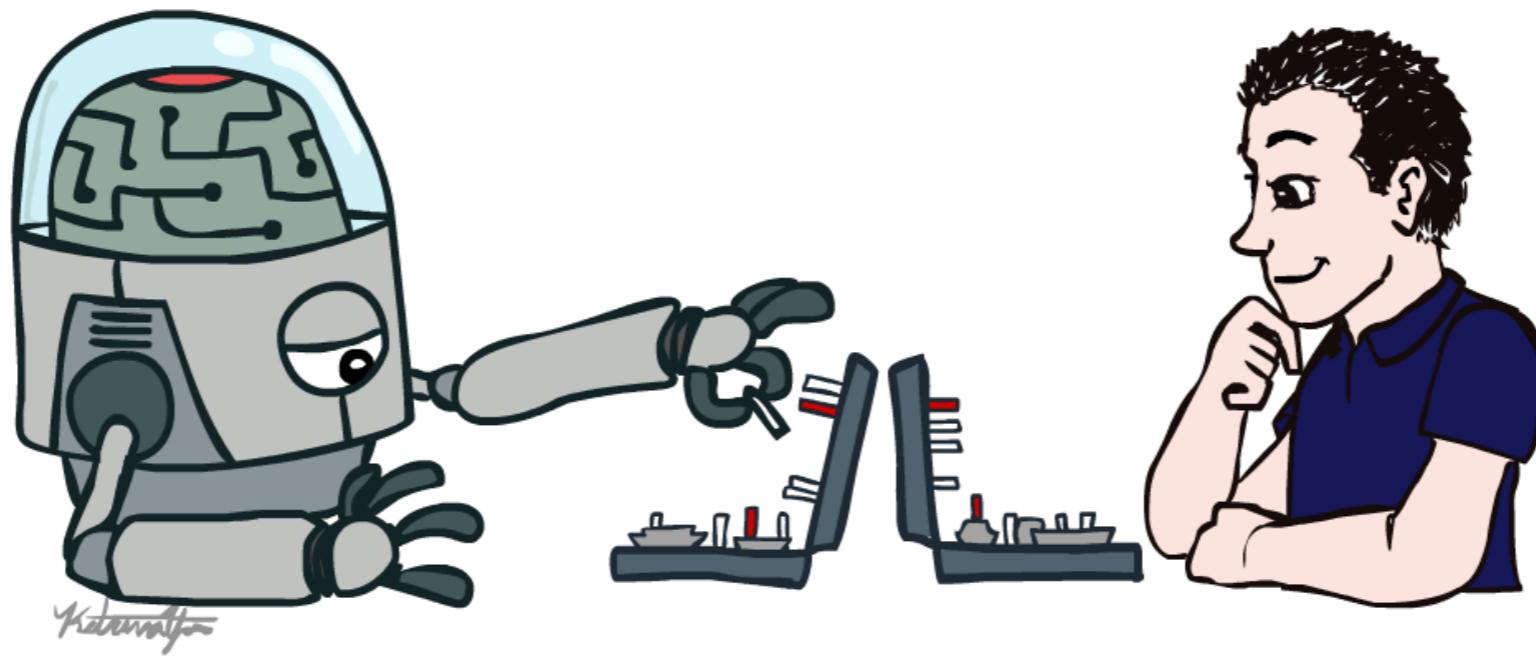
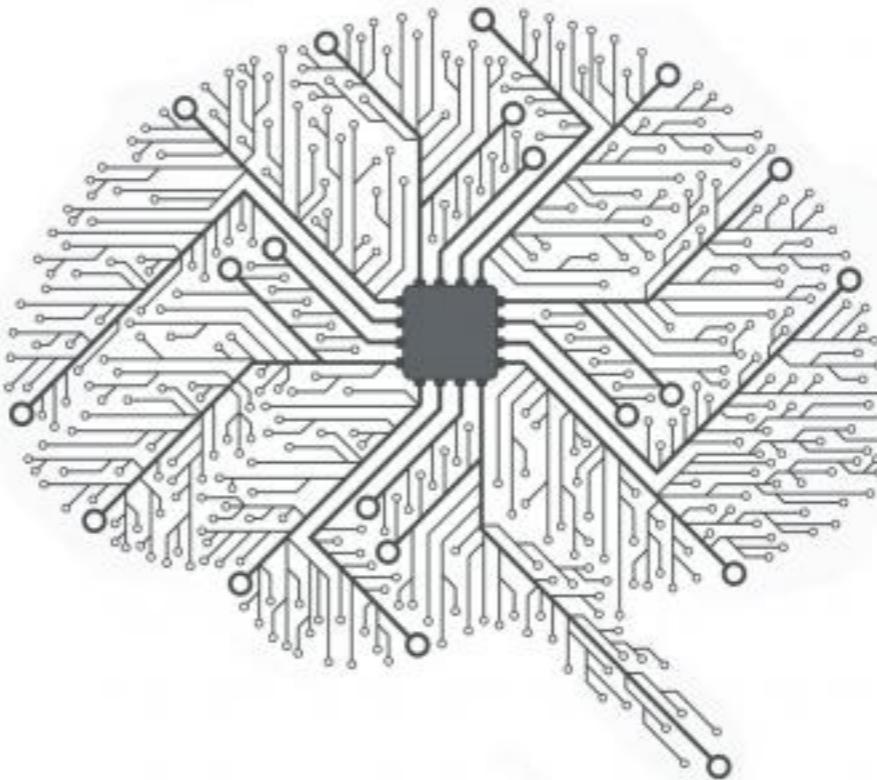


人工智能



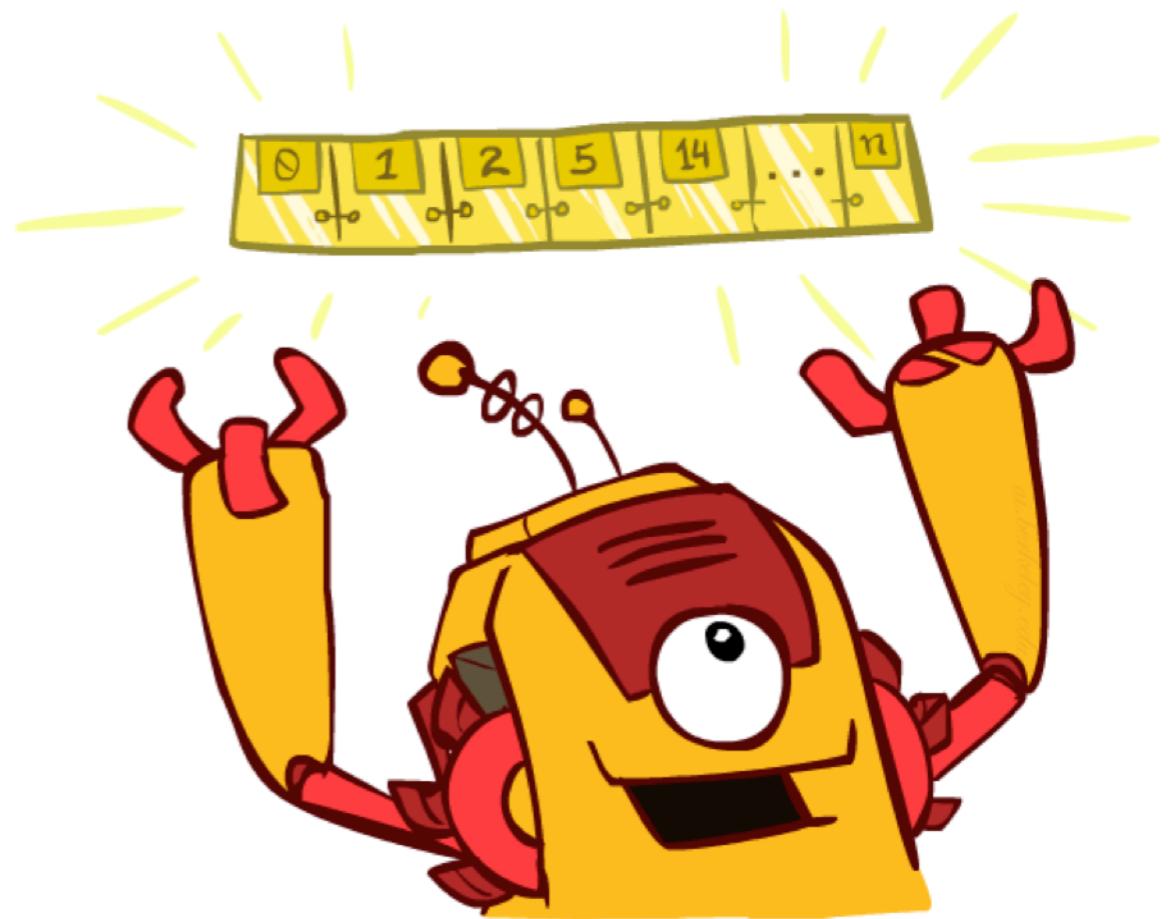
第三章·有信息的搜索算法

- 搜索启发法
- 贪心搜索
- A star 搜索
- 可接纳的启发式函数



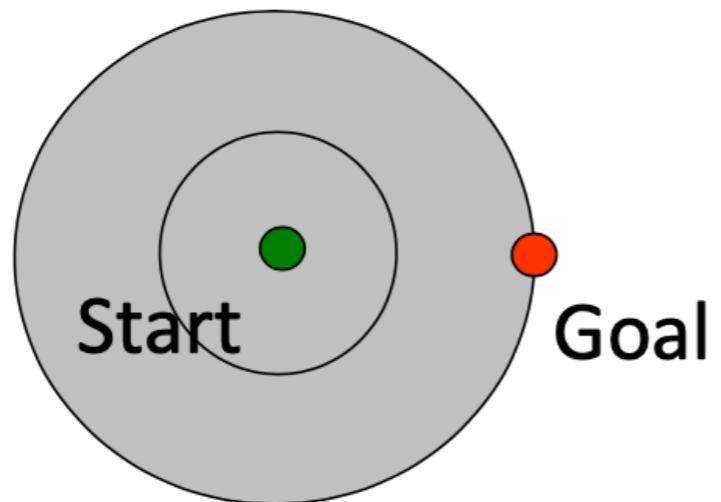
都是一个队列

- 除了边缘策略，所有这些搜索算法是相同的
 - 从概念上讲，所有的边缘都是优先队列(即集合与附加的节点优先级)
 - 实际上,DFS和BFS可以用栈和队列，避免 $\log(n)$ 的排序开销
 - 核心问题是**如何排序**

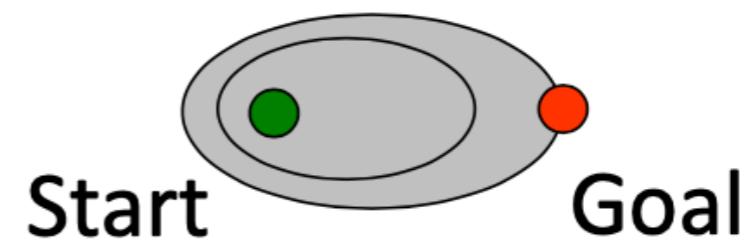


启发式搜索

- 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



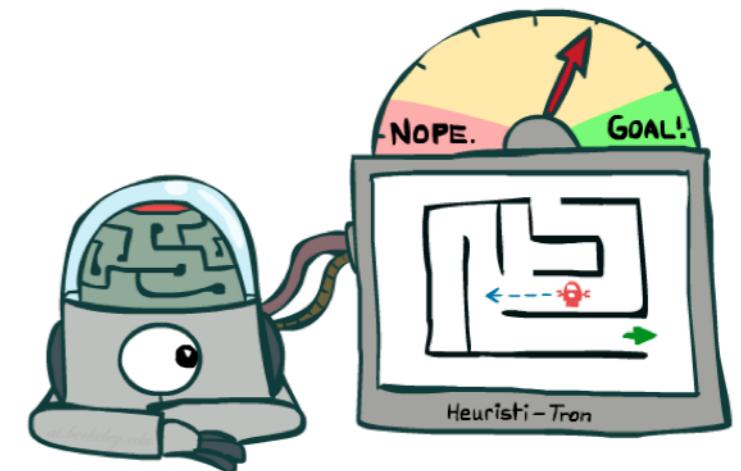
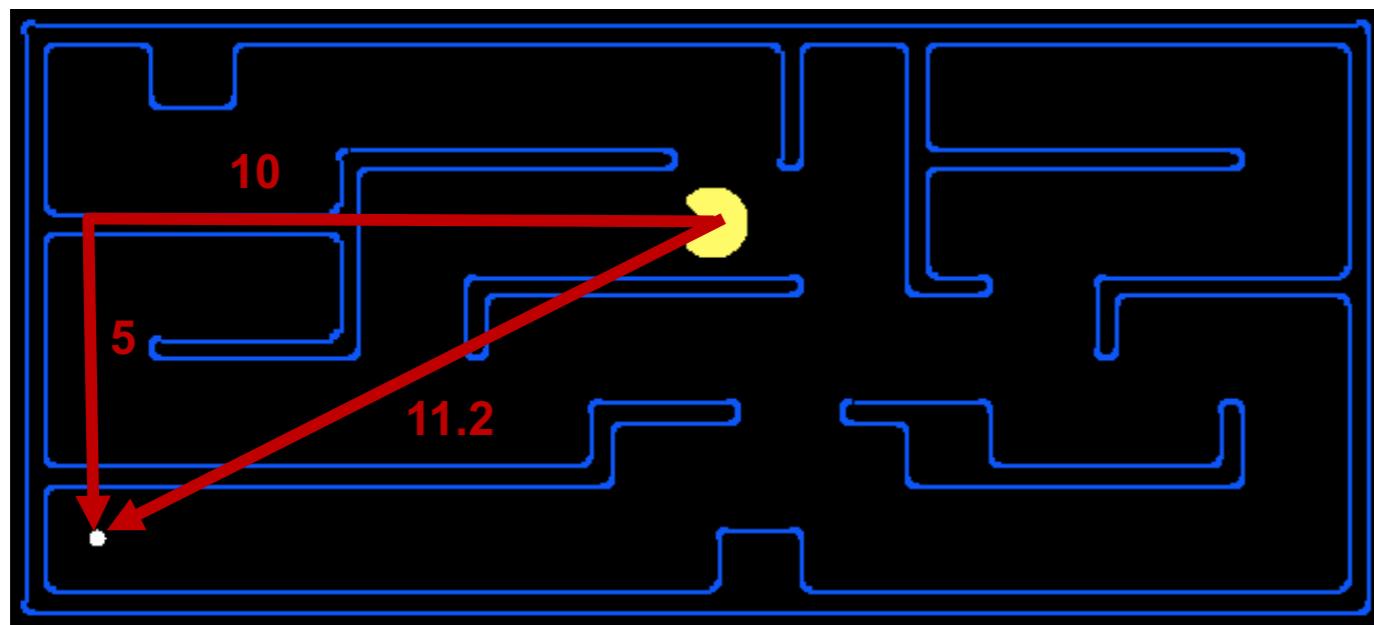
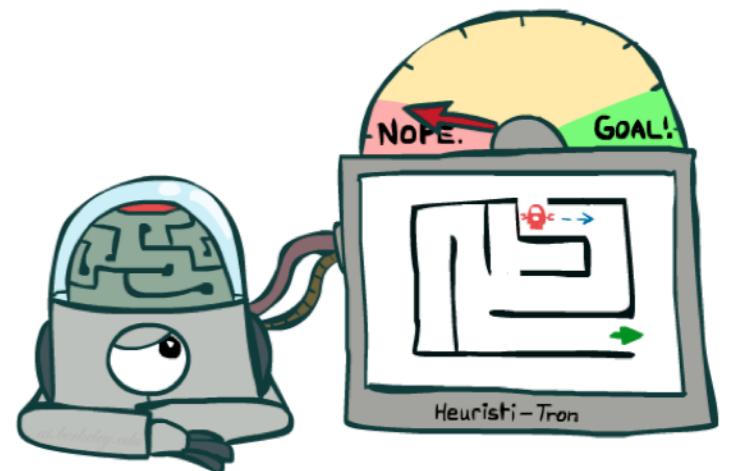
基础搜索(无信息启发)



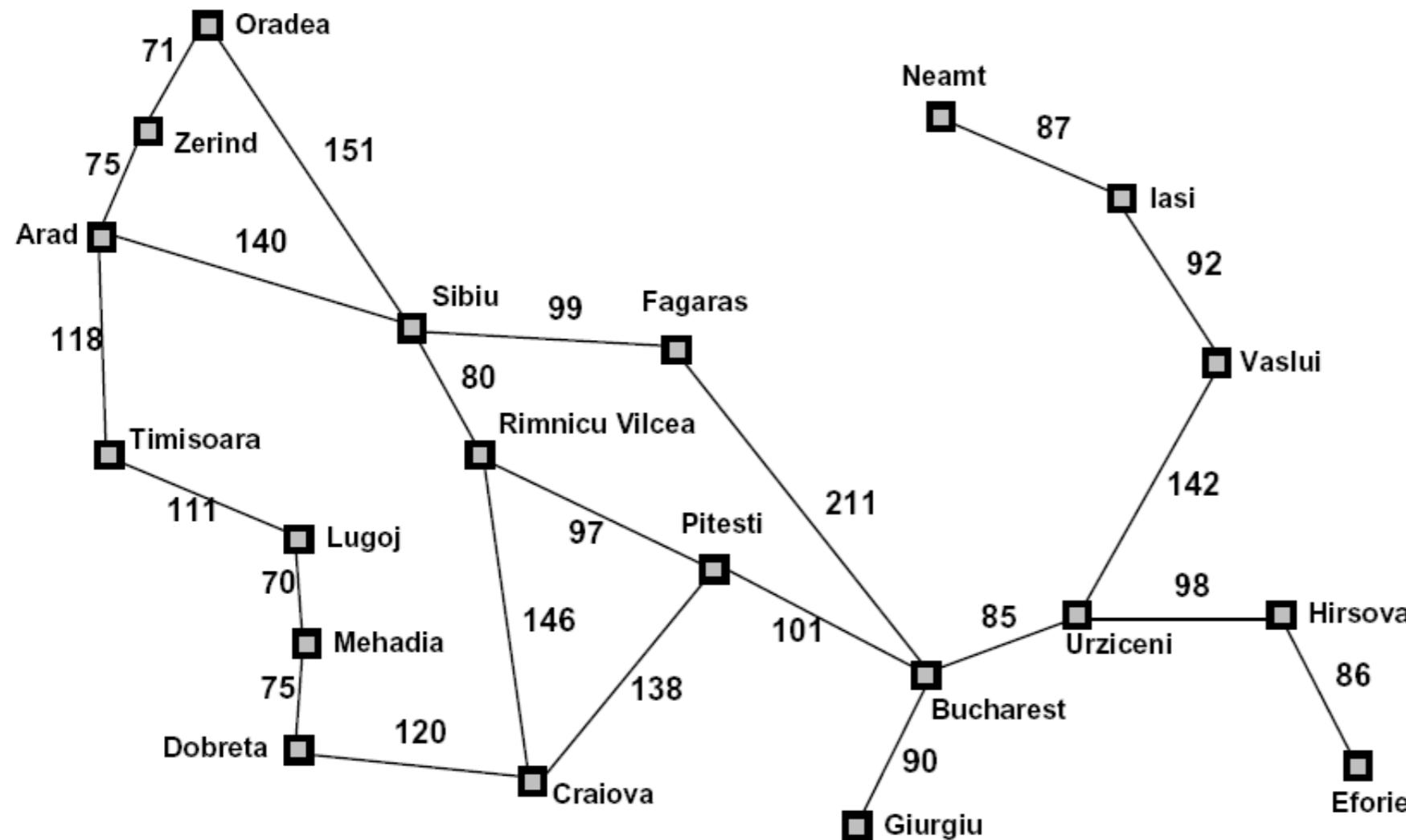
信息启发的

启发函数(Heuristics)

- Heuristic :
 - 一个用以估计一个状态有多接近一个目标的函数
 - 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



例子：启发式函数



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

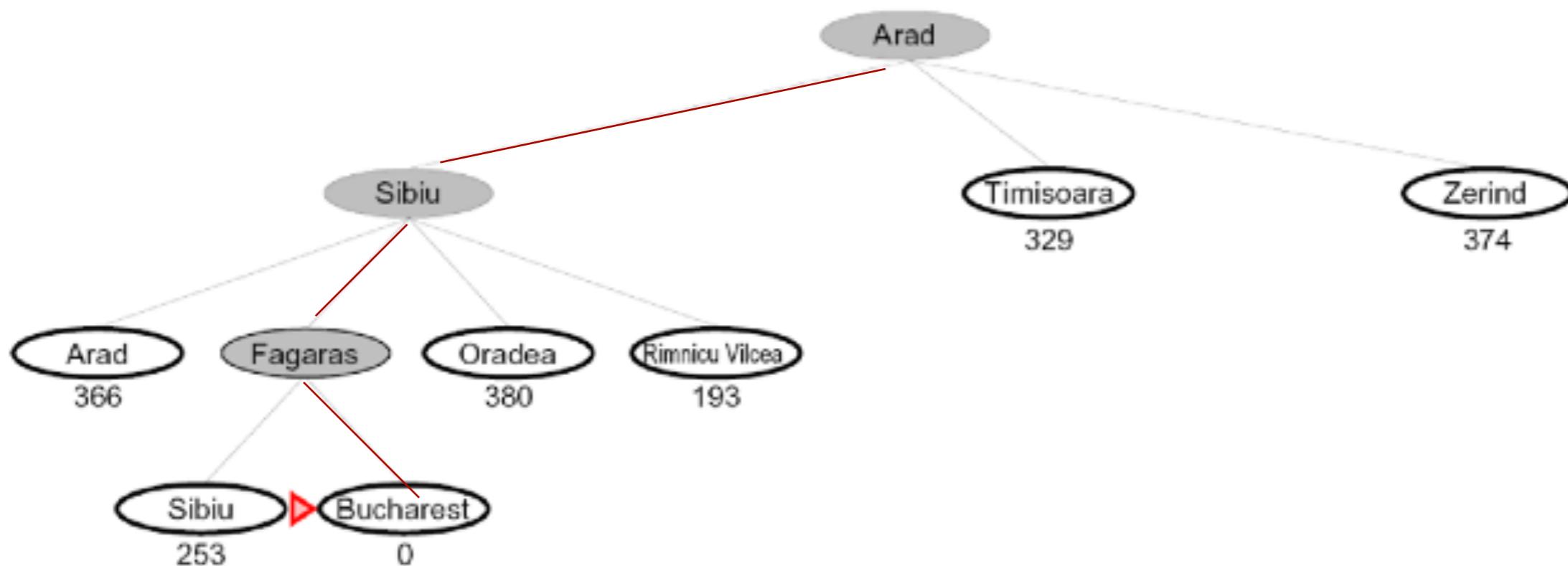
$h(x)$

贪心搜索 (Greedy Search)



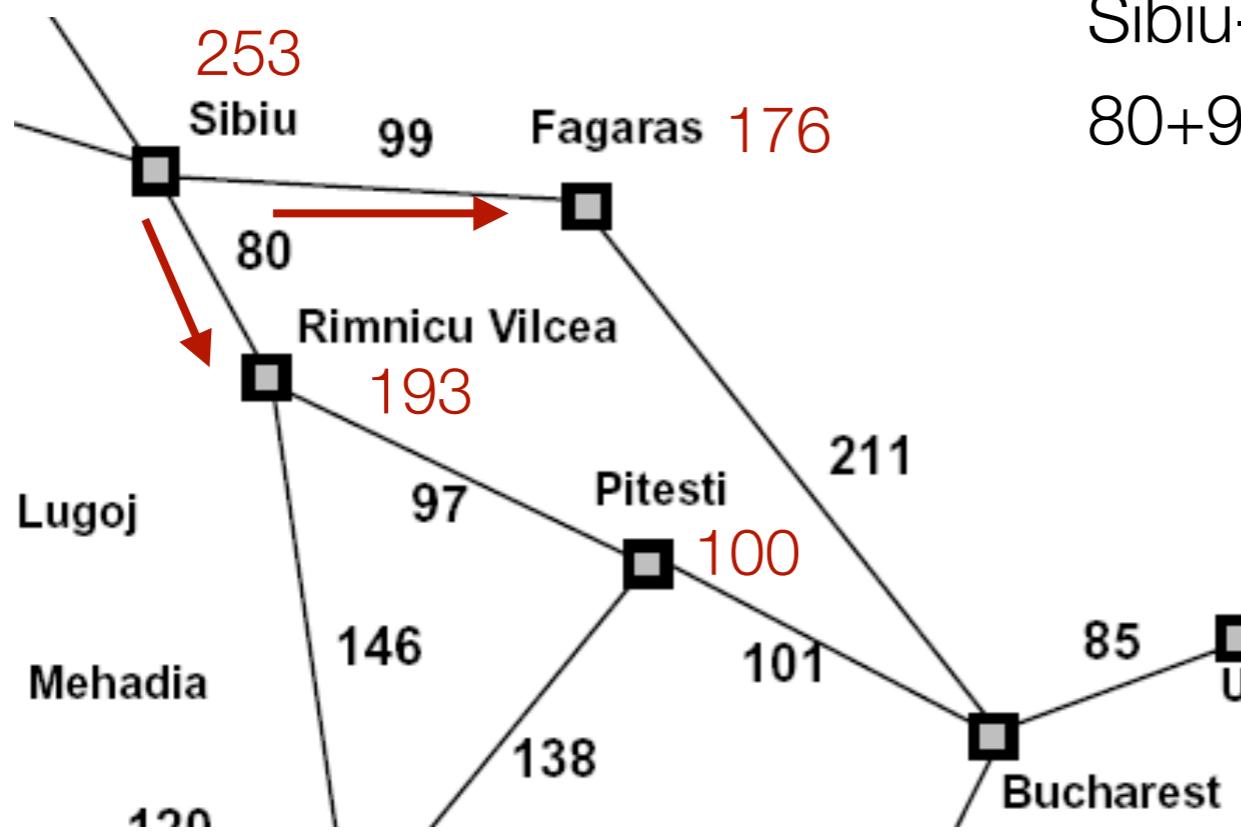
贪心搜索

- 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)



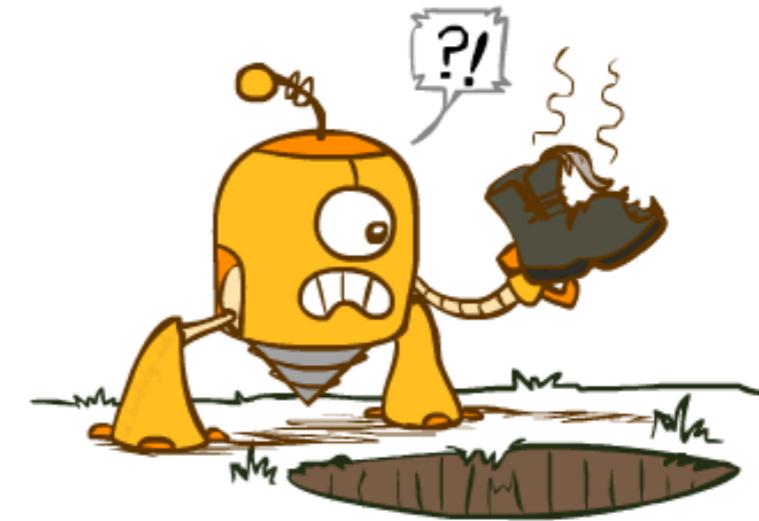
贪心搜索

- 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?



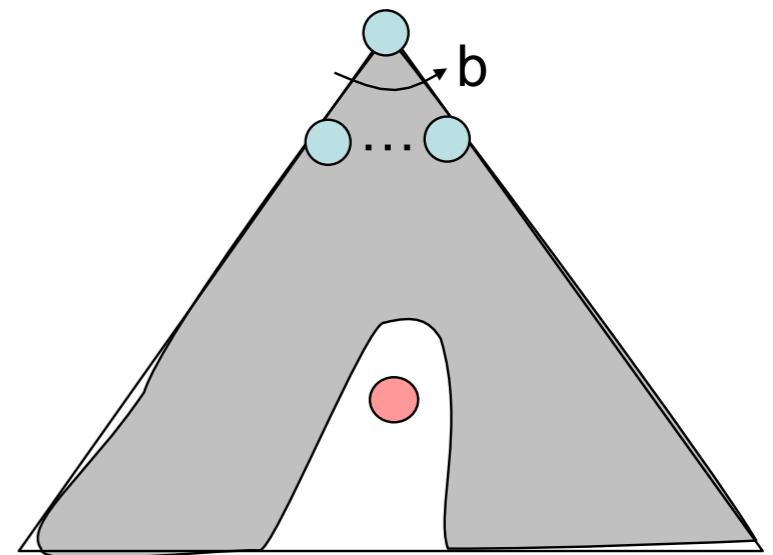
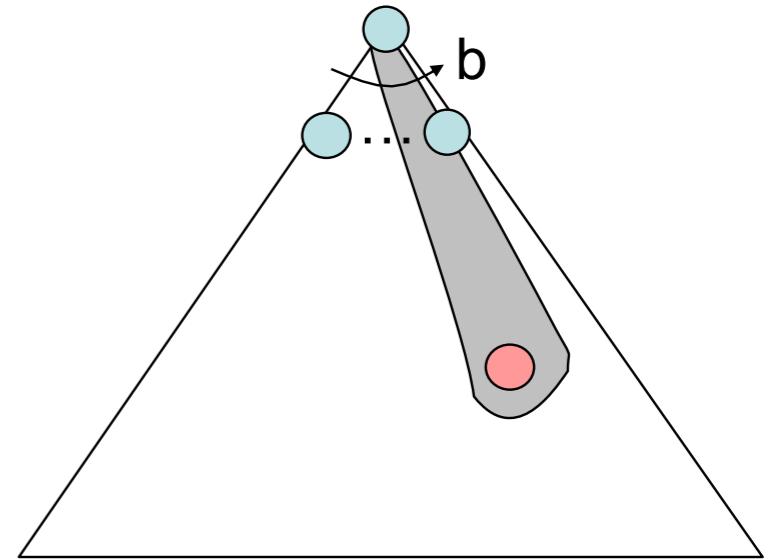
$$\text{Sibiu-Fagaras-Bucharest} = 99 + 211 = 310$$

$$\text{Sibiu-Rimnicu Vilcea-Pitesti-Bucharest} \\ 80 + 97 + 101 = 278$$

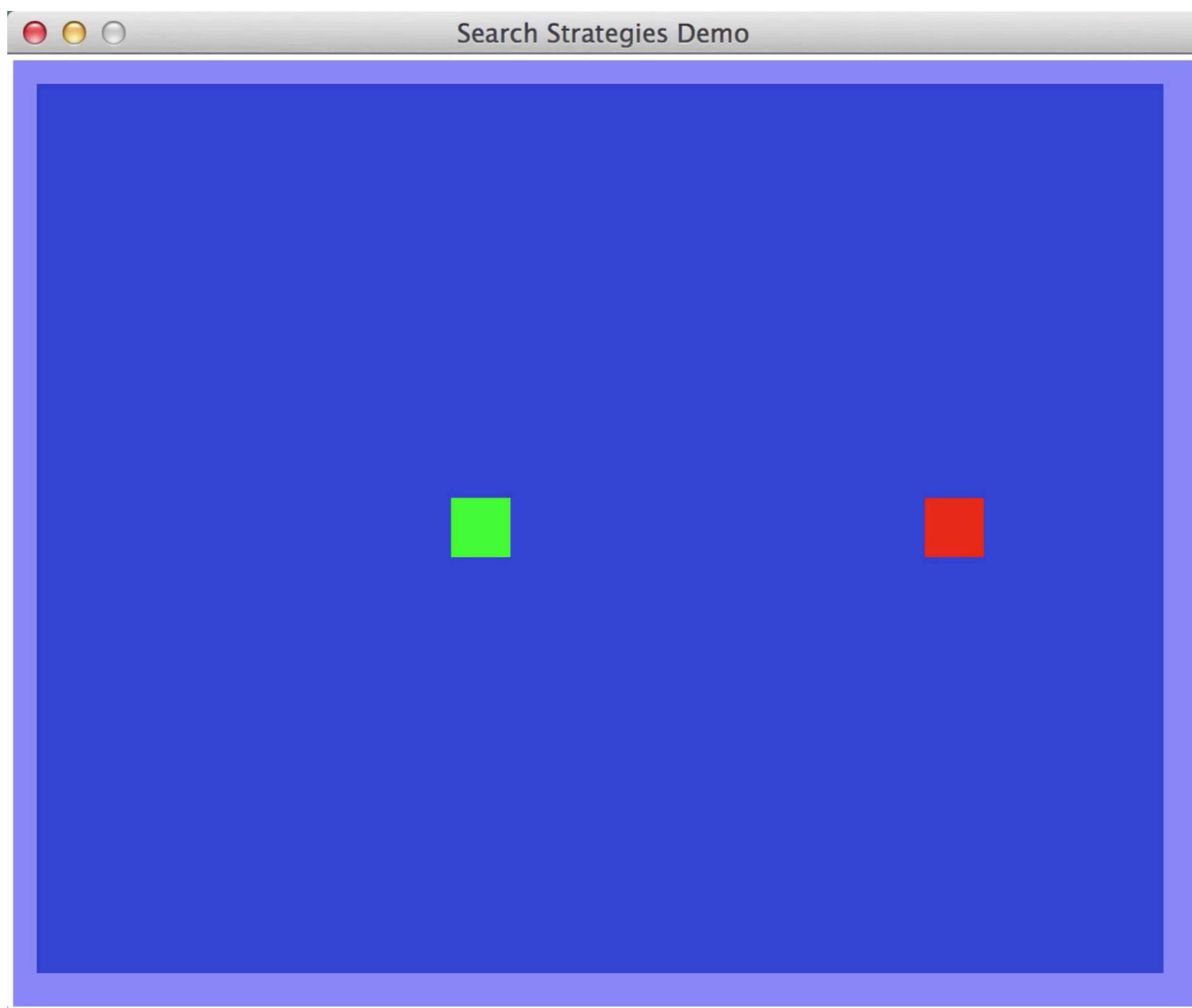


贪心搜索

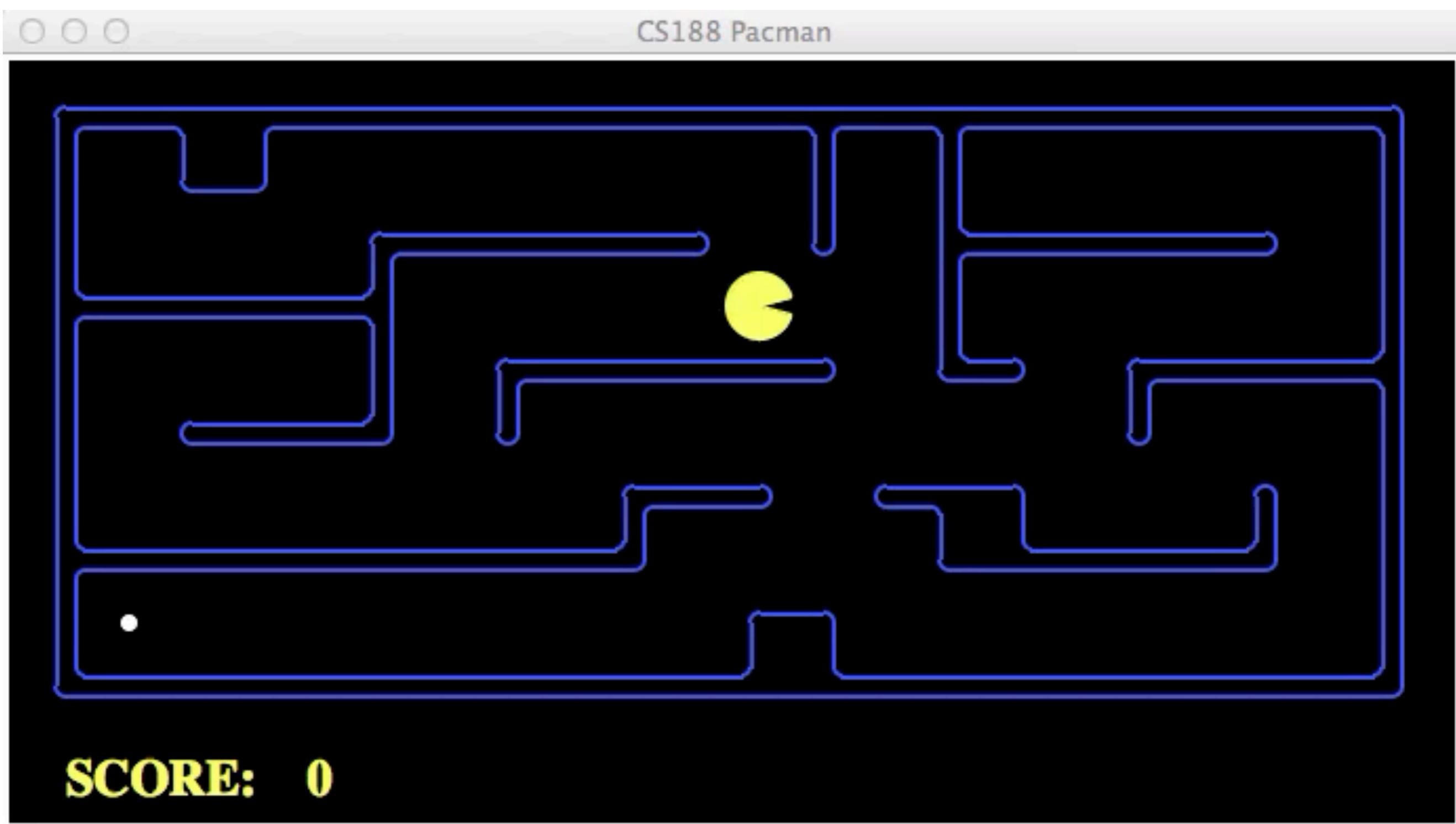
- 策略：根据 h 值，扩展一个似乎离目标节点最近的节点，
 - 问题1：没有考虑行动(步骤)成本，导致选择的路径可能长而蜿蜒
 - 问题2：依赖于 h 值，可是它也有可能完全是错的



贪婪算法(空环境里)



贪婪算法(Pacman小迷宫)



A* 搜索





基于成本的统一
搜索(UCS)



贪心搜索(Greedy)



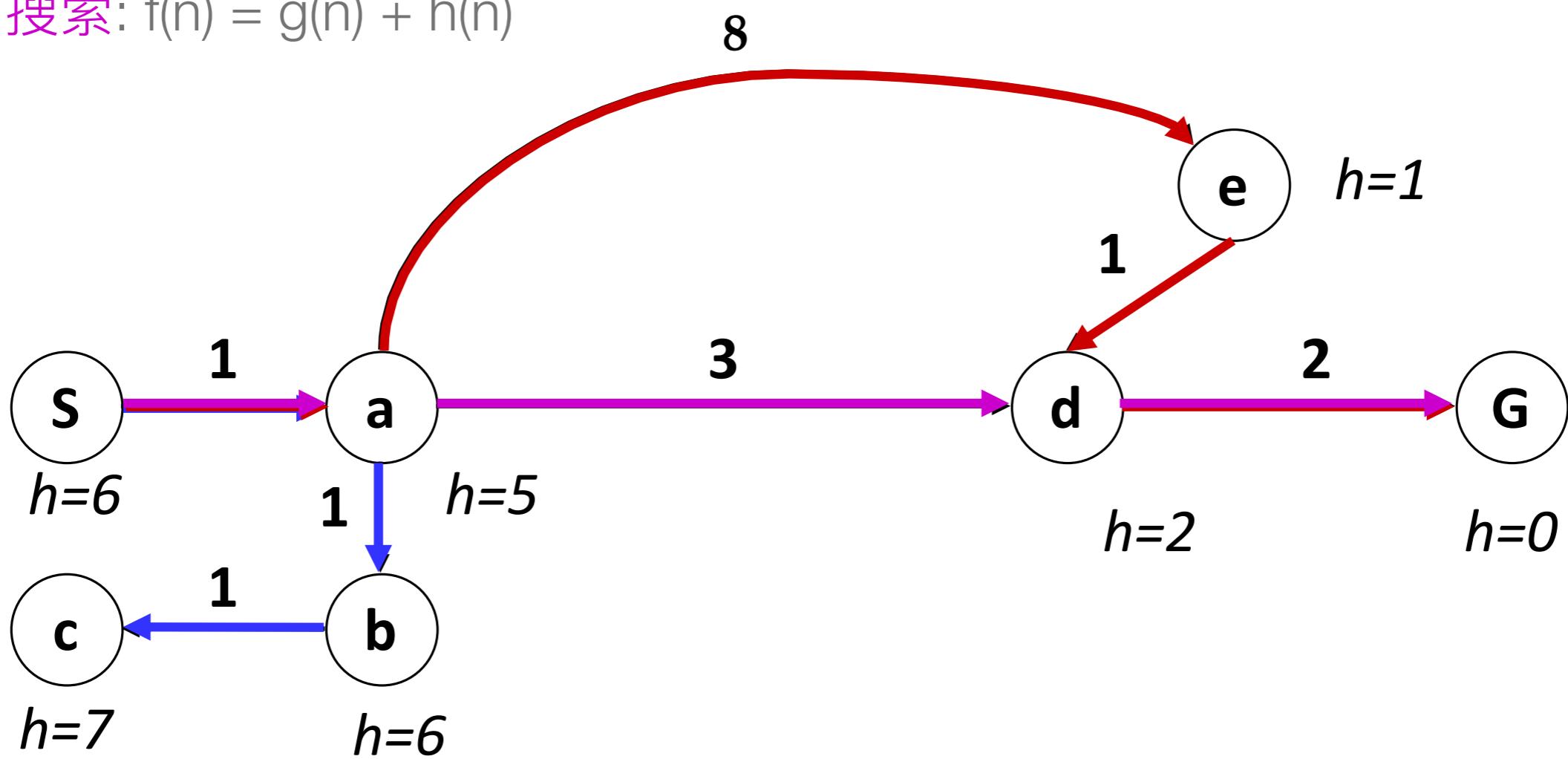
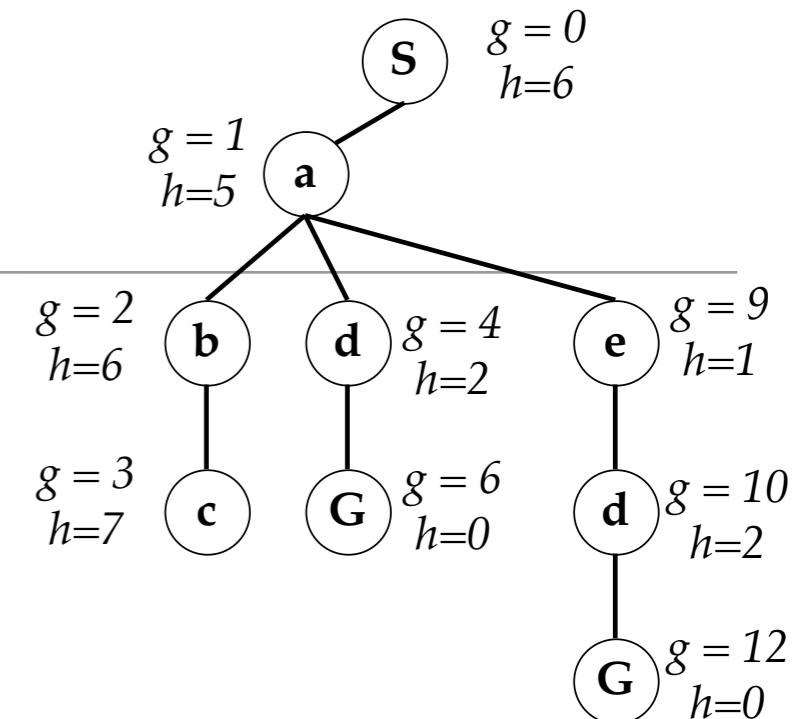
A star

结合统一搜索和贪婪搜索

- UCS: 把路径成本排序, 即来程的成本 $g(n)$

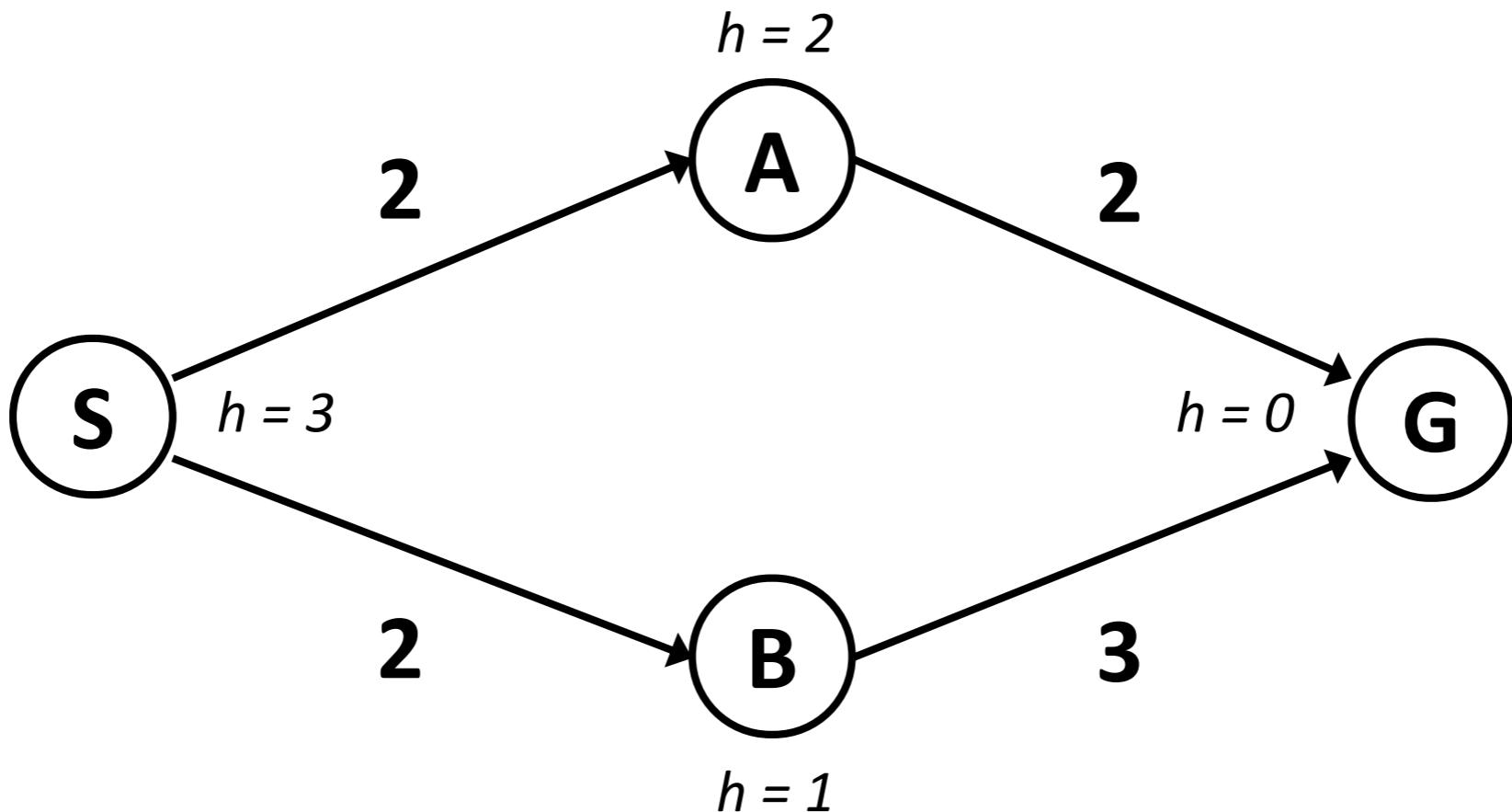
- Greedy: 排序按照与目标的临近性, 即前程成本 $h(n)$

- A* 搜索: $f(n) = g(n) + h(n)$



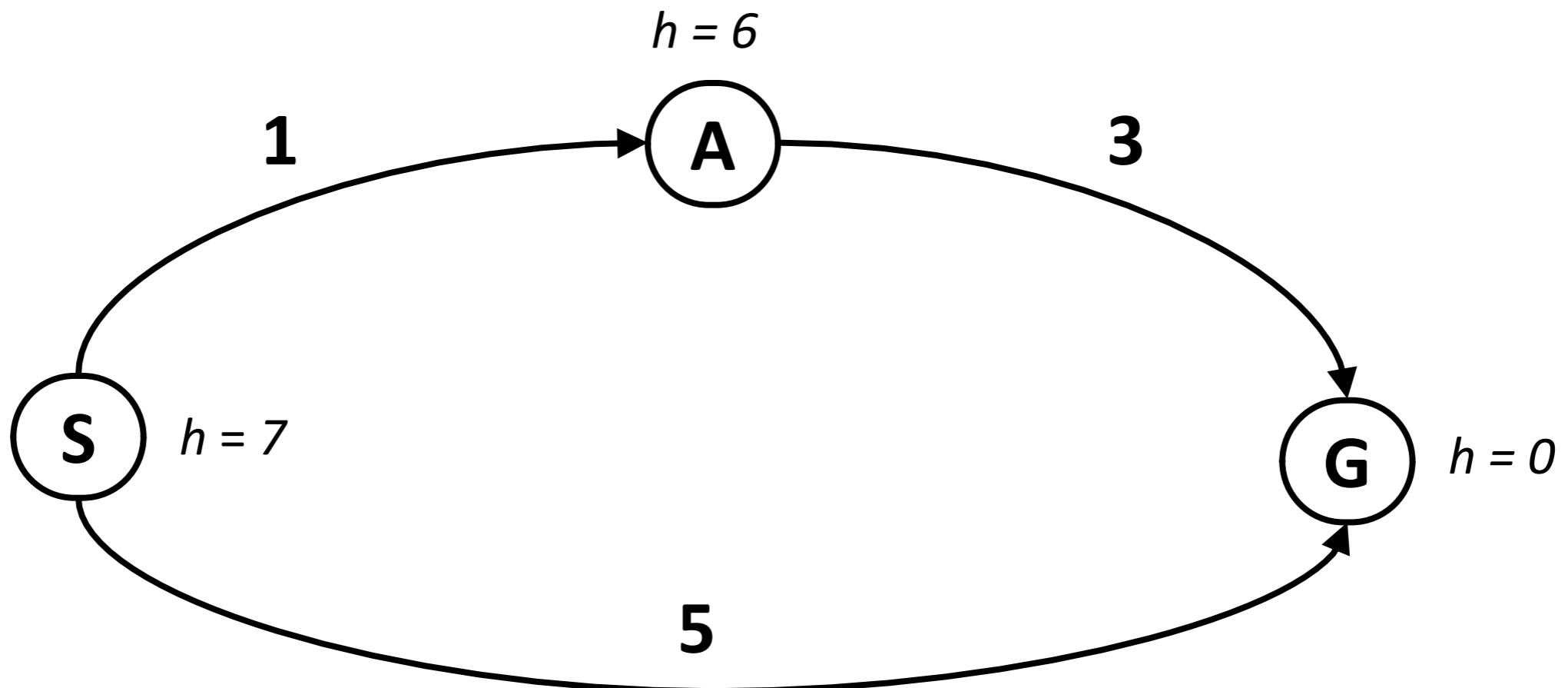
A*应该何时终止？

- 当目标入队列时，应该停止吗？



- 不，只有当目标出队列时，才应该停止。

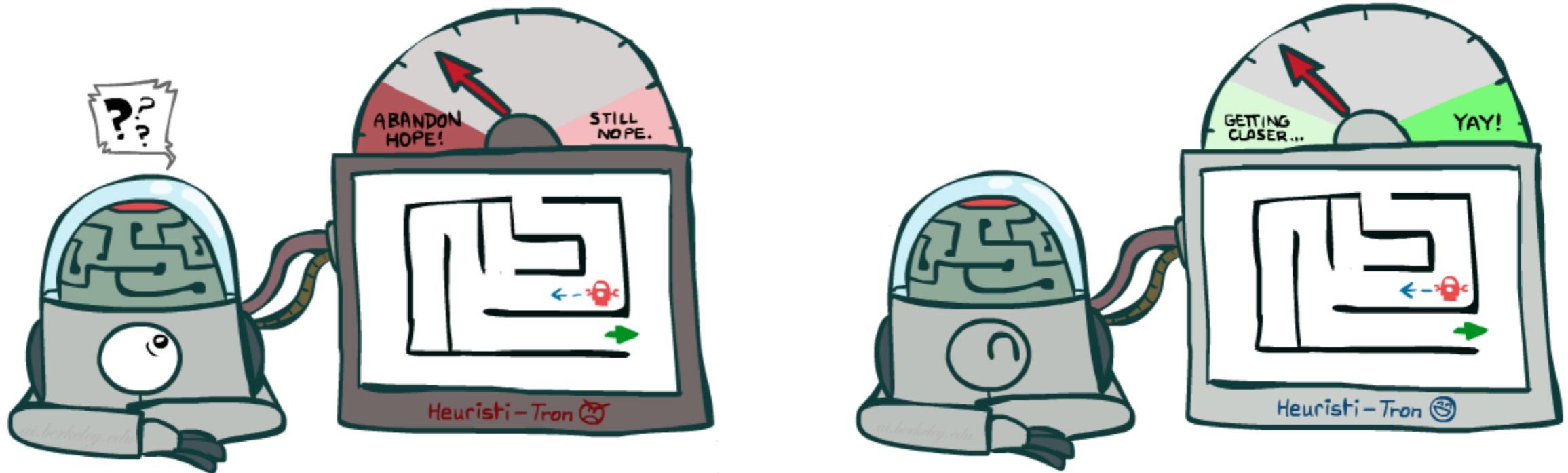
A* 是最优的吗?



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

可接受的启发式函数 (Admissible Heuristics)



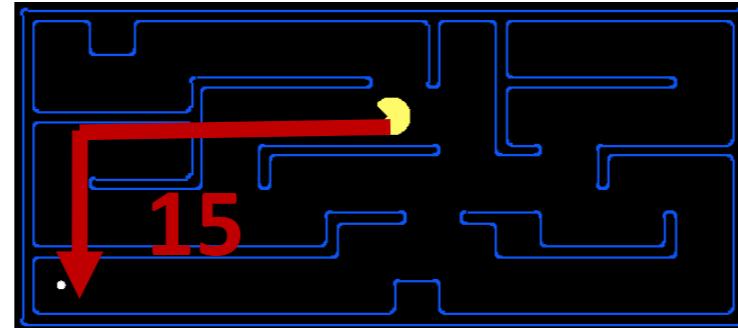
不可接受的(悲观的)启发式函数通过在边缘错误估计好的计划来破坏最优性

可接受的(乐观的)启发式函数能减缓糟糕的计划，但绝不会超过真正的成本

可接受的启发式函数

- h 是可接纳的 (乐观的, 想的比实际好) :

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

- 其中 $h^*(n)$ 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。

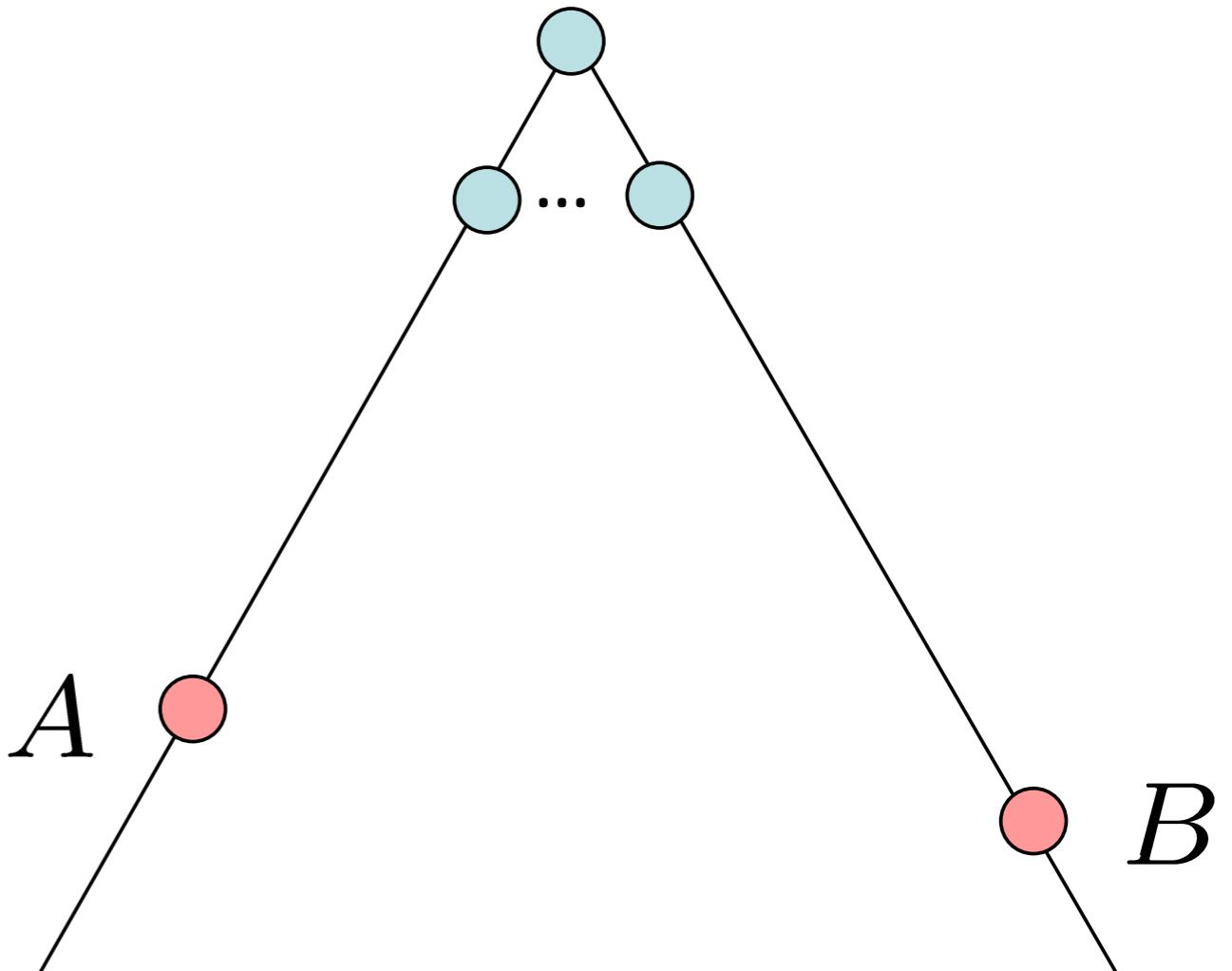
A* 搜索的最优性

- 假定:

- A :一个最优目标节点
- B :一个次优目标节点
- h 是可接受的

- 如果最优性成立, 那么:

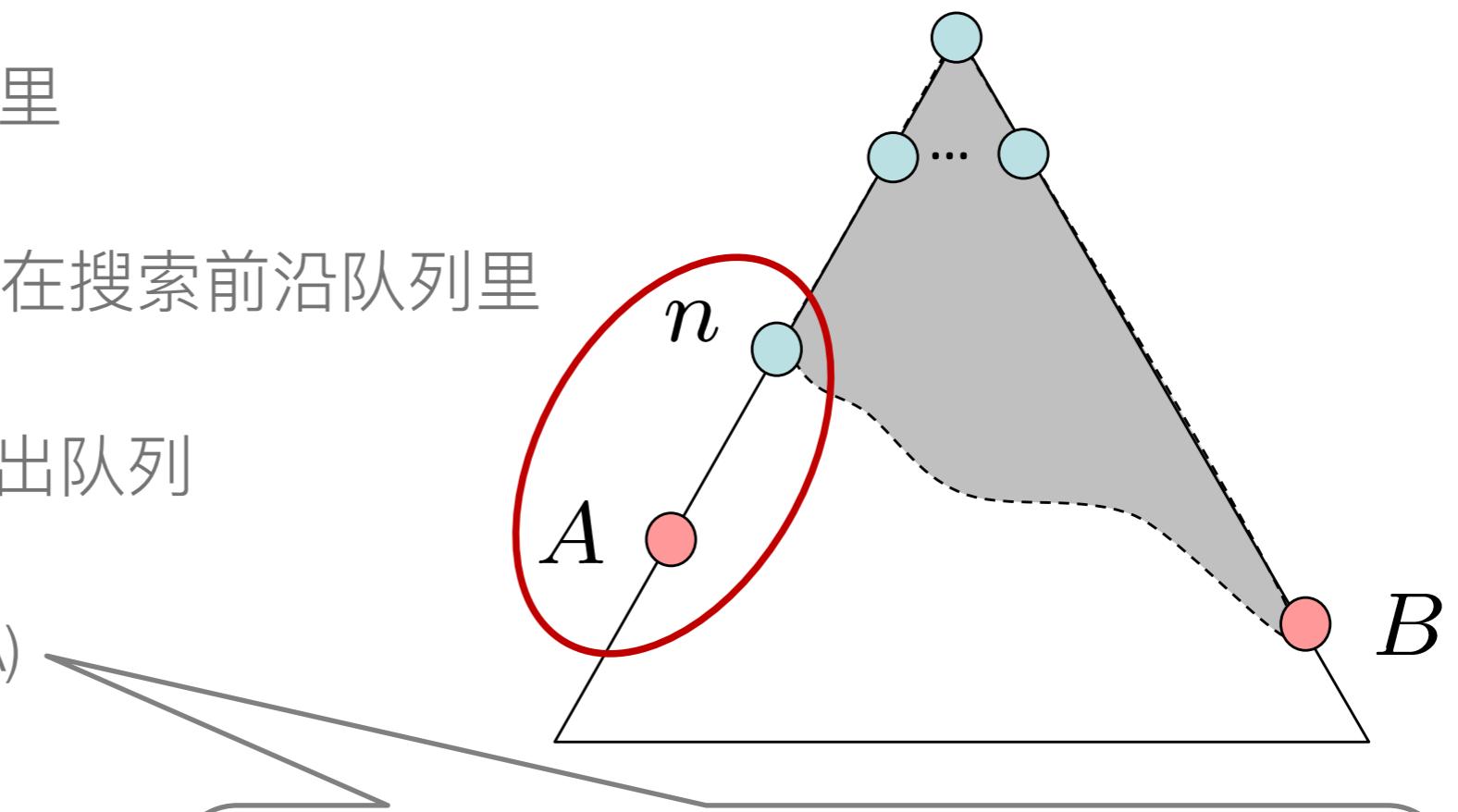
- A 将会在B之前被选择先出队列



A* 树搜索的最优性

- A* 树搜索的最优性 证明:

- 假设B在搜索前沿队列里
- A的某个祖先节点 n 也在搜索前沿队列里
- 首先证明: n 会先于 B 出队列
 - 1. $f(n)$ 小于或等于 $f(A)$



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

$$f(n) \leq g(A)$$

$$g(A) = f(A)$$

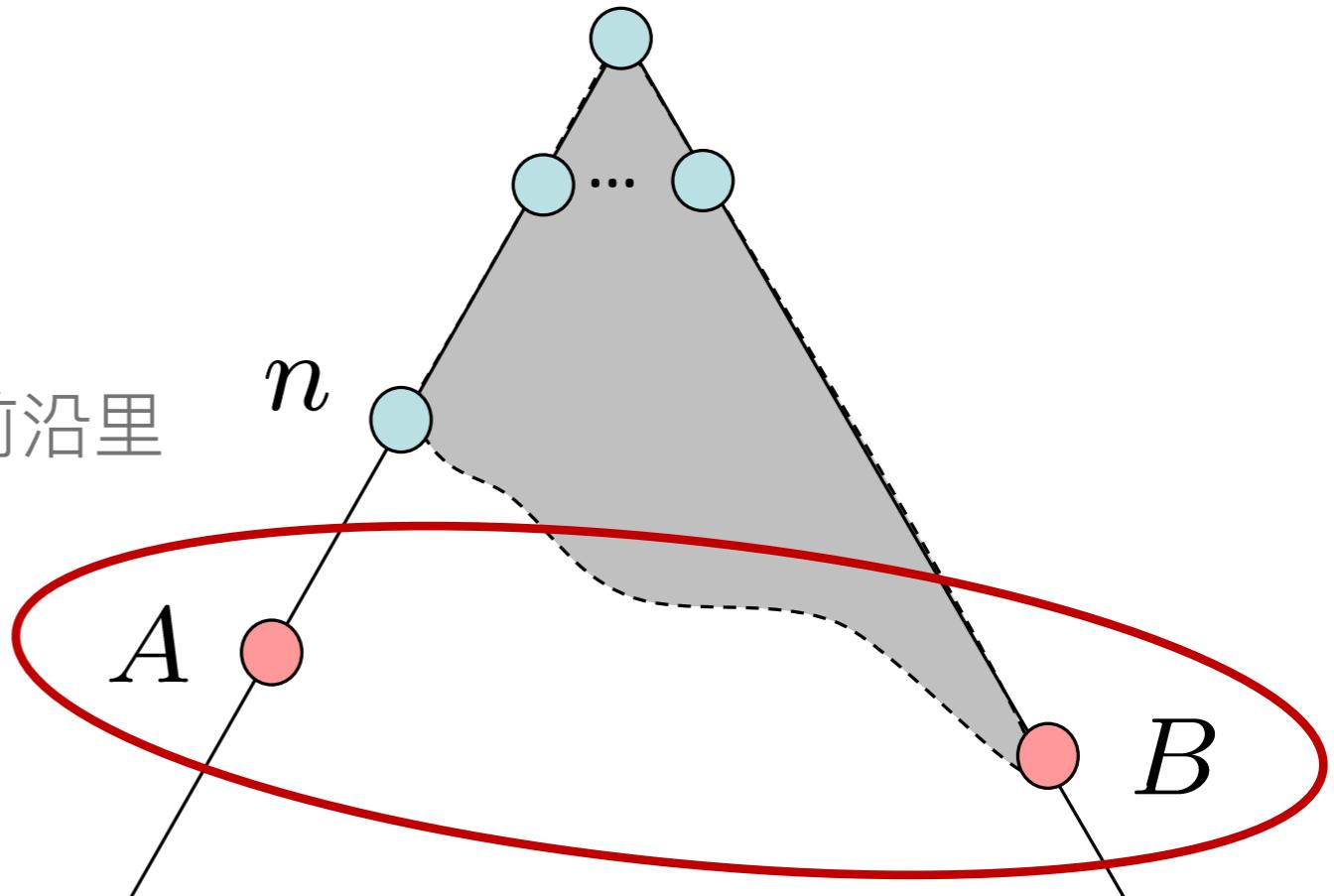
Definition of f-cost

Admissibility of h

$h = 0$ at a goal

A* 树搜索的最优性

- 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
 - 首先证明: n 会先于 B 出队列
 - 1. $f(n)$ 小于或等于 $f(A)$
 - 2. $f(A)$ 小于 $f(B)$



$$g(A) < g(B)$$

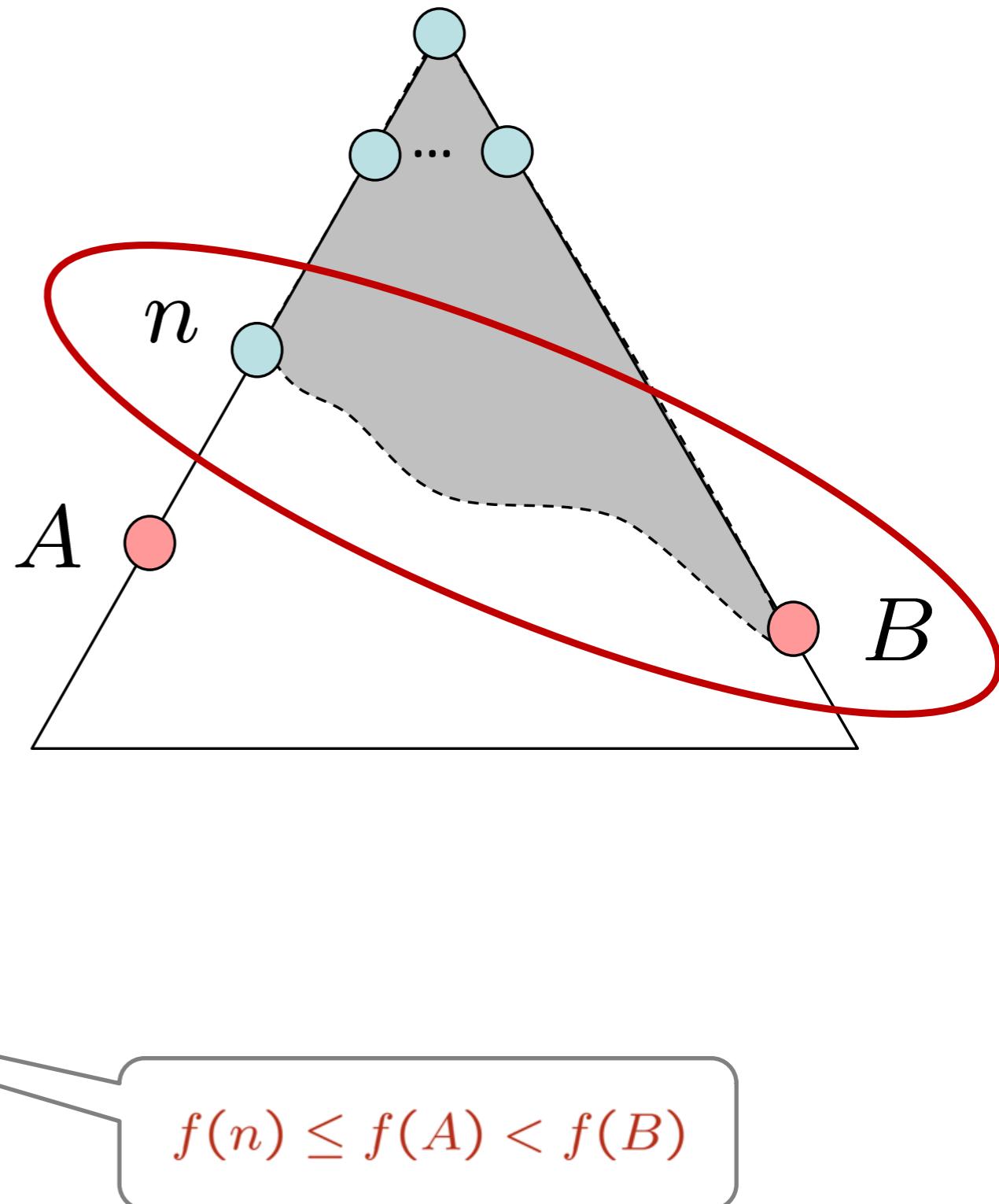
$$f(A) < f(B)$$

B is suboptimal

$h = 0$ at a goal

A* 树搜索的最优性

- 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
 - 首先证明: n 会先于 B 出队列
 - 1. $f(n)$ 小于或等于 $f(A)$
 - 2. $f(A)$ 小于 $f(B)$
 - 3. n先于B出队列

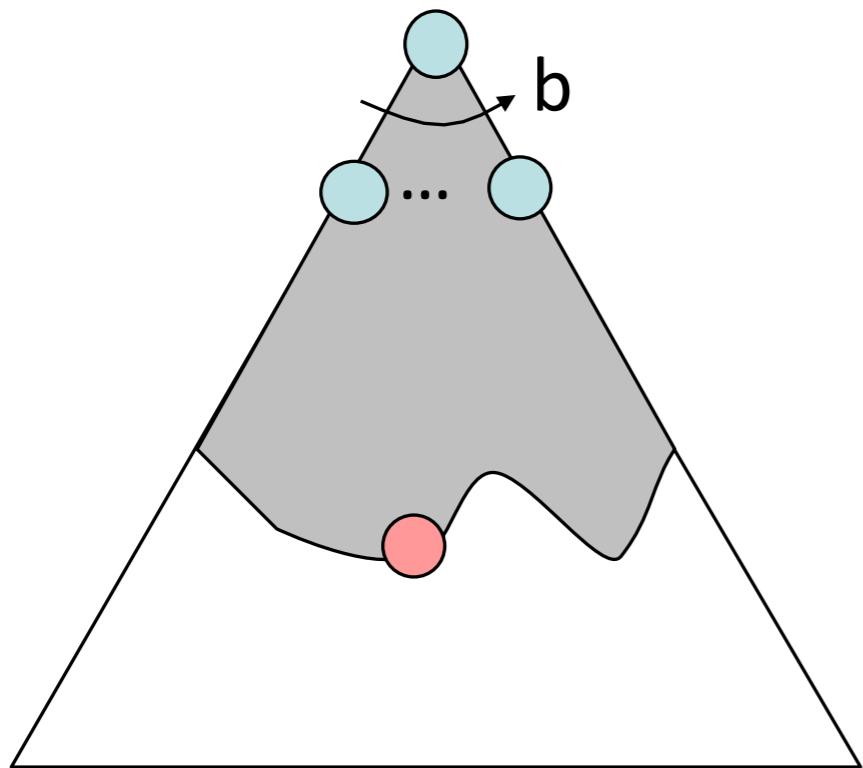


A* 树搜索的最优性

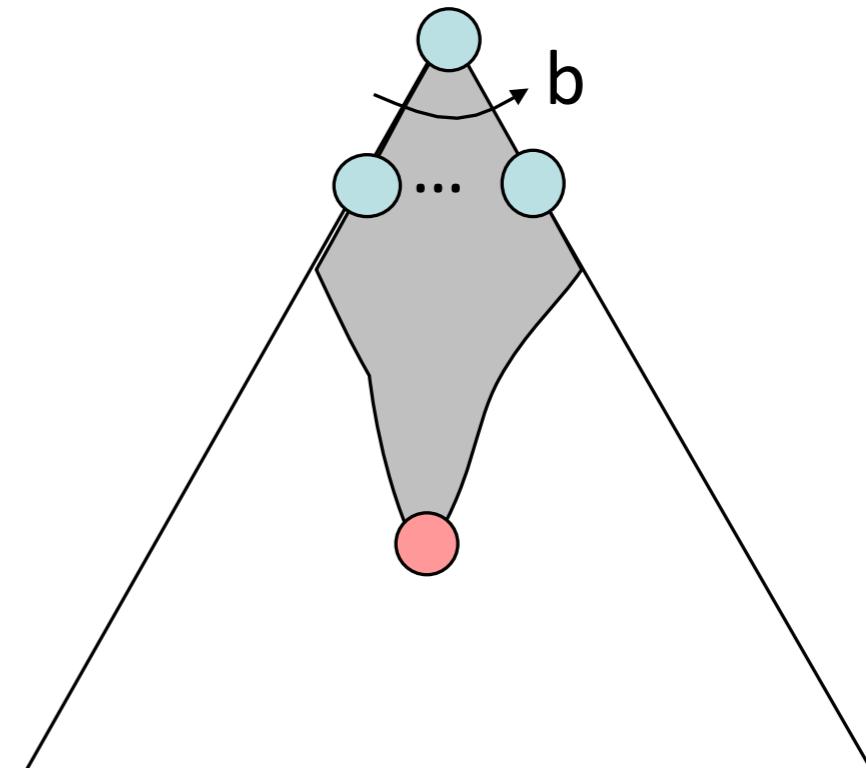
- 证明：
 - 假设B在搜索前沿里
 - A的某个祖先节点 n 也在搜索前沿里
 - 已证明：n 会先于 B 出队列
 - 所有 A的祖先节点都会在B之前出队列
 - A 先于 B 出队列
 - A* 树搜索是最优的

A*搜索的属性

成本统一搜索

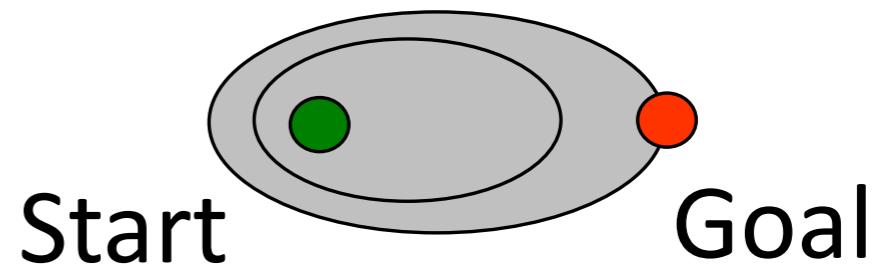
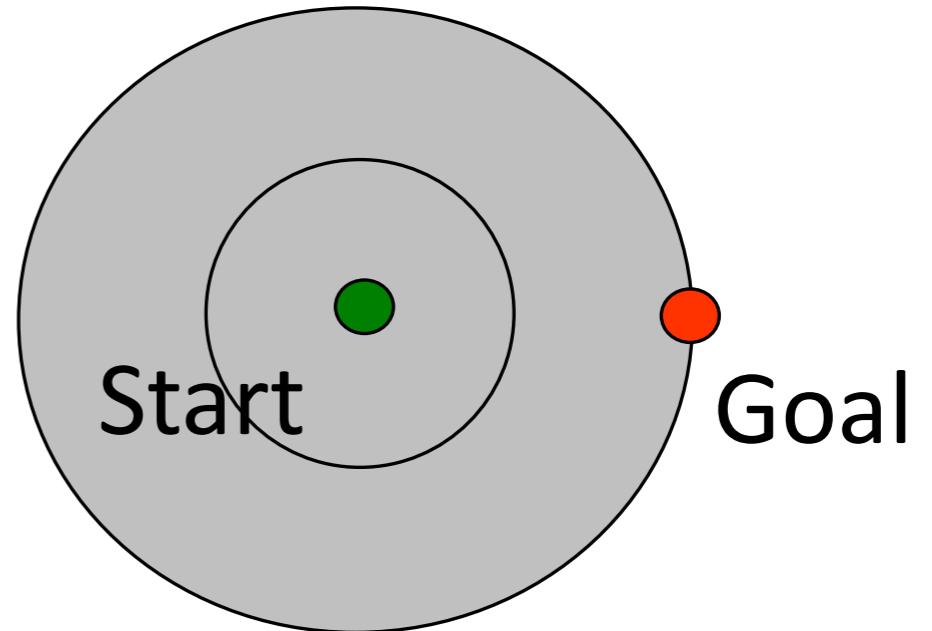


A*搜索

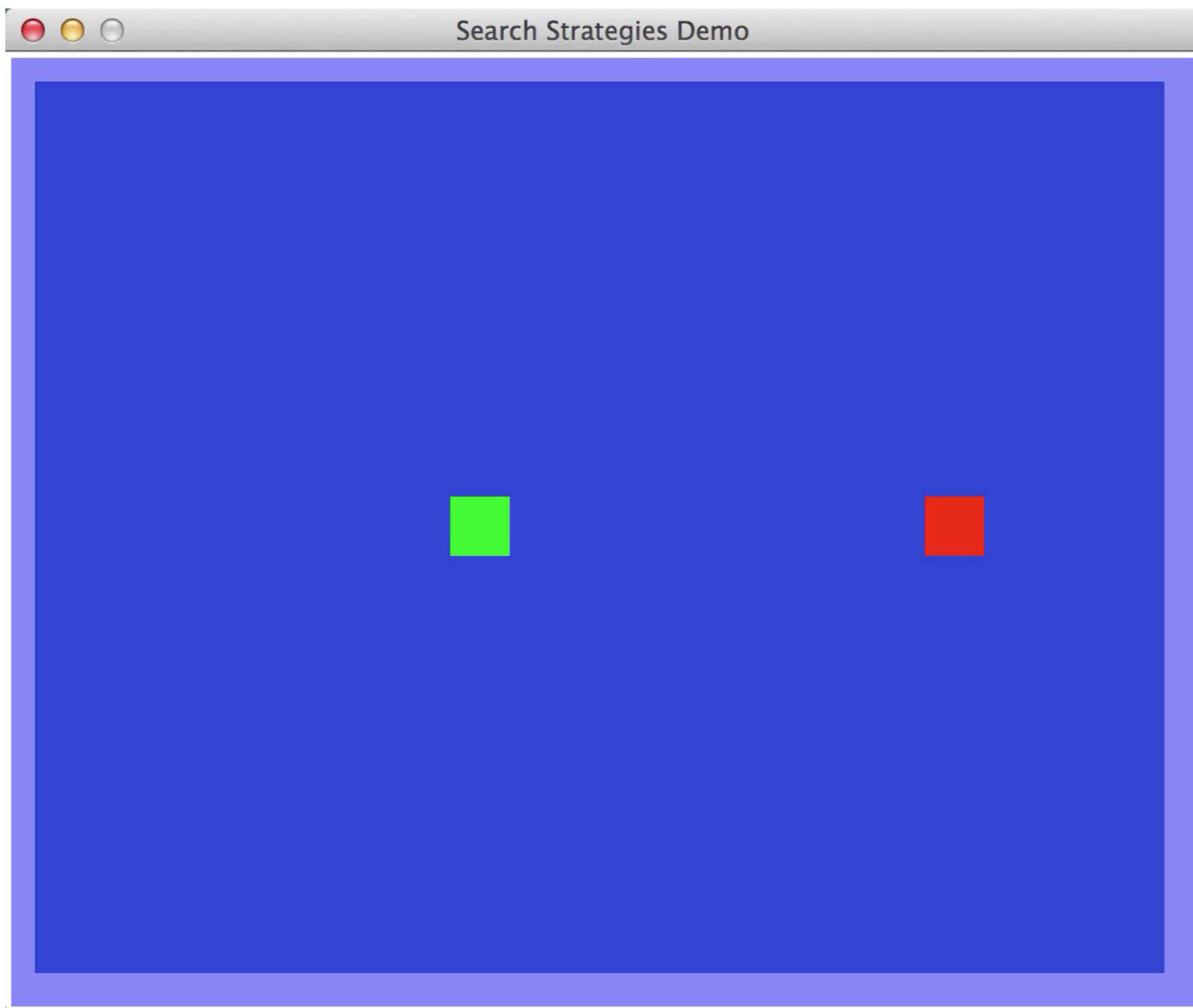


成本统一搜索 vs A* 搜索轮廓

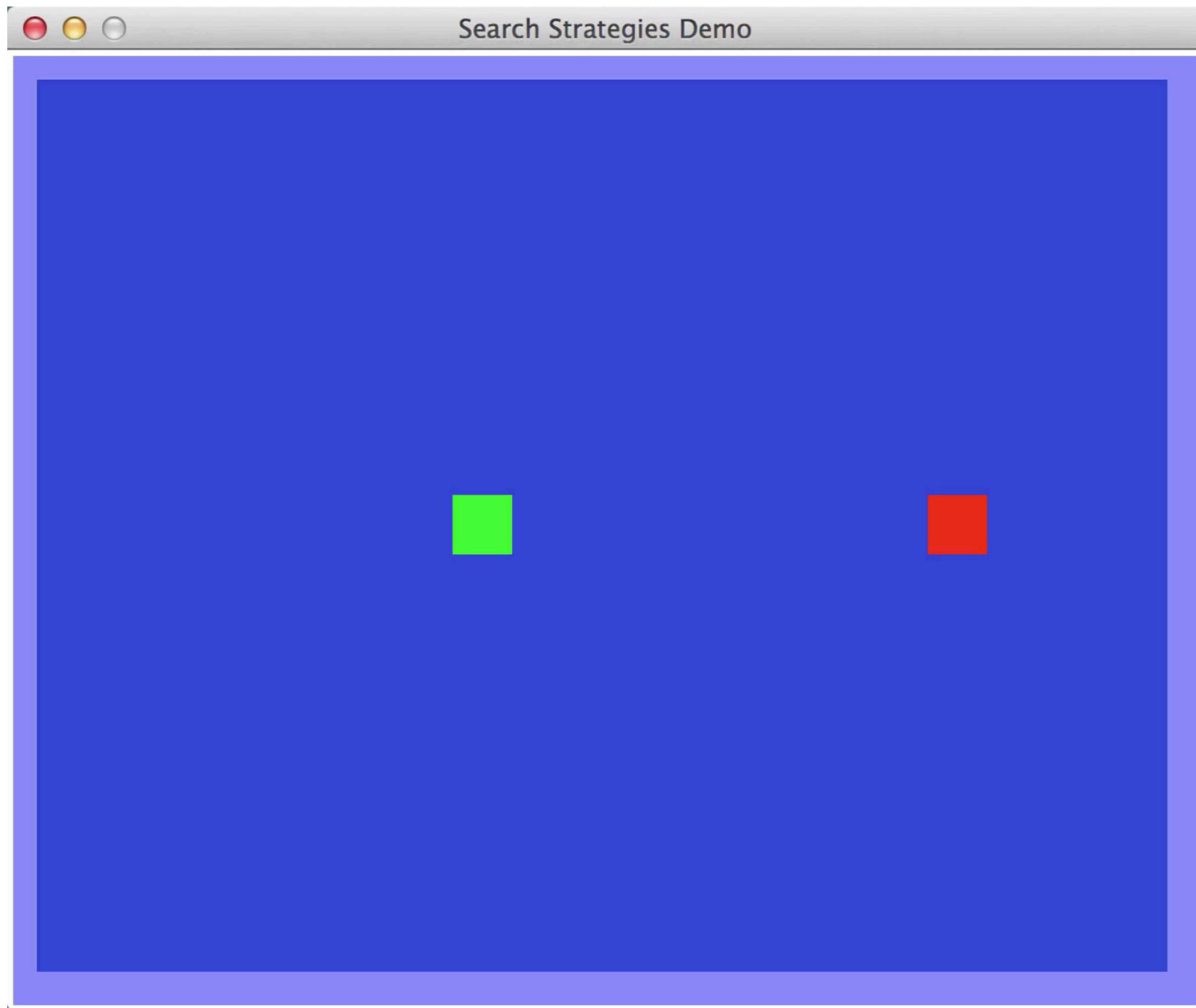
- 基于成本的统一搜索在各个方向上均匀探索
- A* 在朝向目标的方向上进行探索，同时保证解的最优化



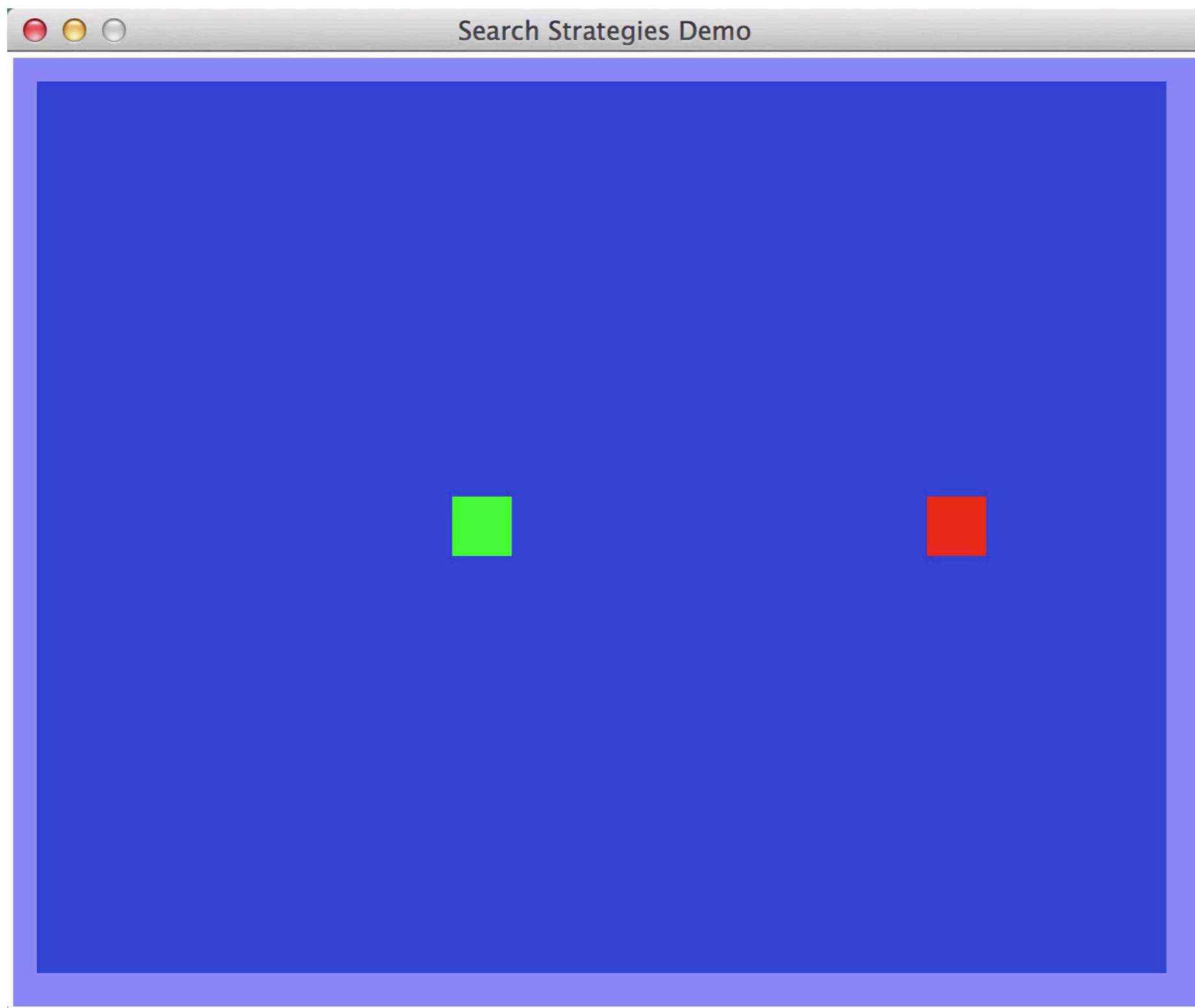
搜索轮廓(空迷宫)-基于成本的统一搜索



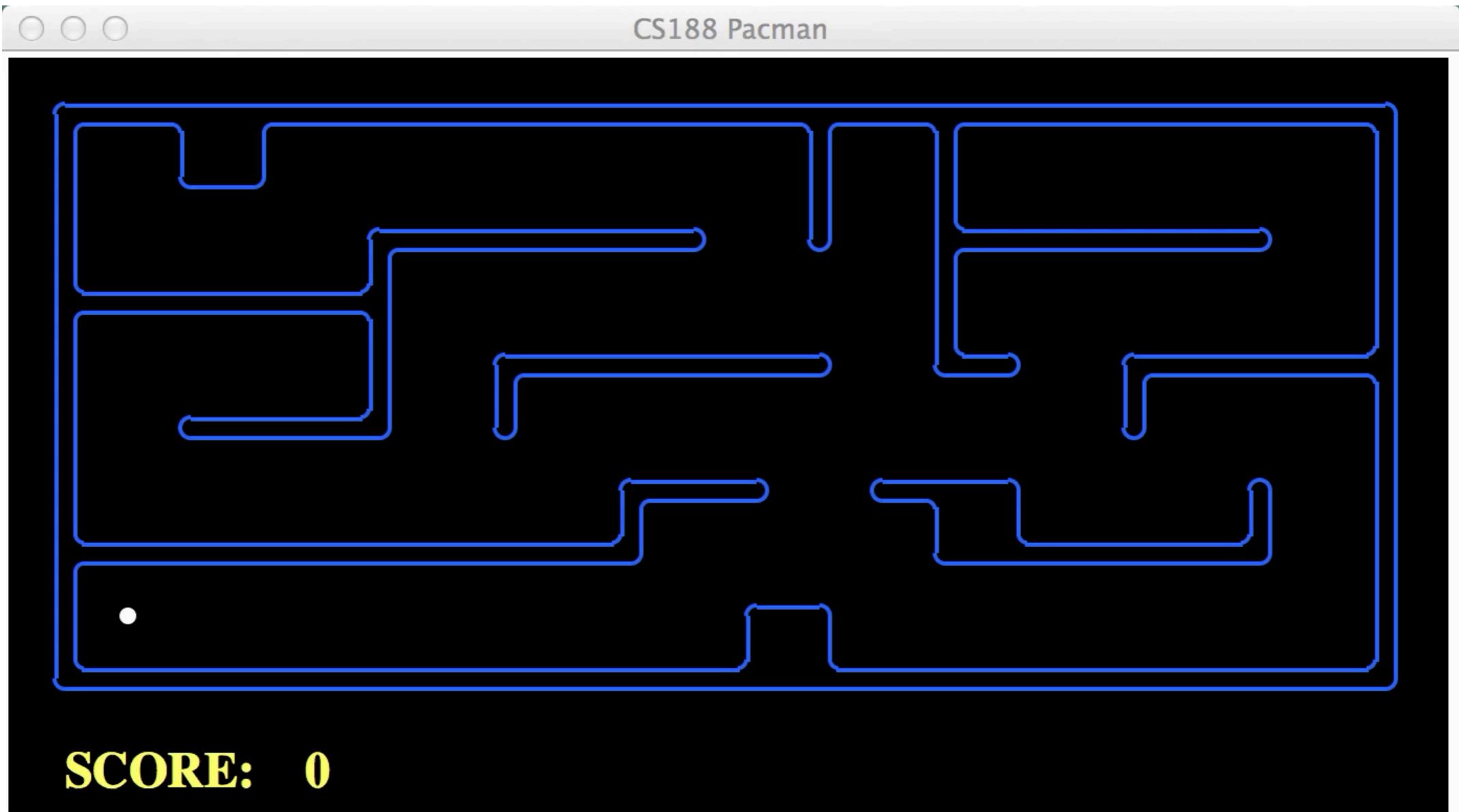
搜索轮廓(空迷宫)-贪婪搜索



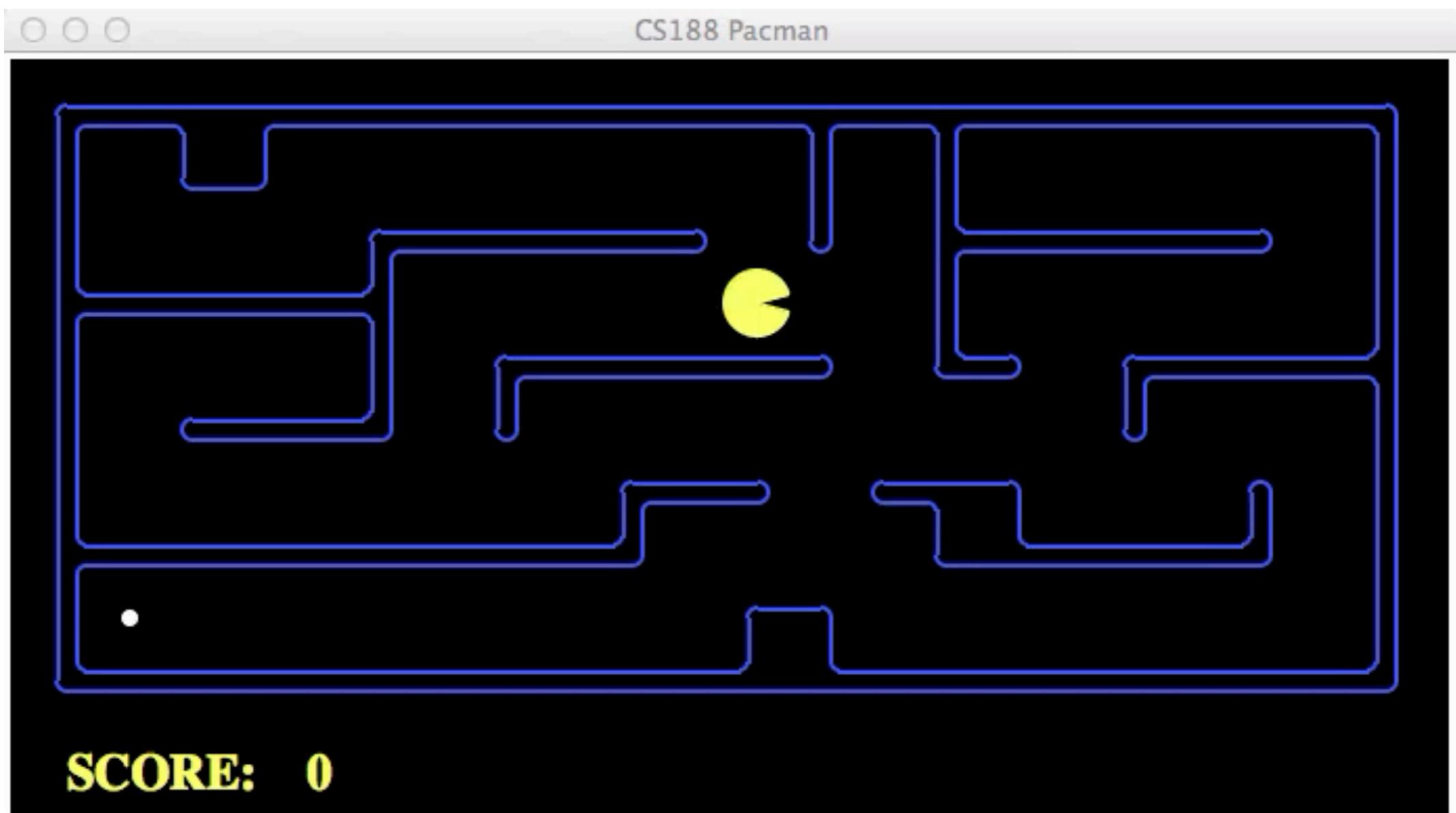
搜索轮廓(空迷宫)-A*



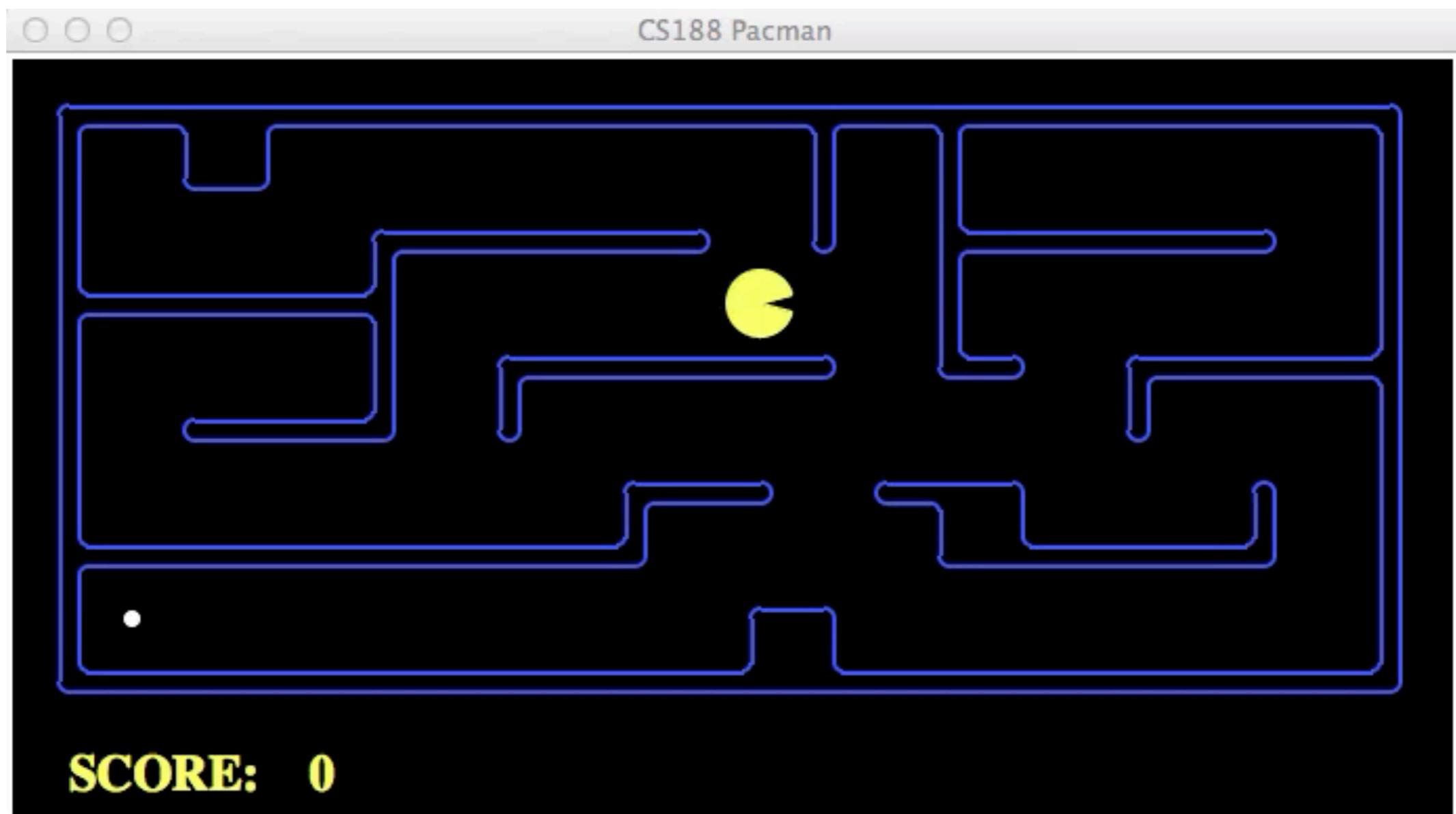
搜索轮廓(Pacman迷宫)-UCS



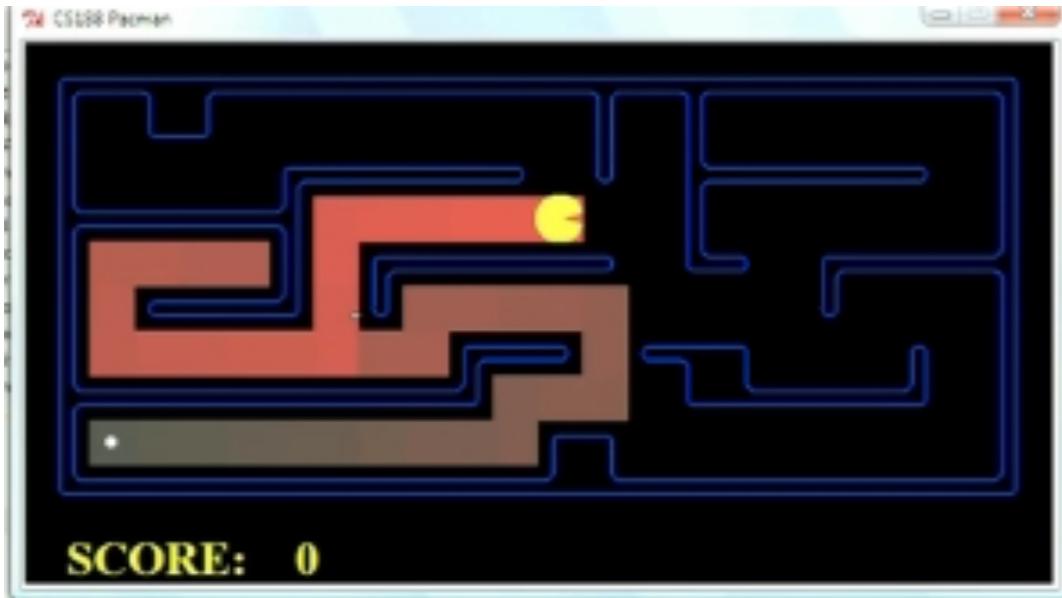
搜索轮廓(Pacman迷宫)–Greedy



搜索轮廓(Pacman迷宫)-A*



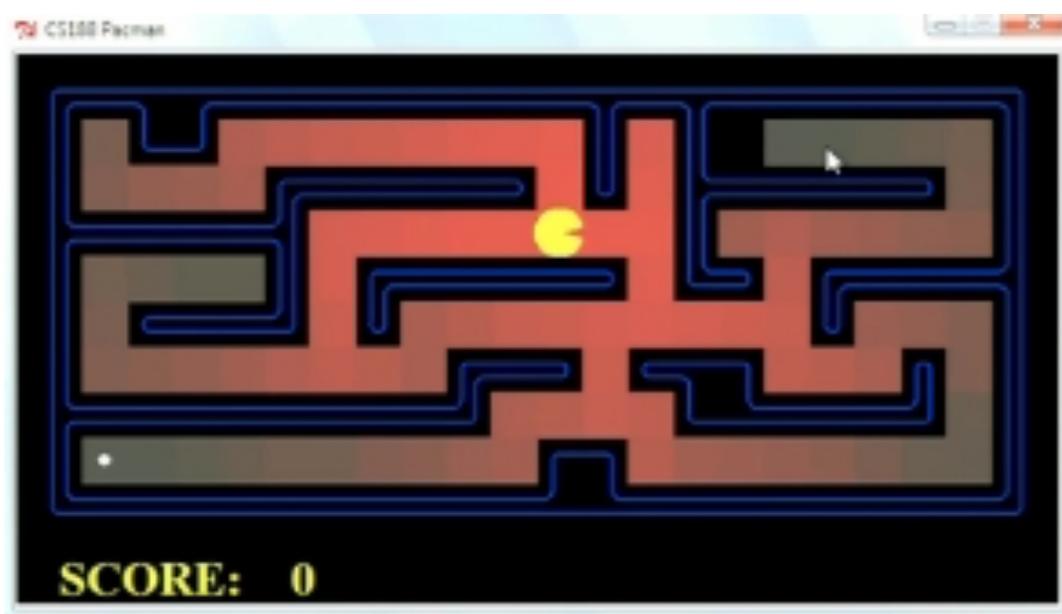
对比



Greedy



A*



Uniform Cost

A* 应用

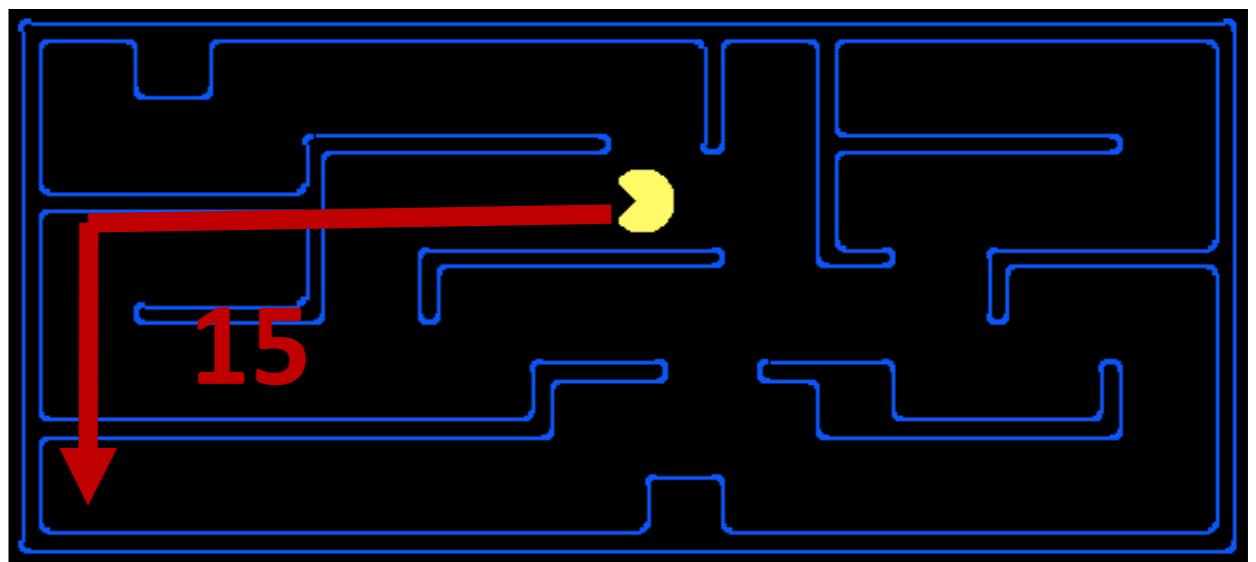
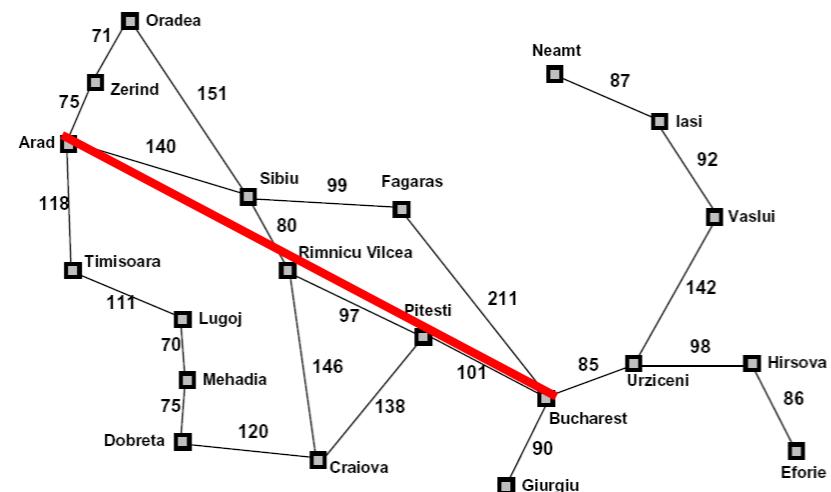
- 视频游戏
- 路径搜索问题
- 资源规划问题
- 机器人移动规划
- 语言分析
- 机器翻译
- 语音识别
- ...



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。

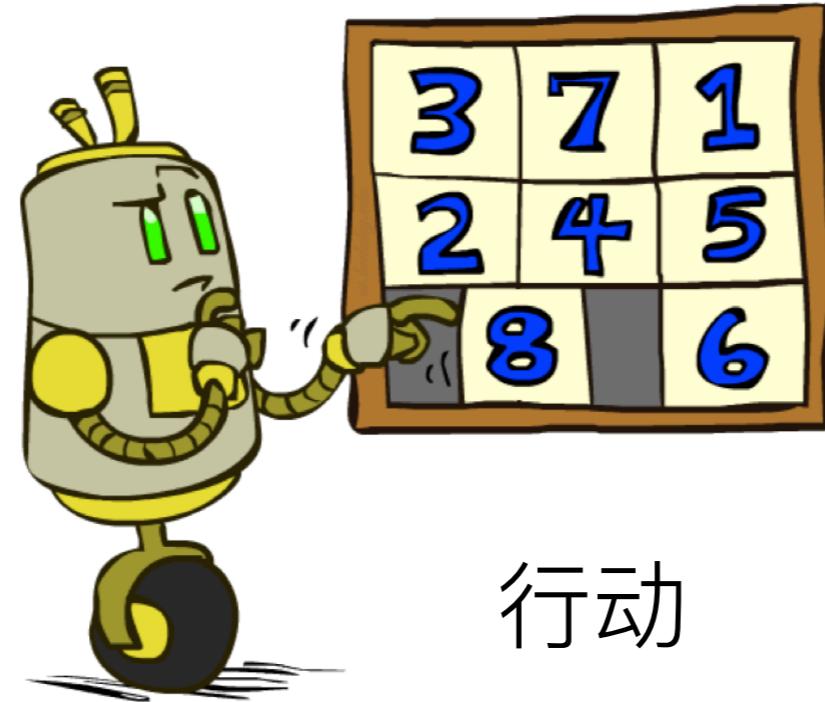
366



举例：8 数字谜题

7	2	4
5		6
8	3	1

开始状态



行动

	1	2
3	4	5
6	7	8

目标状态

状态：总共有多少种状态？

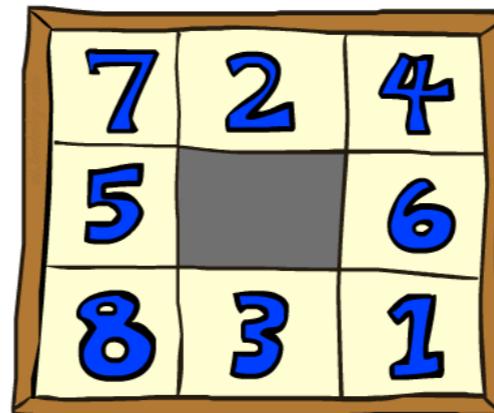
行动：开始状态有多少种可选行动？

行动成本？

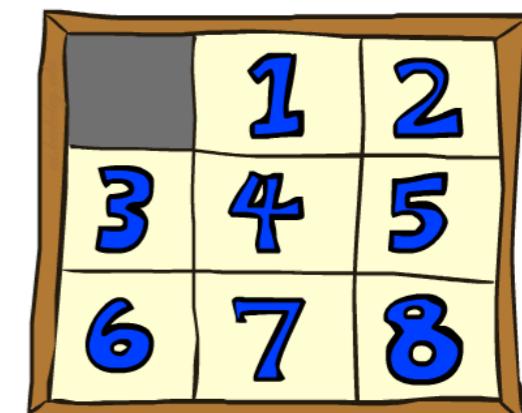
8 数字谜题

启发式: 错位方块的数量
为什么是可接受性的?

$$h(\text{开始}) = 8$$

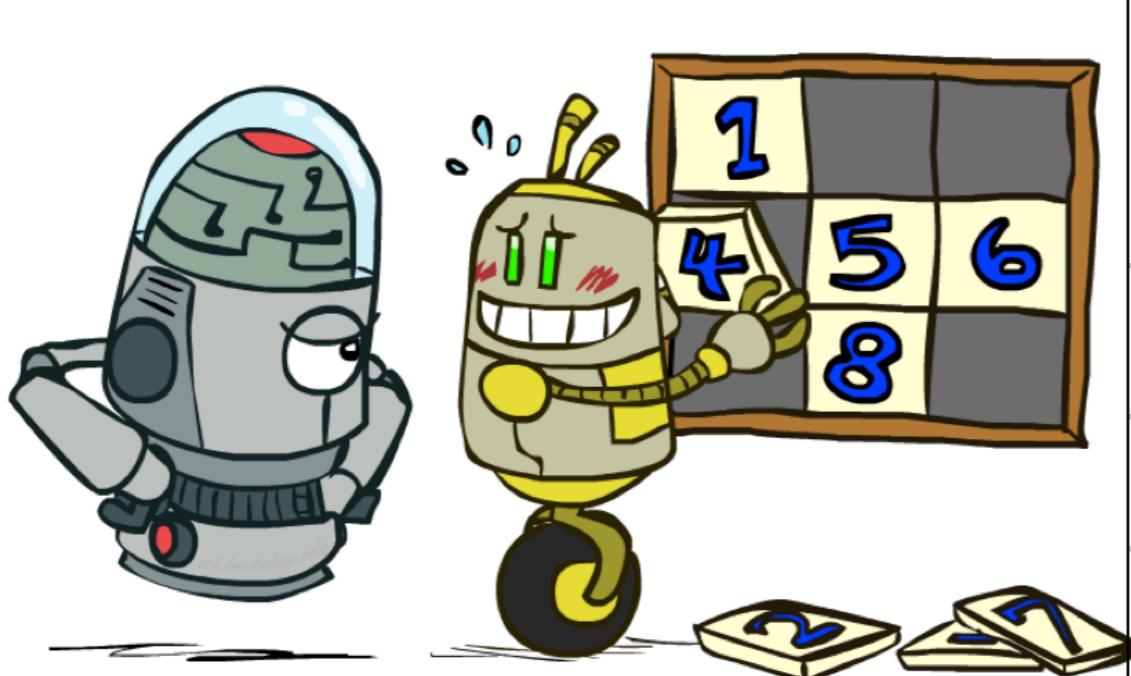


Start State



Goal State

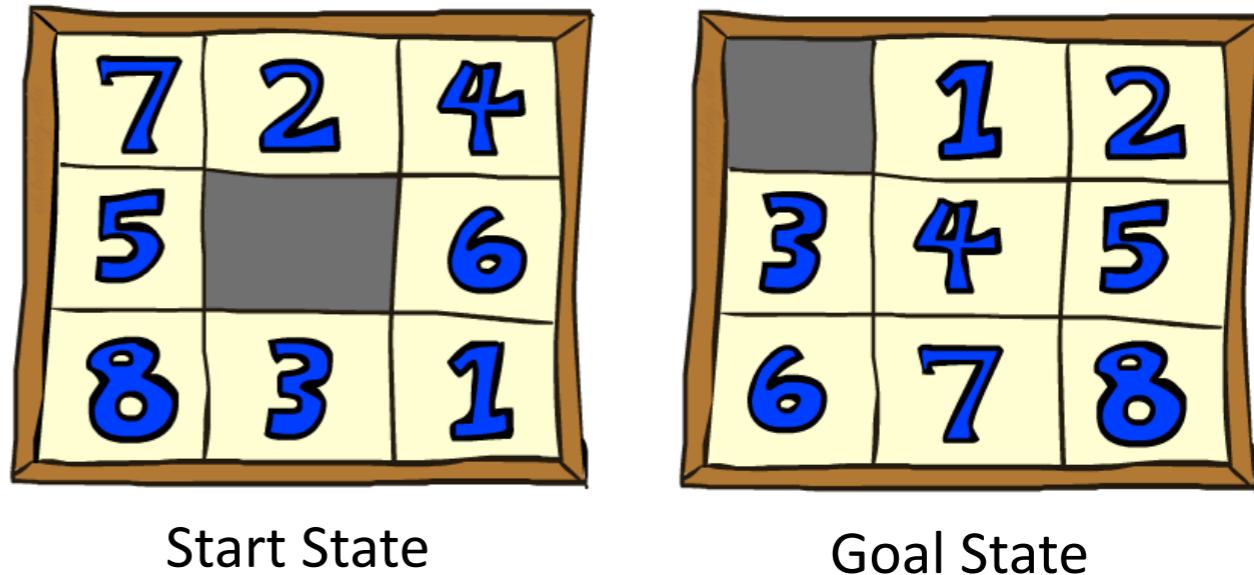
松弛问题的启发信息(假设可直接跳到位)



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

8 数字谜题

- 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
- 即曼哈顿距离是否是可接受的？
- $H(\text{开始}) = 3+1+2+\dots=18$



	平均节点扩展数，当最优解的深度是：		
	...4步	...8步	...12步
A* TILES	13	39	227
A* MANHATTAN	12	25	73

选择启发式函数

- 可否可以使用实际成本作为启发式函数?
 - 这是可以接受的吗?
 - 会节省扩展节点吗?
 - 有什么问题吗?
- A*:评估质量和计算量之间的权衡
 - 随着启发式越来越接近真实的成本，将扩展更少的节点，但通常会在每个节点上做更多的工作来计算启发式本身

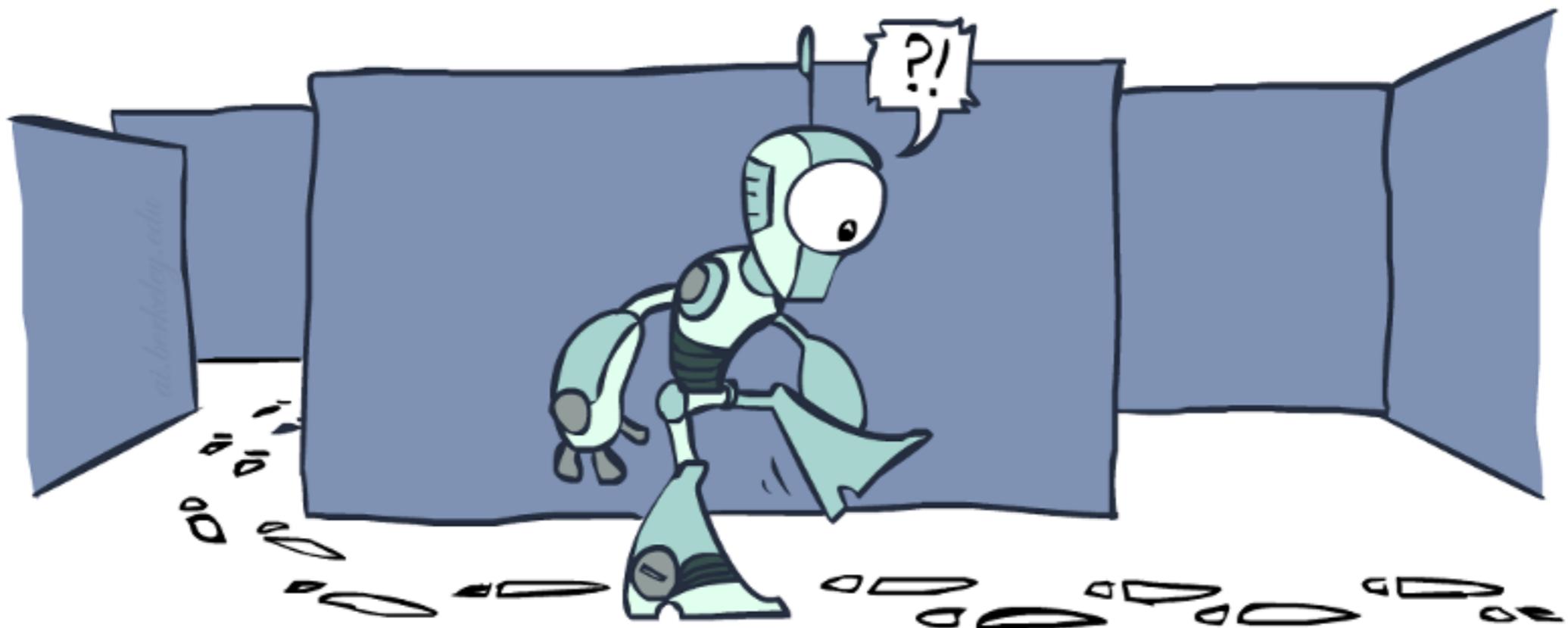
组合启发式(函数)信息

- 支配优势: $h_a \geq h_c$ 如果满足:

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

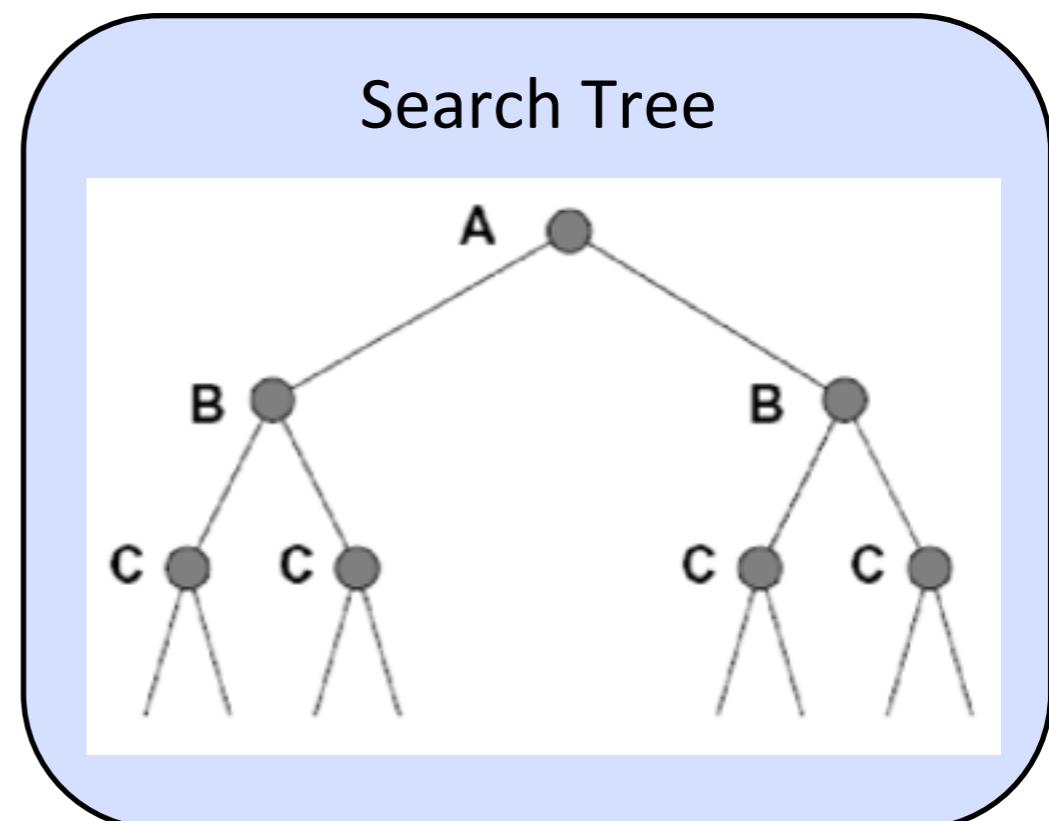
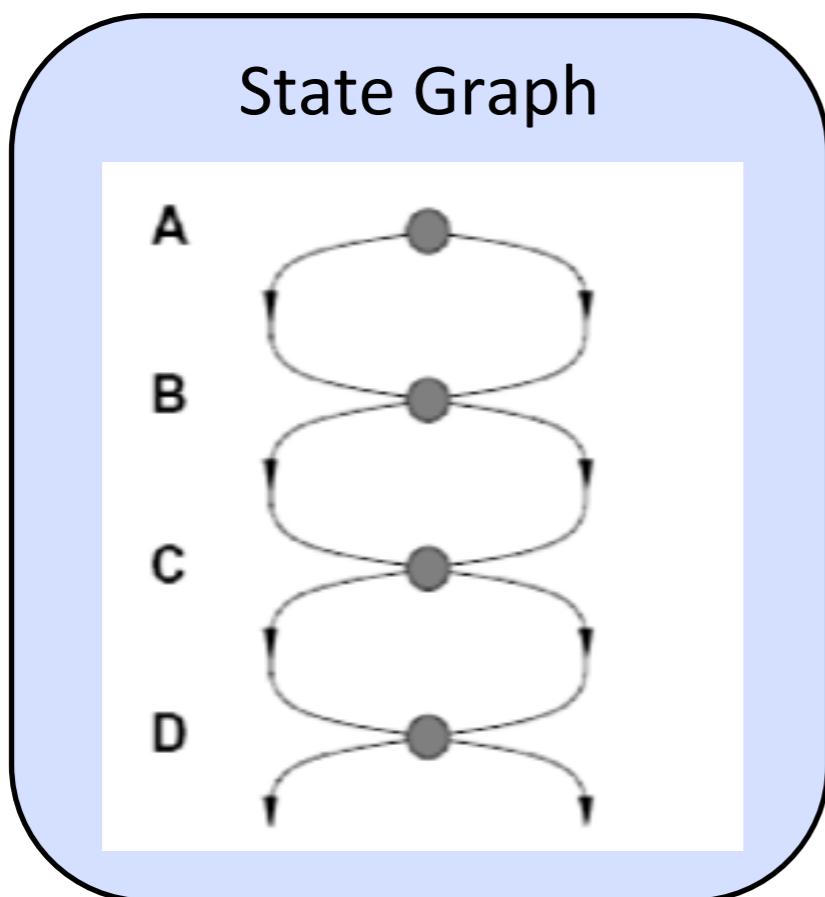
- 一般讲, 越大越好, 只要保持可接纳性。
 - H 是0的话, 比较糟, (A^* 变成什么, 如果 $h=0$?).
 - 和真实目标成本相同最好, 但很难达到!
-
- 如果两个启发式函数, 都不支配对方怎么办?
 - 形成一个新的, 通过最大式组合: $h(n) = \max(h_a(n), h_b(n))$
 - 这个既是可接纳的, 也是对之前任一个都有支配优势的启发式函数。

图搜索



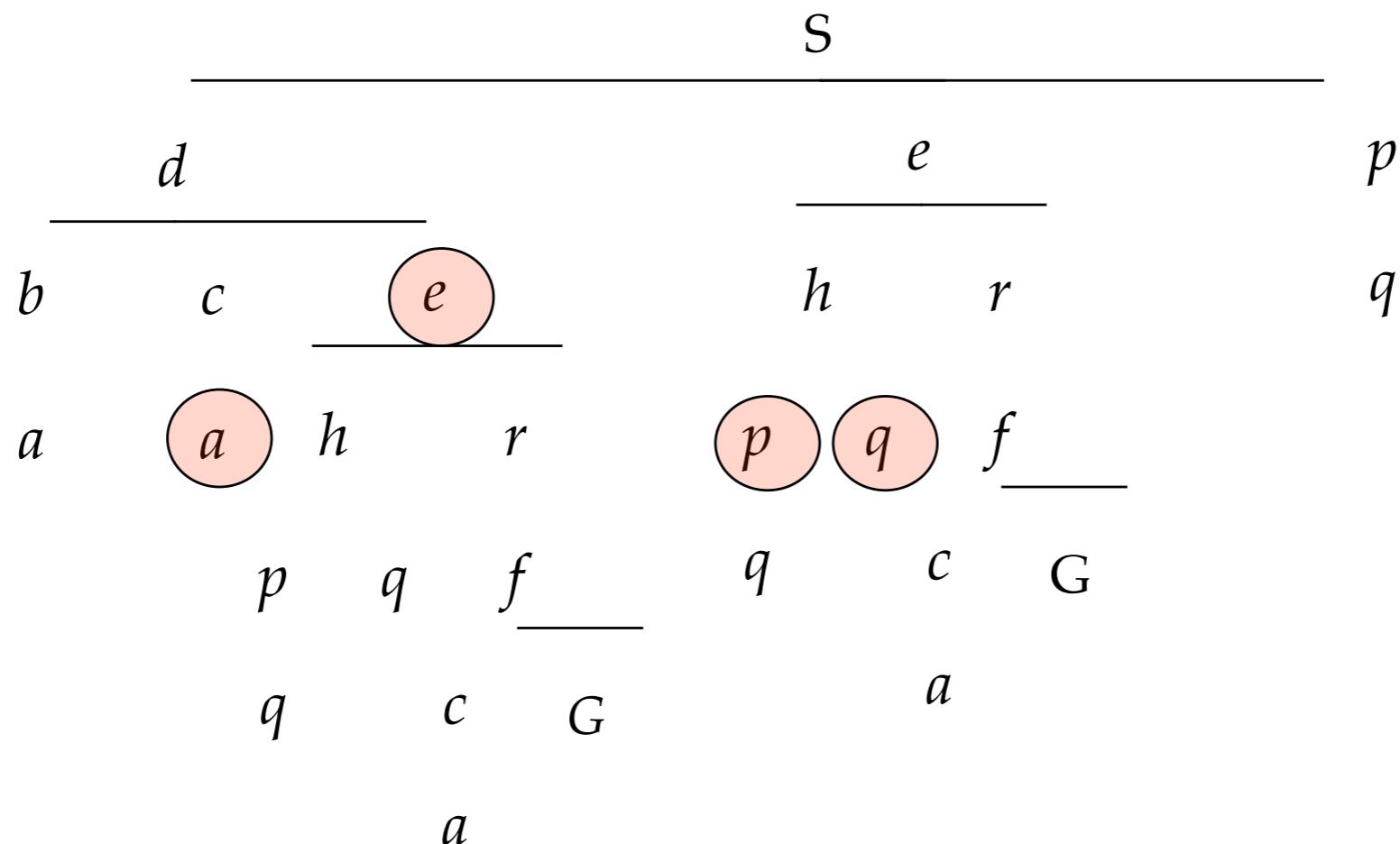
树搜索:额外的工作!

- 不能检测到重复的状态会导致成倍增加的工作量。



图搜索

- 例如，在BFS中，我们不应该费心扩展圈起来的节点(为什么?)

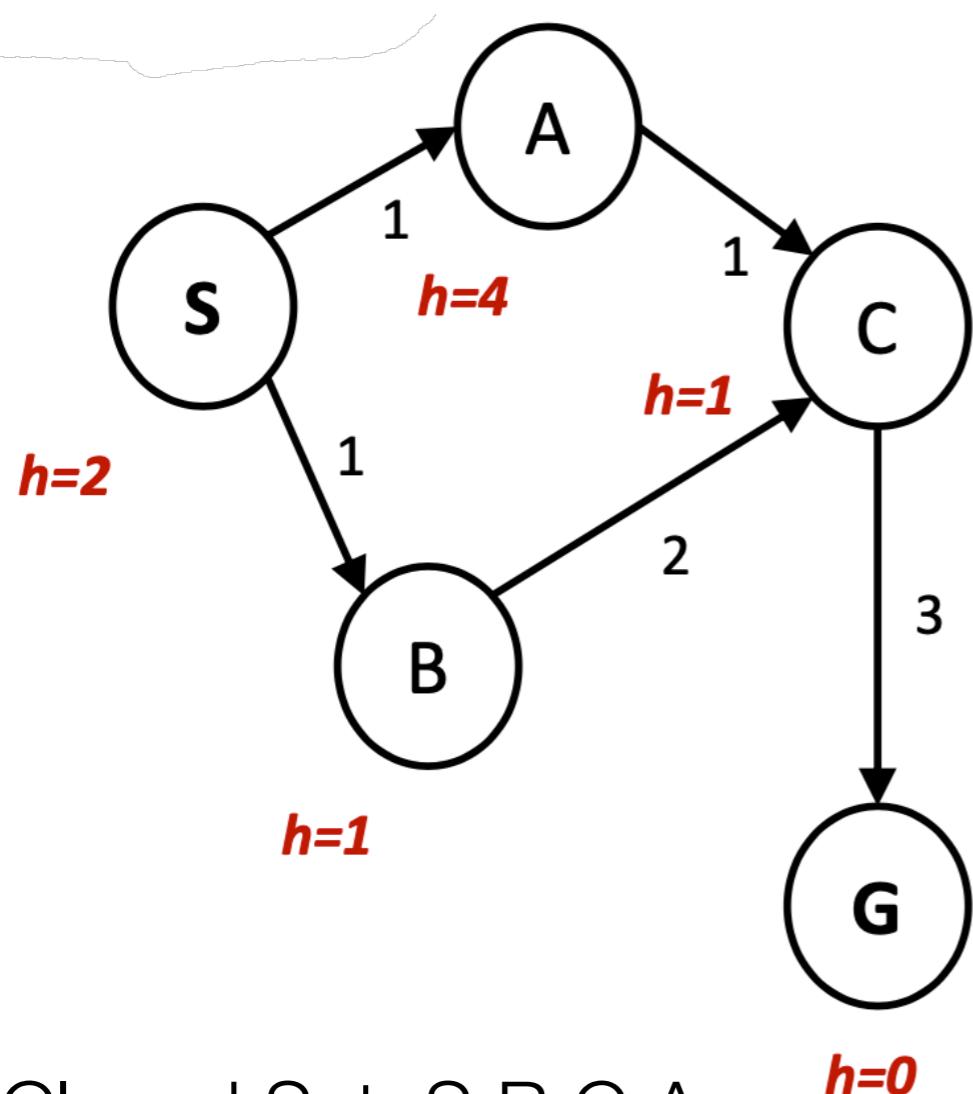


图搜索

- 想法:不要两次扩展(expand)同一个状态
- 如何实现:
 - 树搜索+已被扩展过的状态集合(“闭集 closed set”)
 - 扩展搜索树节点之前, 检查以确保其状态从未被扩展过
 - 如果不是新的状态, 则跳过它; 如果是, 则加入到闭集中
- 重要: 存储闭集作为一个集合(set), 而不是一个列表(list)
- 图搜索是否会破坏完整性?为什么/为什么不?
- 最优性如何?

A*图搜索走错了吗？

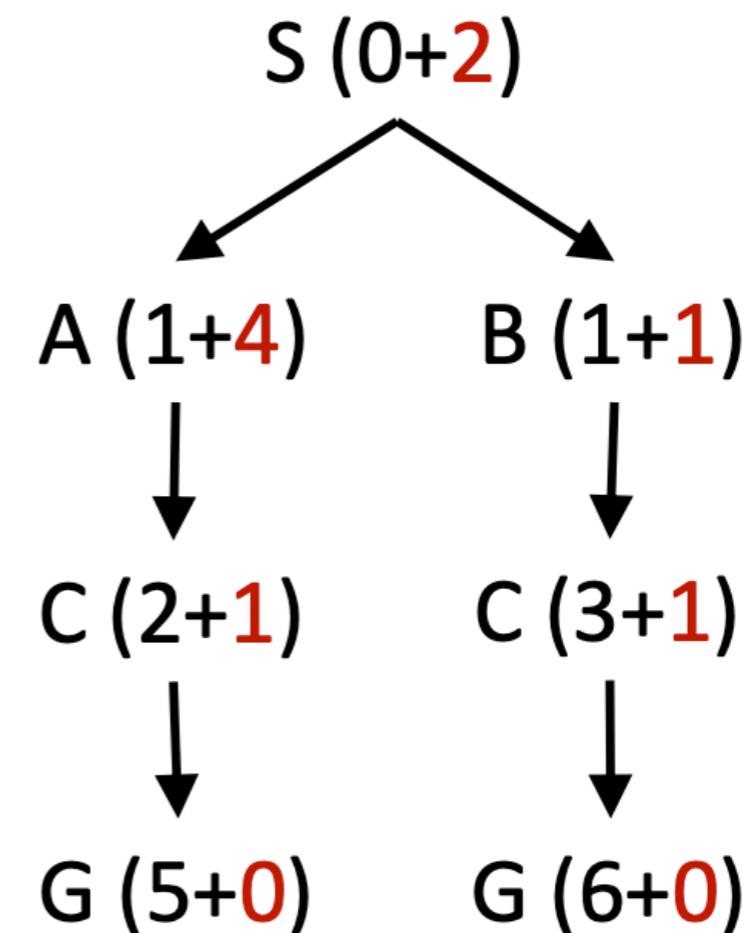
状态空间图



Closed Set: S B C A

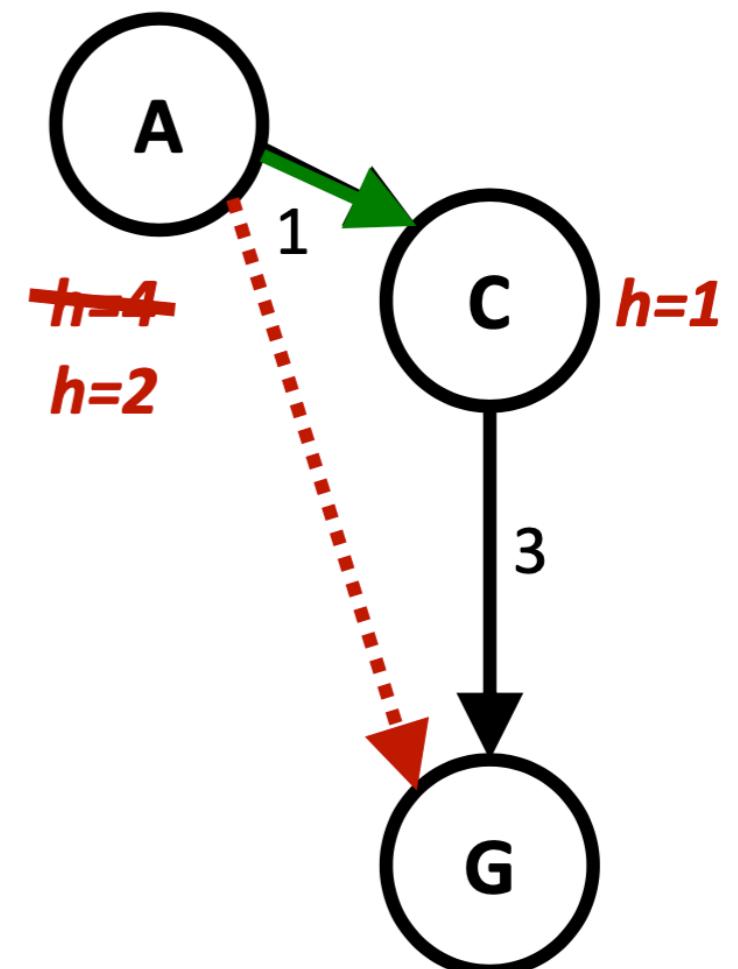
A-C不会被扩展，因为C已被访问过，因此错过了最优路径。

搜索树



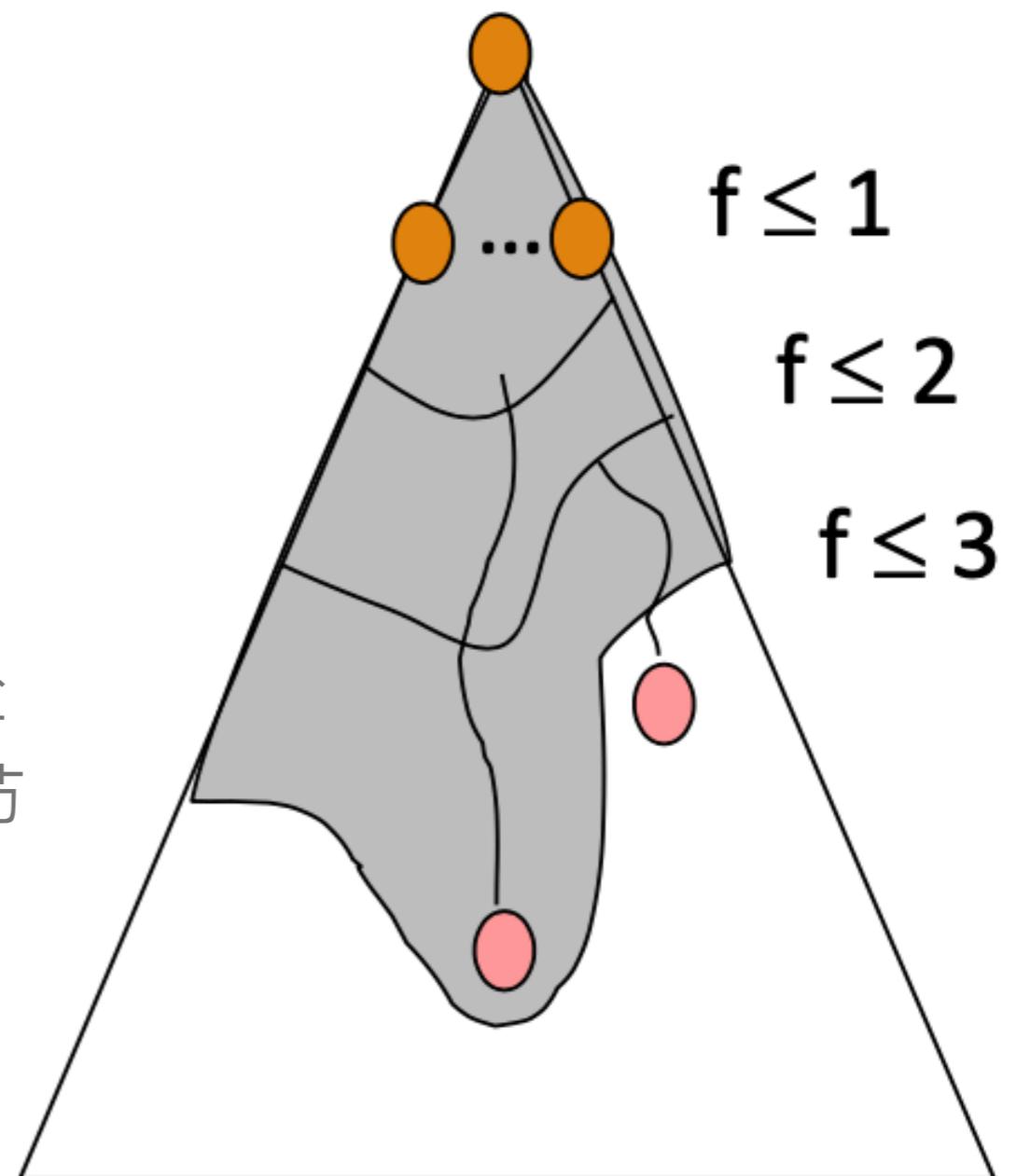
启发性函数的一致性

- 主体思想: 估计成本 < 实际成本
 - 可接受性: 启发函数估计成本 ≤ 实际路径成本
$$h(A) \leq \text{从 } A \text{ 到 } G \text{ 的实际路径成本}$$
 - 一致性: 启发函数估计的步骤成本(弧成本) ≤ 实际步骤成本
$$h(A) - h(C) \leq \text{cost}(A \text{ 到 } C)$$
- 一致性的结果:
 - f 值在同一路径搜索中不会减少
$$h(A) \leq \text{cost}(A \text{ 到 } C) + h(C)$$
 - A*图搜索是最优的(找到的解是最优解)



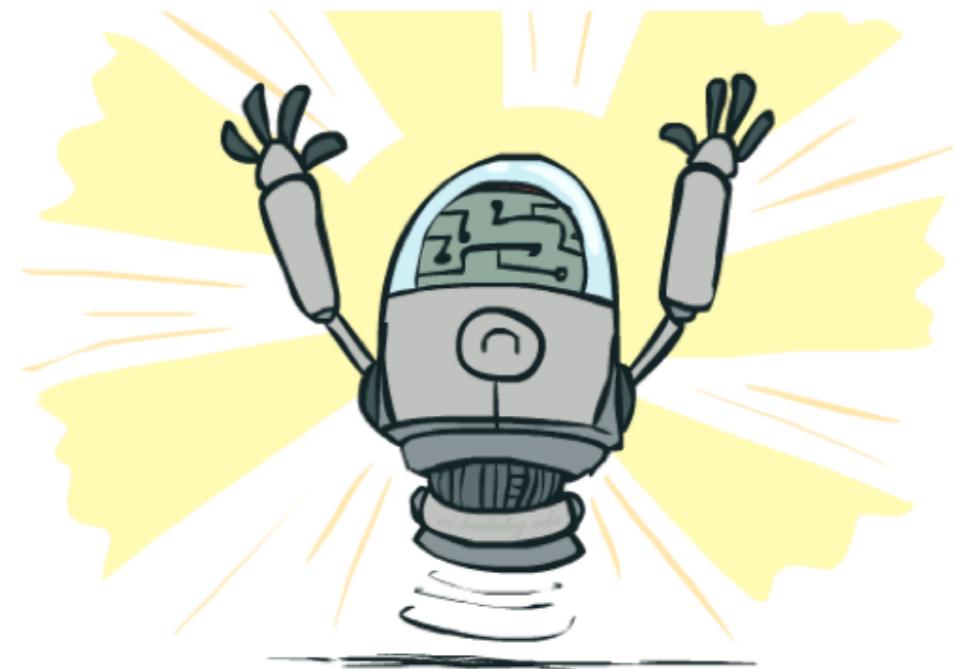
A* 图搜索的最优性(Optimality)

- 如果A*的 h 是一个一致性的启发式函数，那么：
 - A*扩展节点过程中， f 值递增(f -轮廓)
 - 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



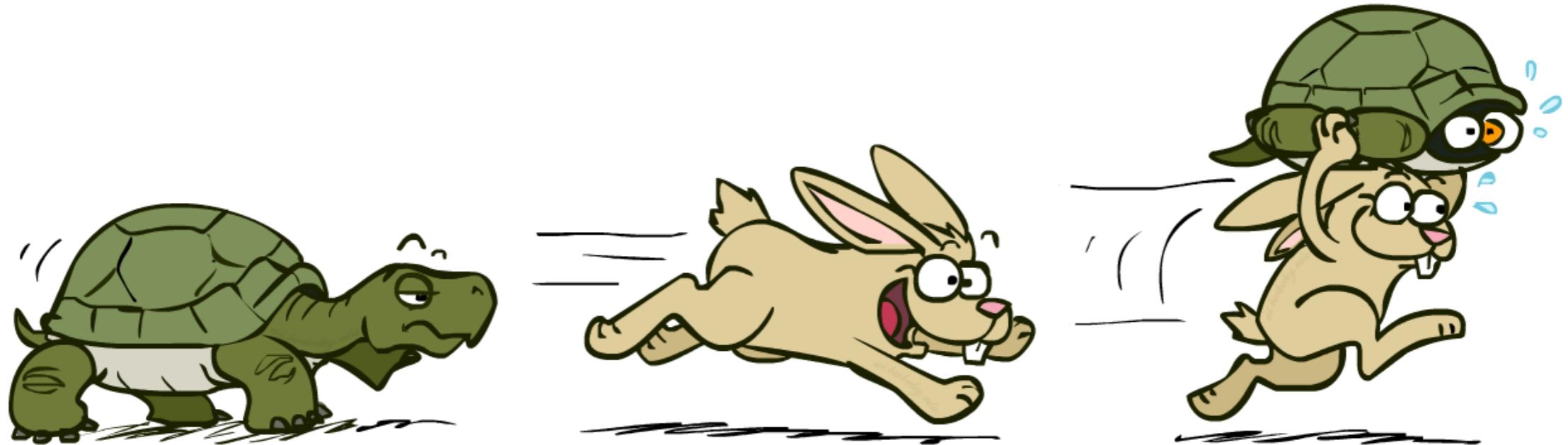
最优化

- 树搜索:
 - A*最优, 如果启发式函数是满足可接纳性的
 - 基于成本的统一搜索(UCS) 是一个特例 ($h = 0$)
- 图搜索:
 - A*最优, 如果启发式函数是满足一致性的
 - 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。



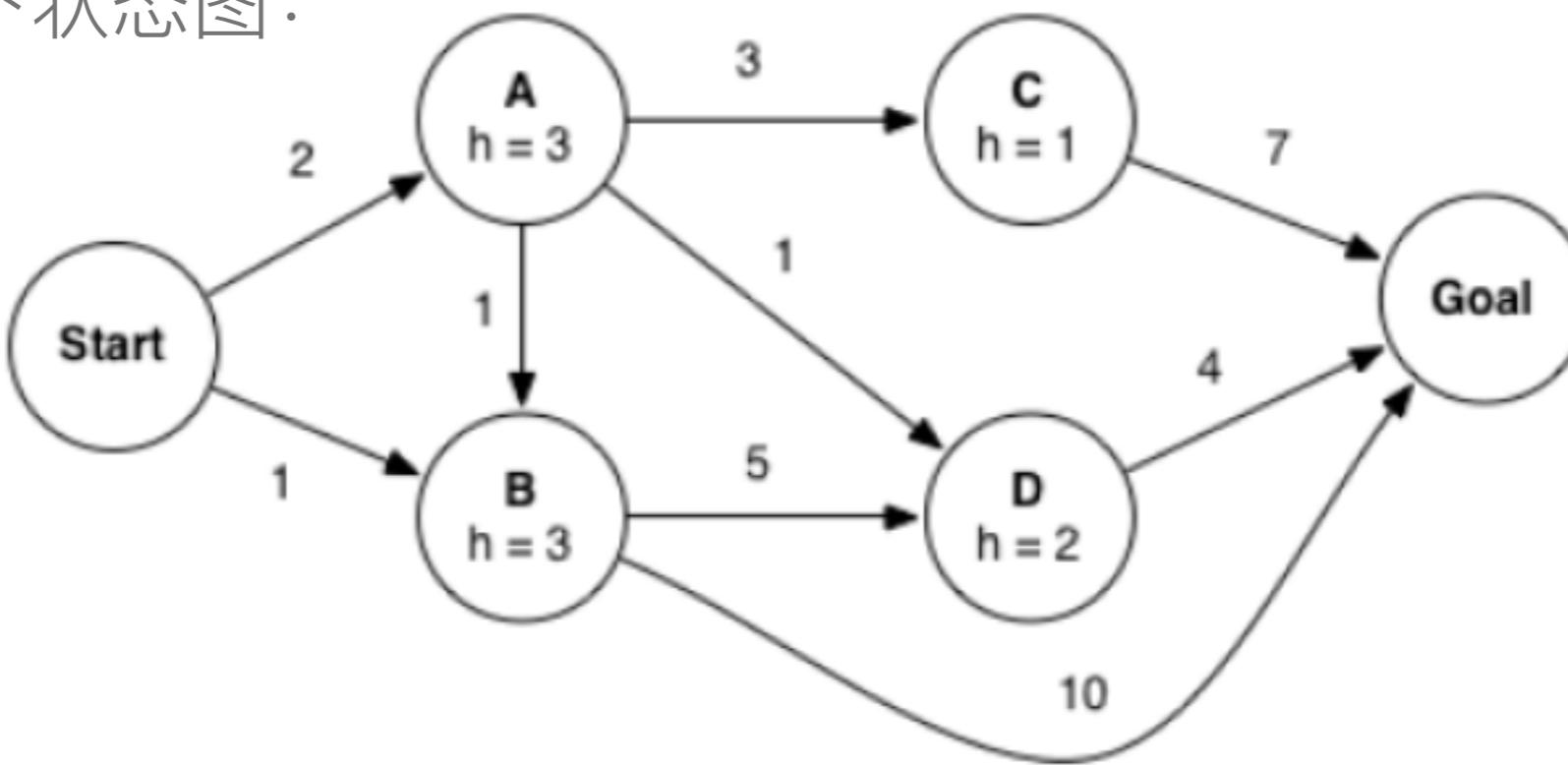
A*: 总结

- A* 既使用了来程路径成本，又利用了估计的前程路径成本
- 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



作业

- 考慮以下状态图：



1. 请编写并执行基于成本的统一搜索算法，给出状态被扩展的顺序是什么？返回的路径是什么？
2. 请编写并执行A*搜索算法，给出状态被扩展的顺序是什么？返回的路径是什么？