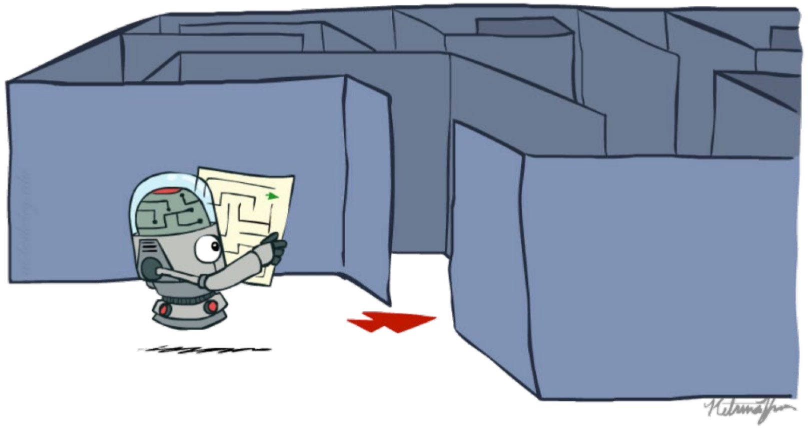
3.5 有信息（启发式）搜索策略

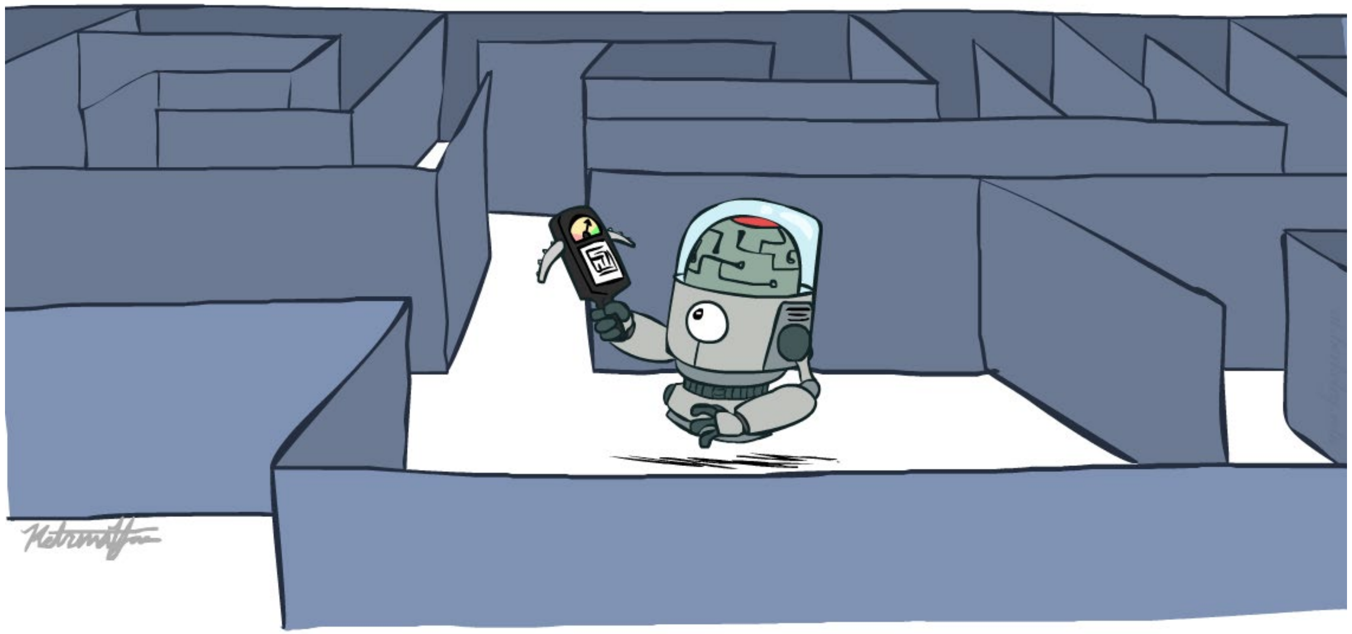


无信息搜索

(除了问题本身外，没有任何额外的信息)

有信息搜索

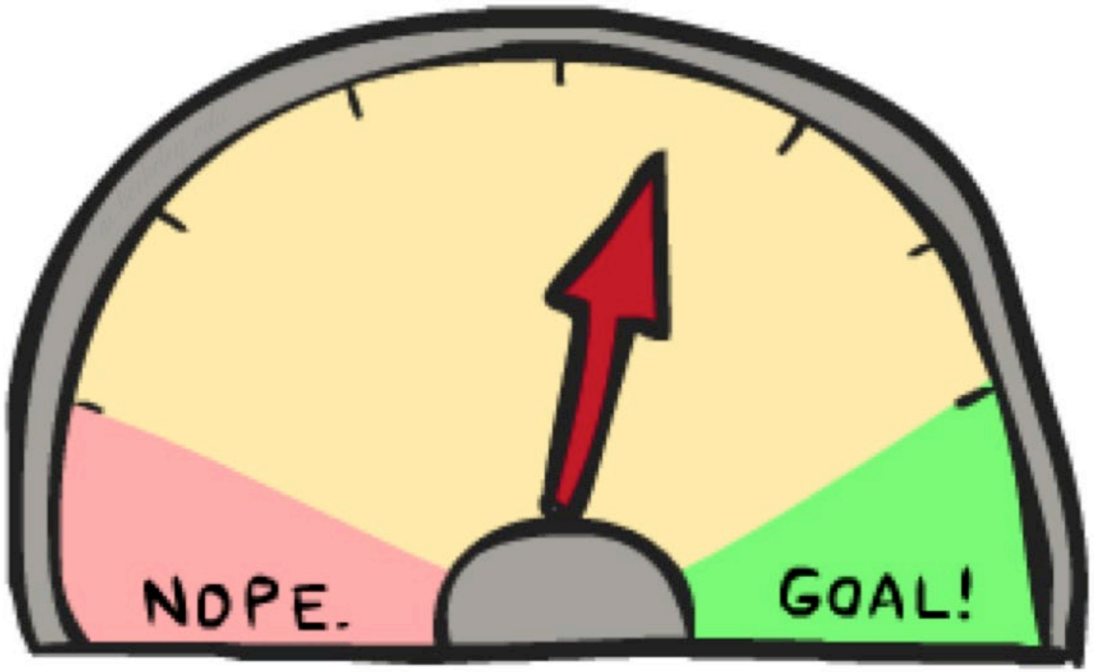
(除了问题本身外，还有启发式信息)



Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley (ai.berkeley.edu).]

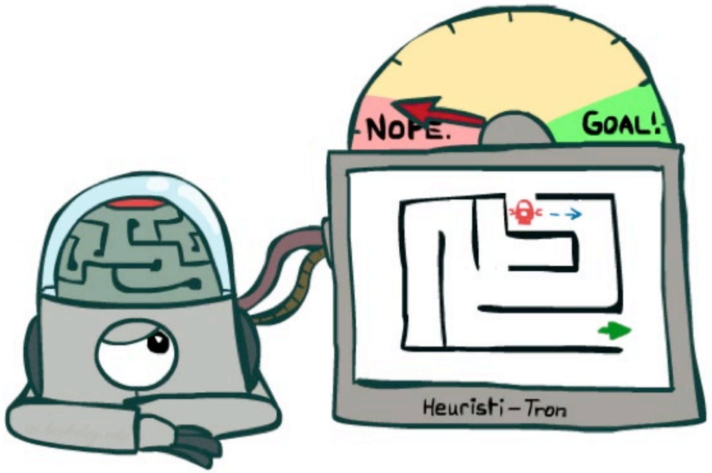
目录

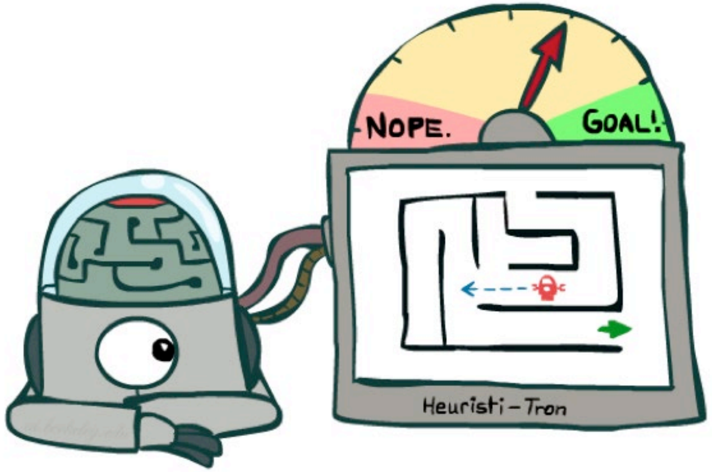
- 启发式函数
- 有信息搜索策略
  - 贪婪最佳优先搜索
  - A* 搜索



最佳优先搜索

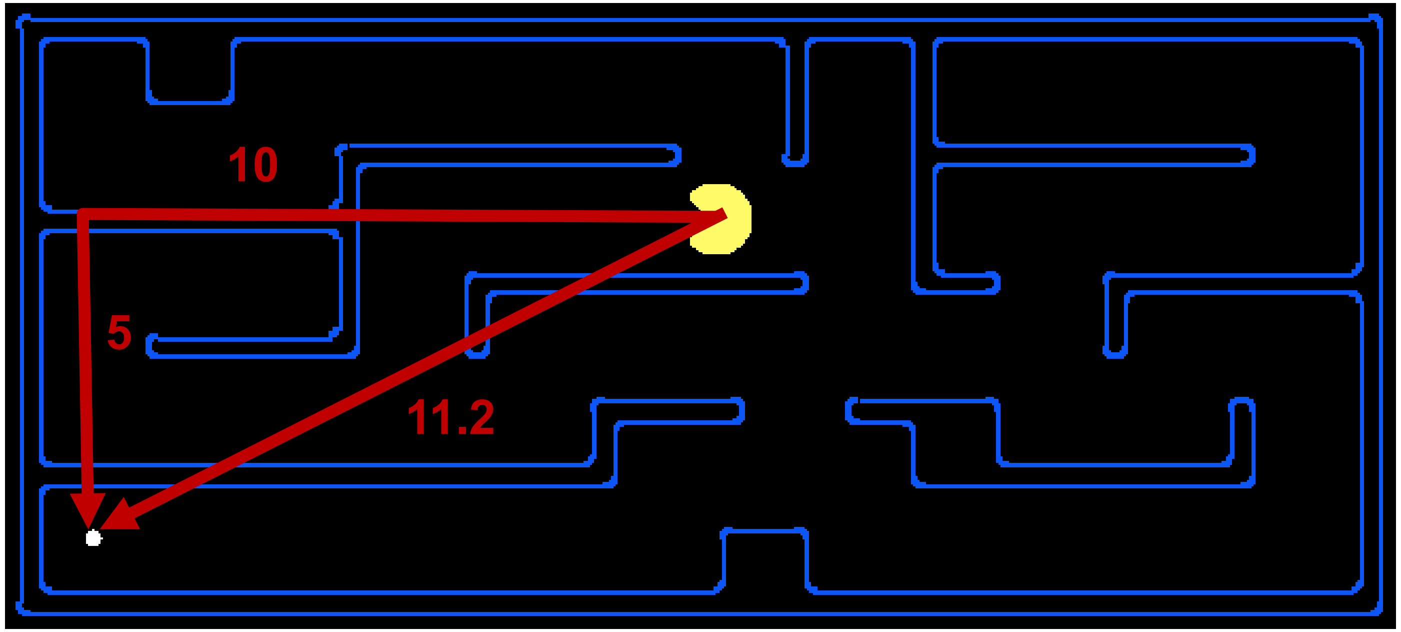
- 想法: 创建一个评估函数f(n) 用于每个结点
  - 评估 “可取性”, 确定哪个结点最有可能在通向目标的最佳路径上
  - 搜索策略： 优先级队列
  - 扩展最可取的结点， **总是选择“最有希望”的结点作为下一个被扩展的结点**

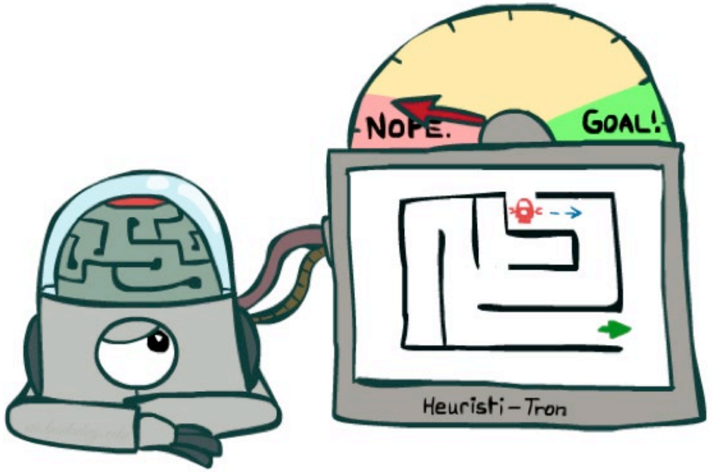


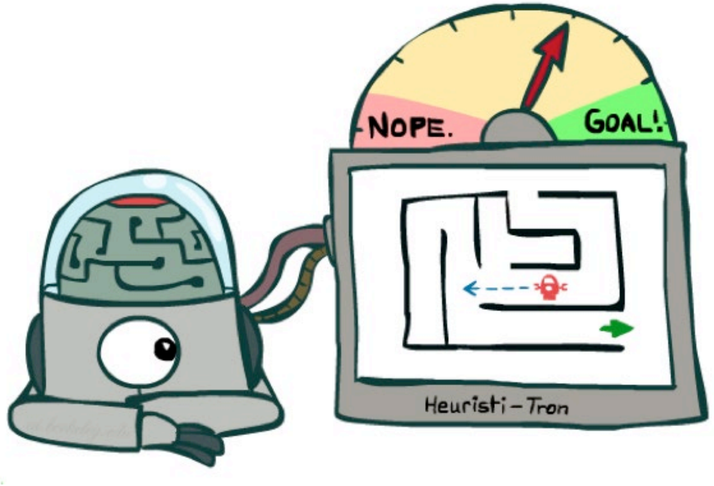


启发式函数

- 评估函数f(n): 评估结点的“可取性”
- 启发式函数h(n): 结点n到目标G的最小代价路径的代价估计值
  - 利用问题的额外信息，由问题而定的函数
  - Examples: Manhattan distance, Euclidean distance for pathing

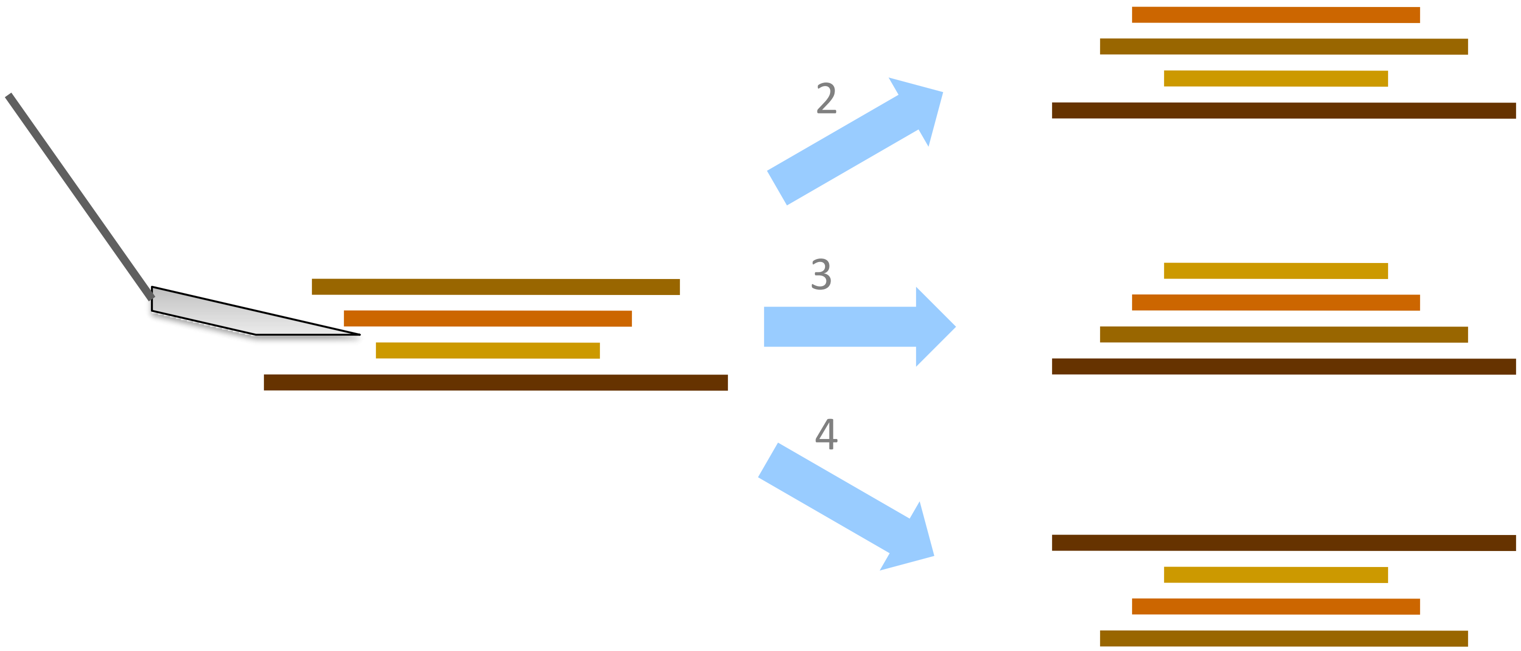




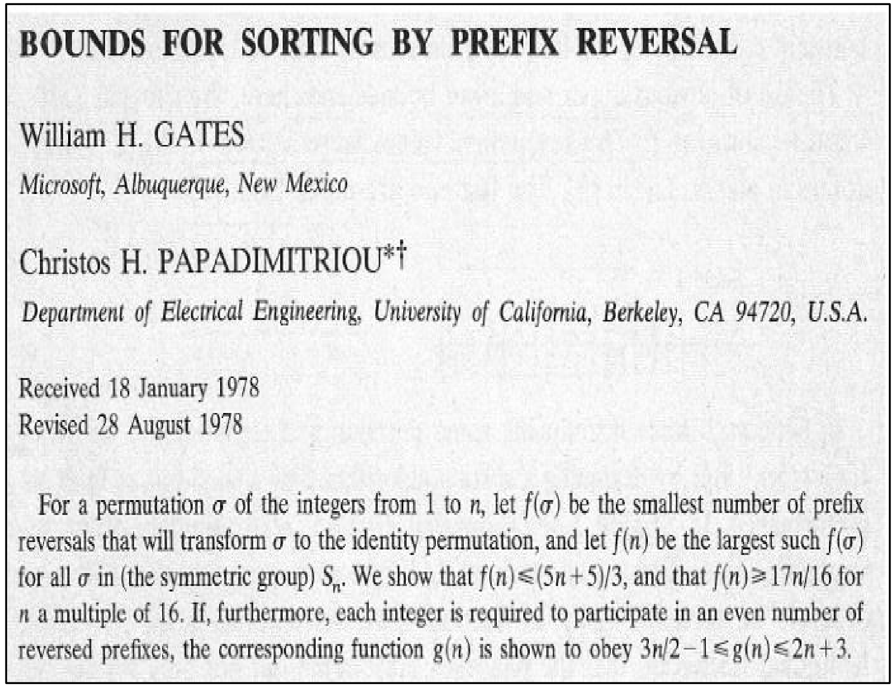


Example: Pancake Problem

Problem Formulation



Cost: Number of pancakes flipped



BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES

Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†

Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

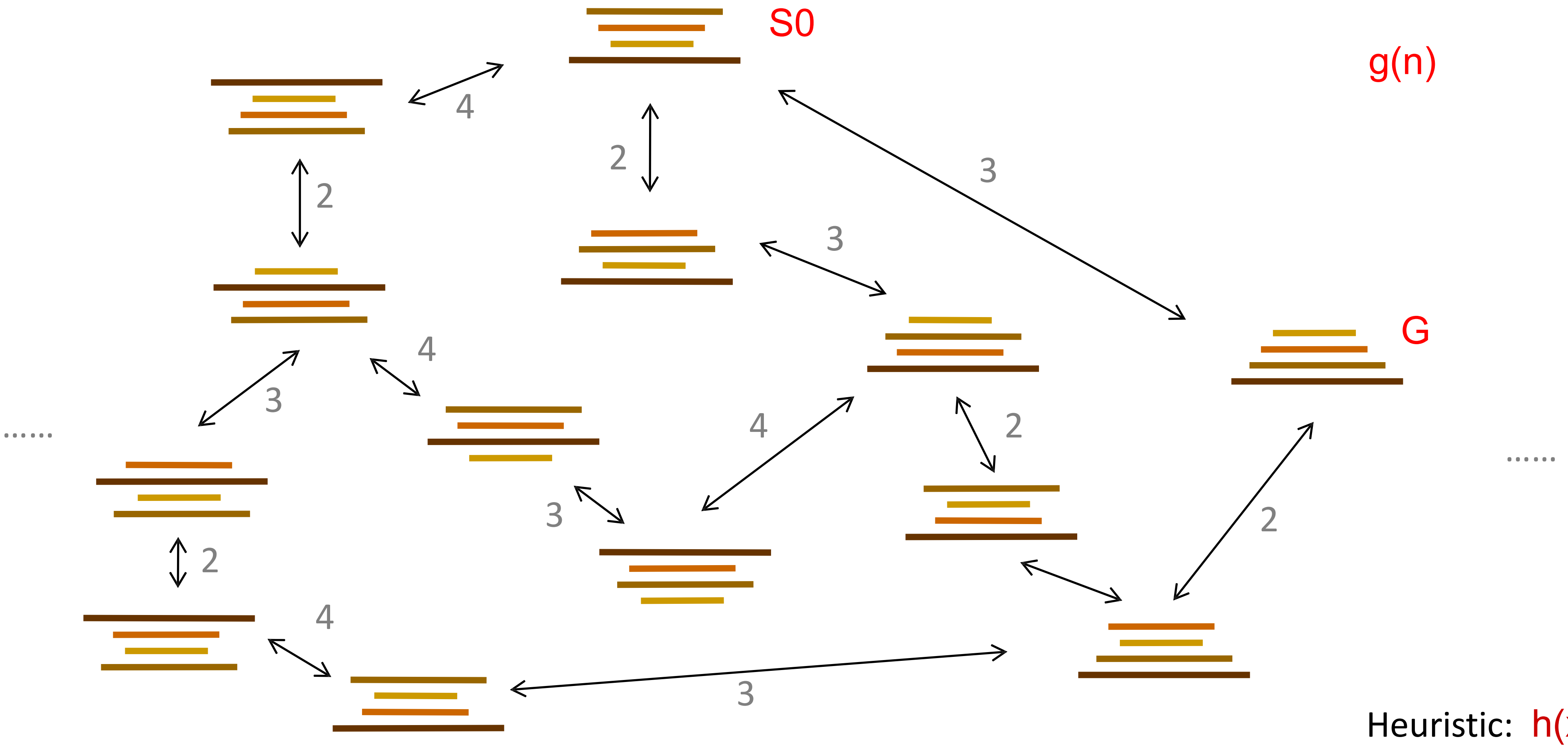
Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leq (5n+5)/3$, and that $f(n) \geq 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leq g(n) \leq 2n+3$.

Example: Pancake Problem

State space graph with costs as weights

Cost: Number of pancakes flipped

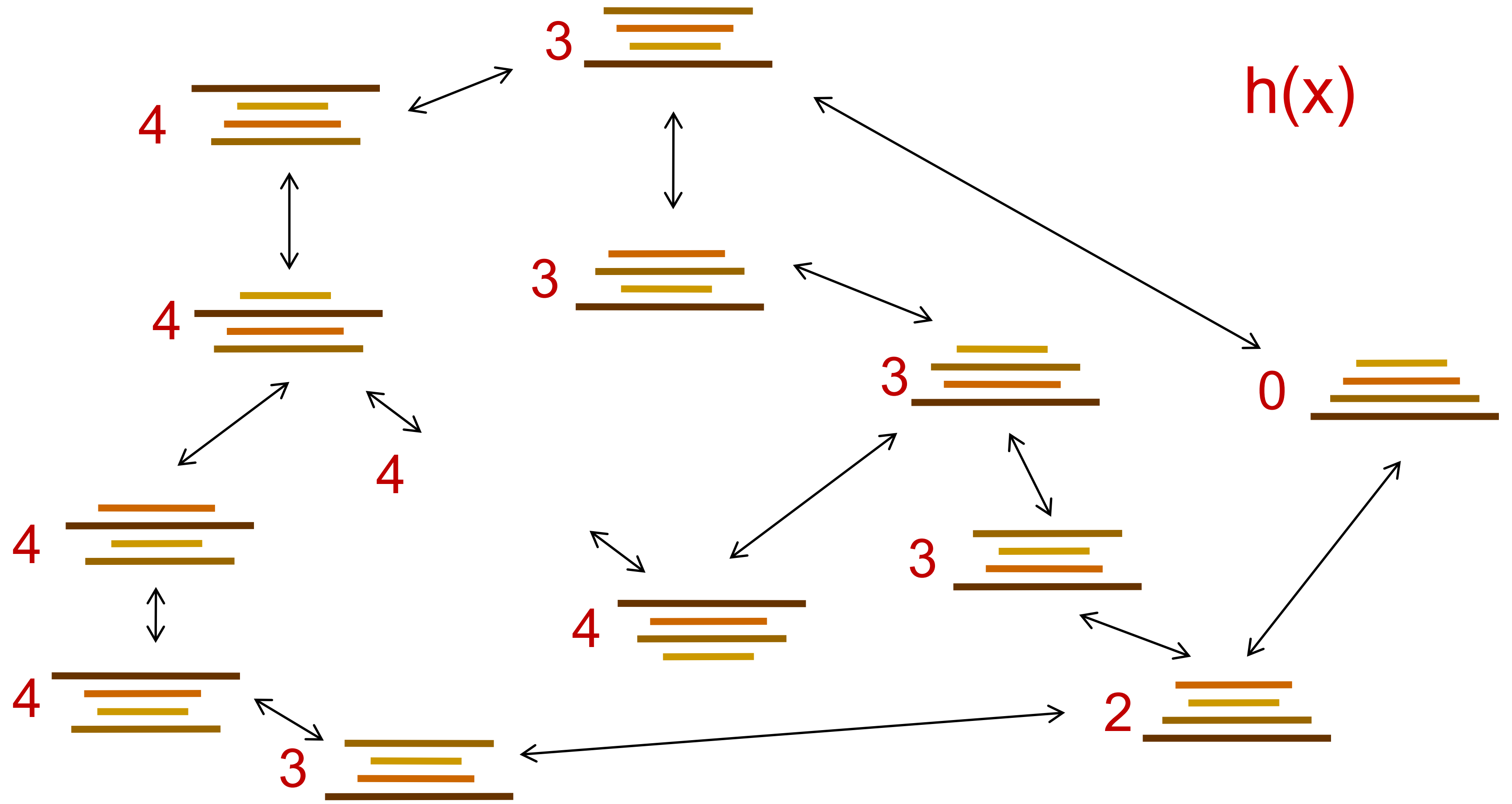


g(n)

Heuristic: h(x)?

Example: Heuristic Function

Heuristic: the number of the largest pancake that is still out of place



贪婪最佳优先搜索

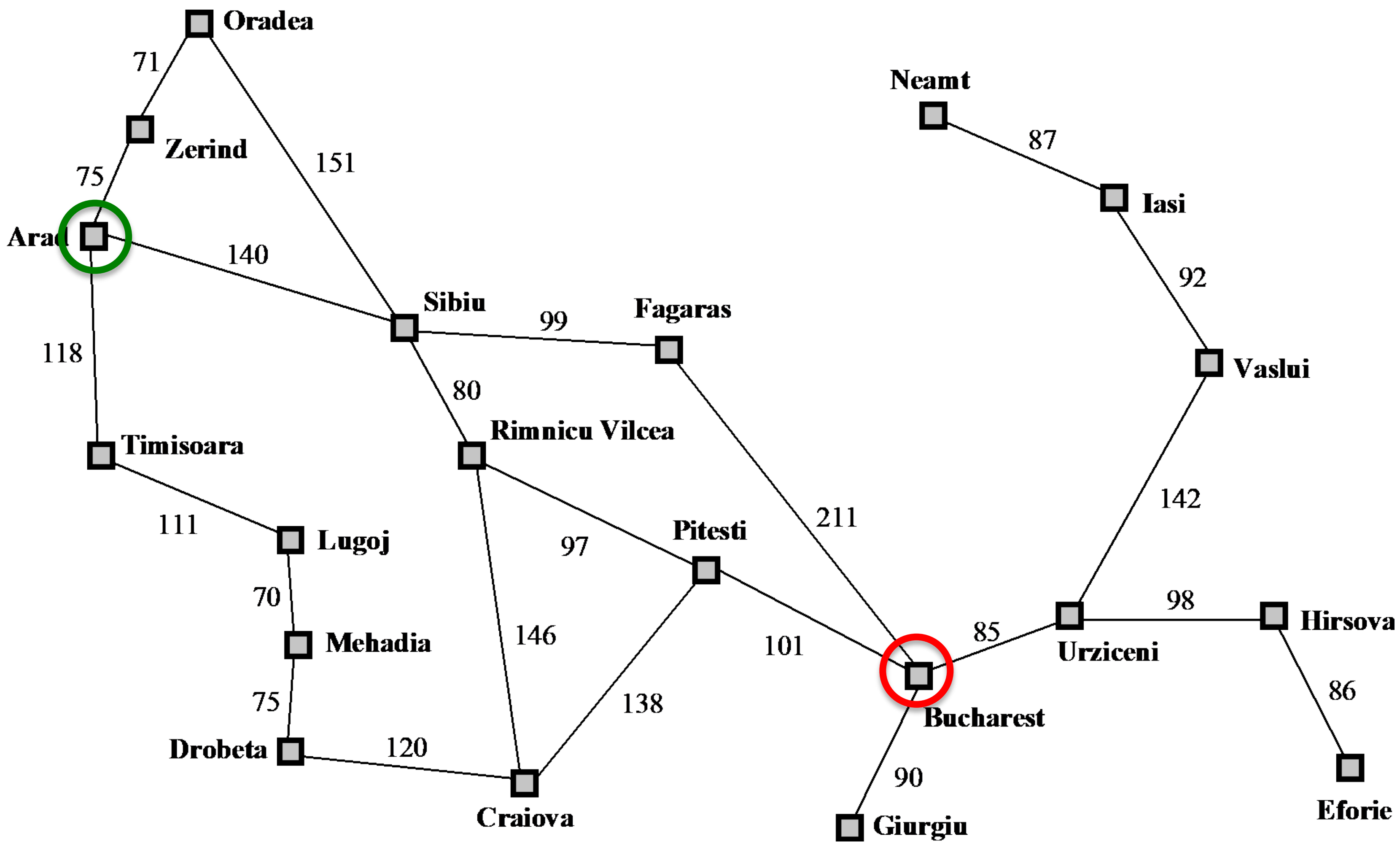
扩展离目标最近的结点，

可以很快的找到解。

只用启发式信息

评估函数f(n) = h(n)

Search Example: Romania



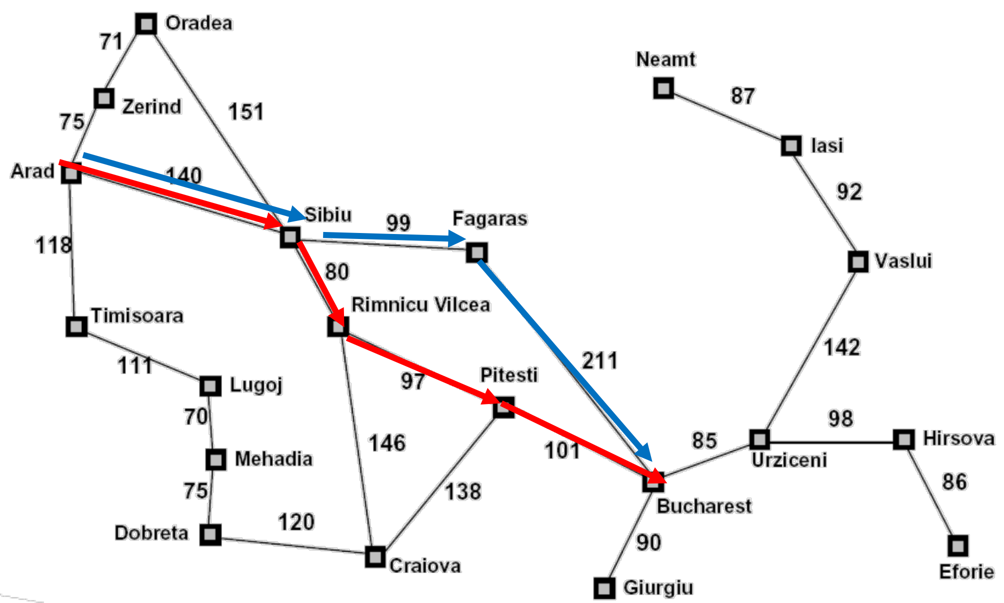
罗马尼亚问题：启发式信息（直线距离）

Straight−line distance
to Bucharest

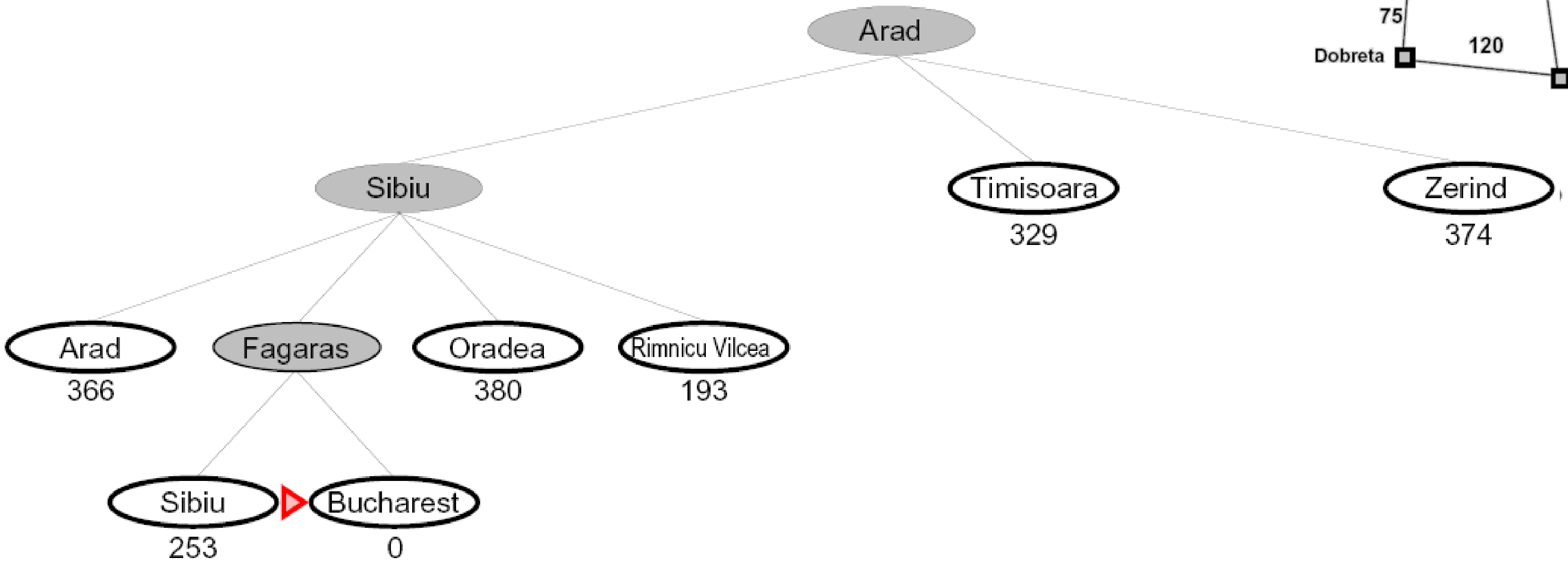
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

贪婪最佳优先搜索



- 贪婪搜索策略：
  - 优先扩展离目标最近的结点



Straight−line distance to Bucharest

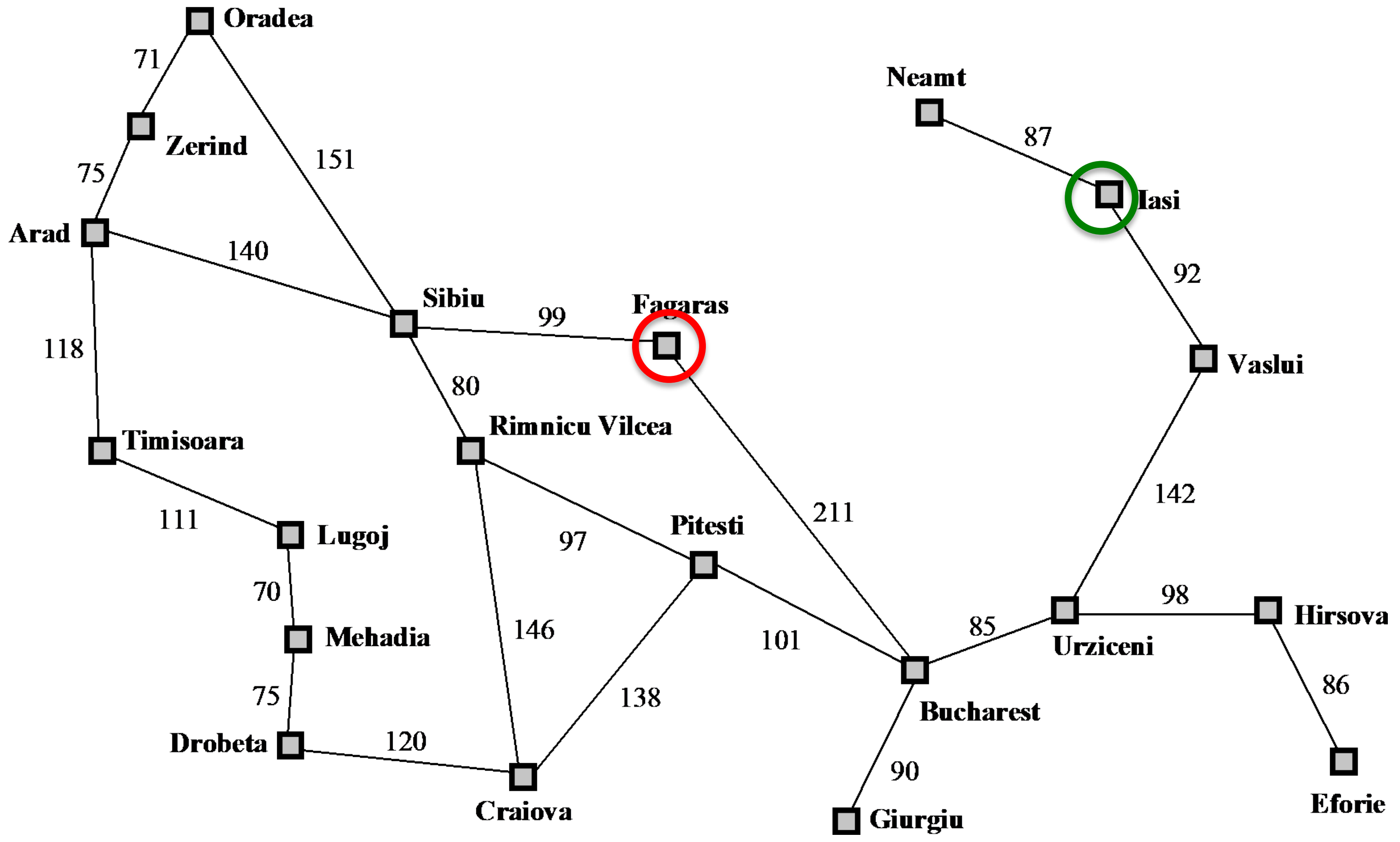
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

- 并不是最优的
  - 经过Rimniuc Vilcea 到 Pitesti 到Bucharest的路径要少32公里

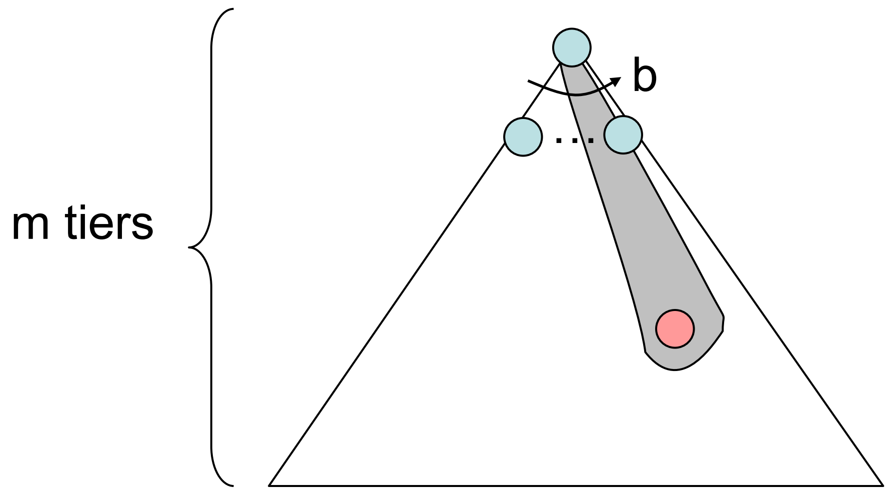
贪婪最佳优先搜索

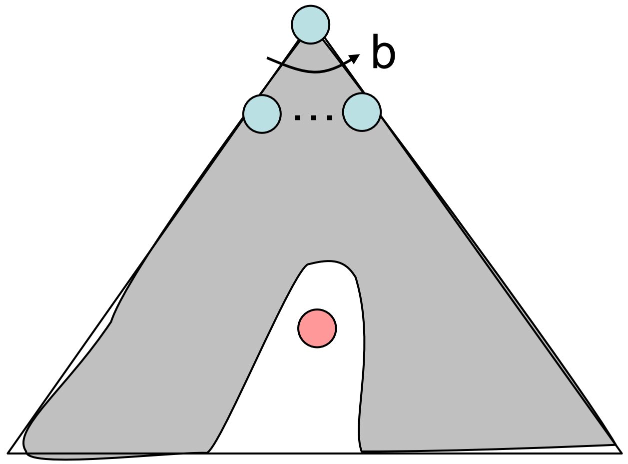
可能陷入死循环



贪婪最佳优先搜索的性能

- 完备性? 否–会陷入死循环
- 最优性? 否
- 时间复杂度? $O(b^m)$
- 空间复杂度? $O(b^m)$
- 一个好的启发式函数可以有效降低复杂度。





A* Search

A* 搜索

- 思路：避免扩展耗散值已经很大的路径
- 评估函数f(n) = g(n) + h(n)，经过结点n的最小代价解的估计代价
  - 代价函数g(n) = 从初始结点S到达结点n已经花费的代价（实际代价）
  - 启发式函数h(n) = 从结点n到目标结点G的最小代价路径的估计值
- 搜索策略：优先级队列，先扩展f(n)的值最小的结点

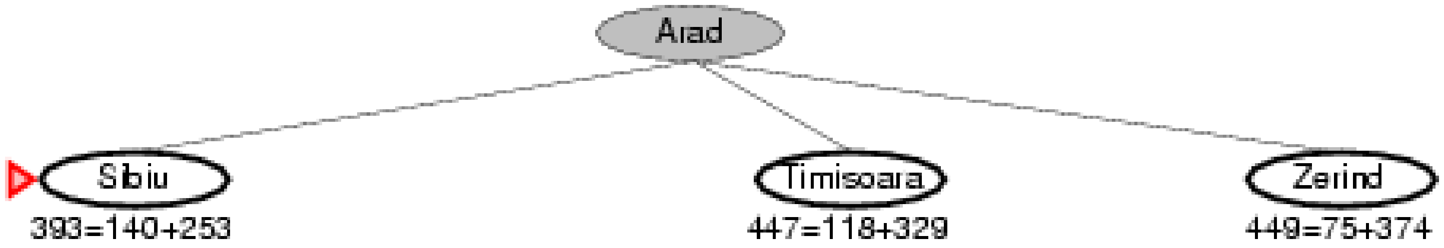
P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths in graphs. IEEE Trans. Syst. Sci. and Cybernetics, SSC-4(2):100-107, 1968

A^* 搜索-案例

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

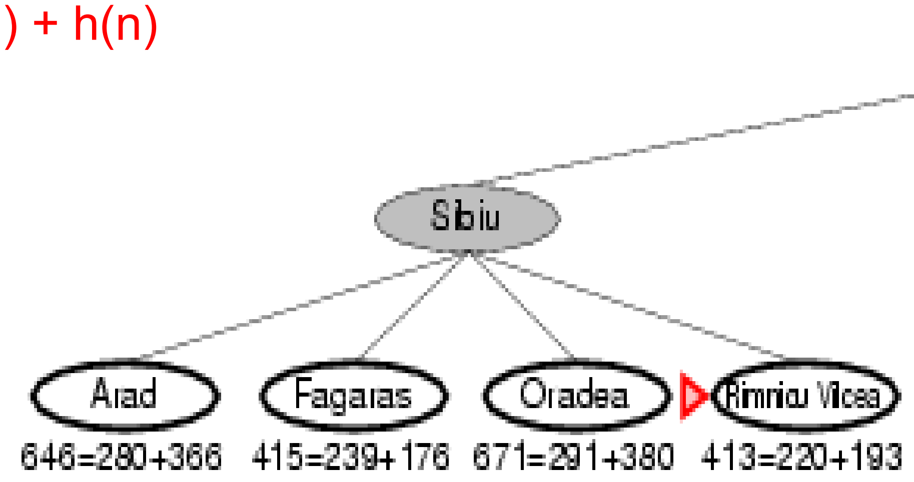
A* 搜索-案例

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



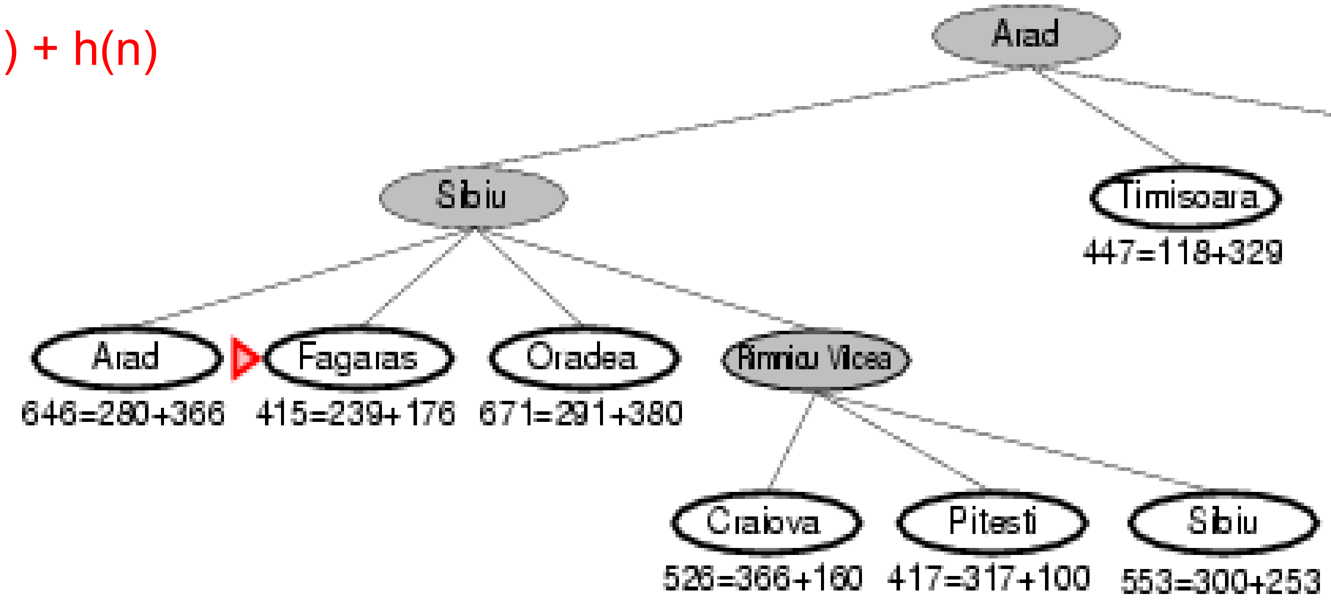
A* 搜索-案例

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



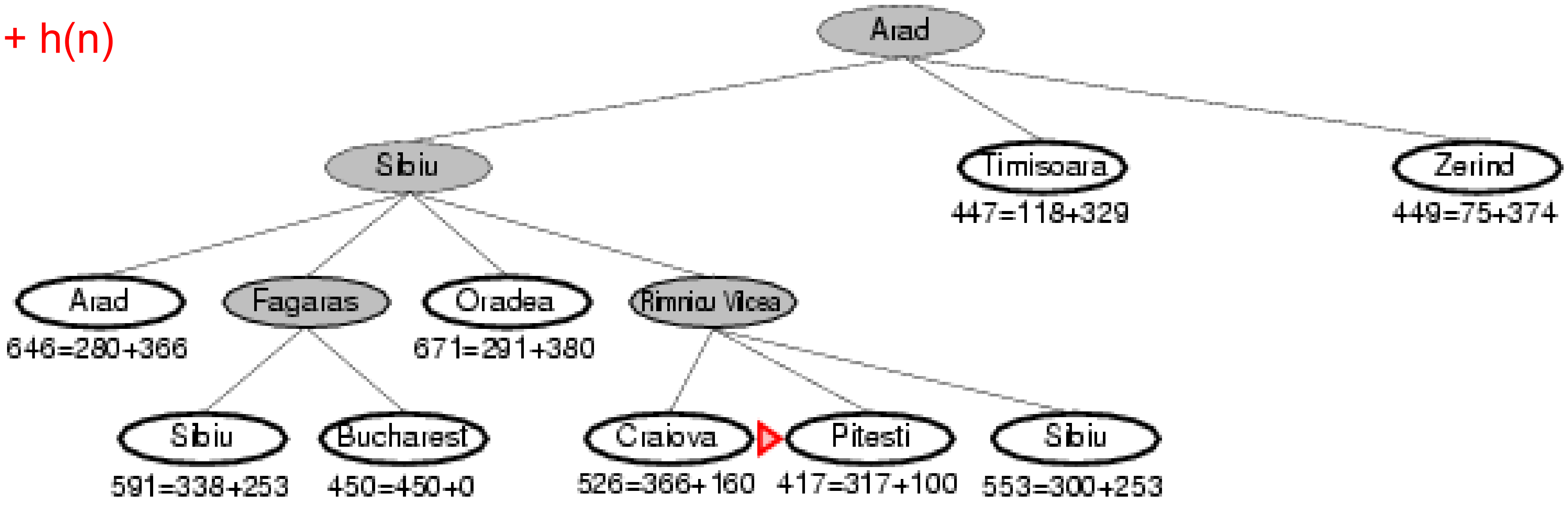
A* 搜索-案例

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



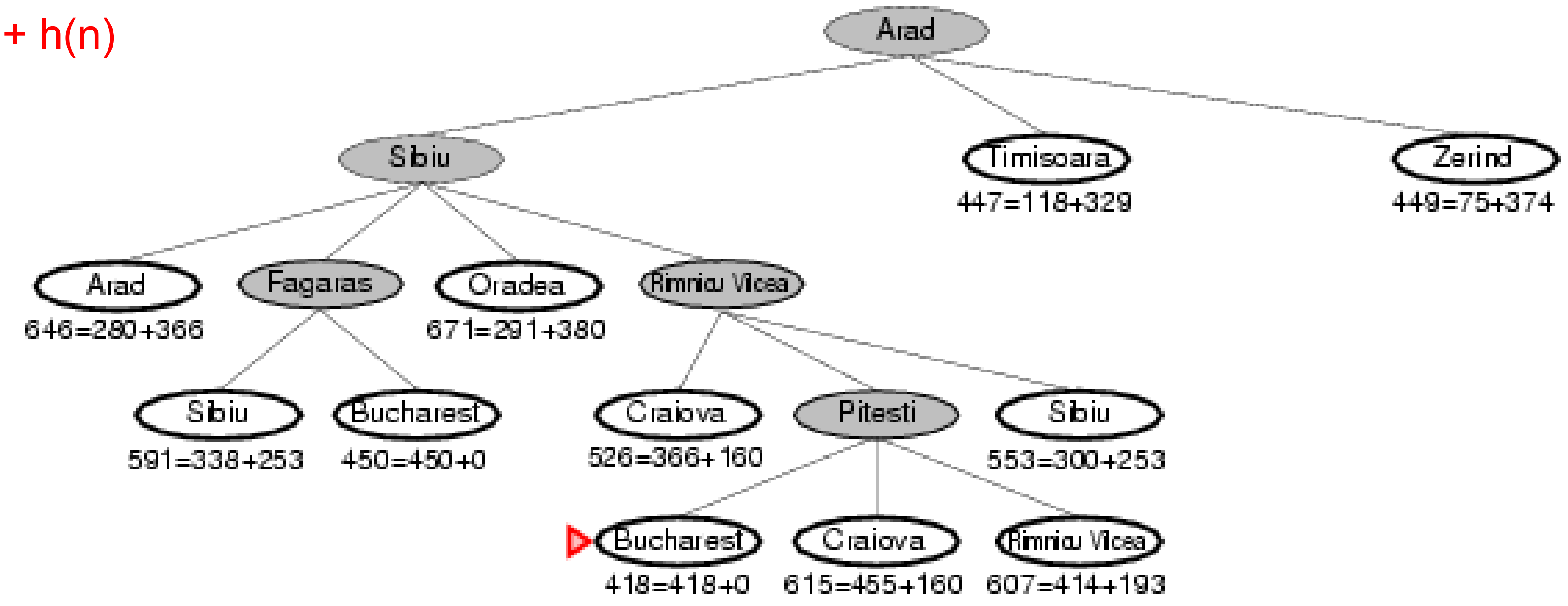
A^* 搜索-案例

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



A* 搜索-案例

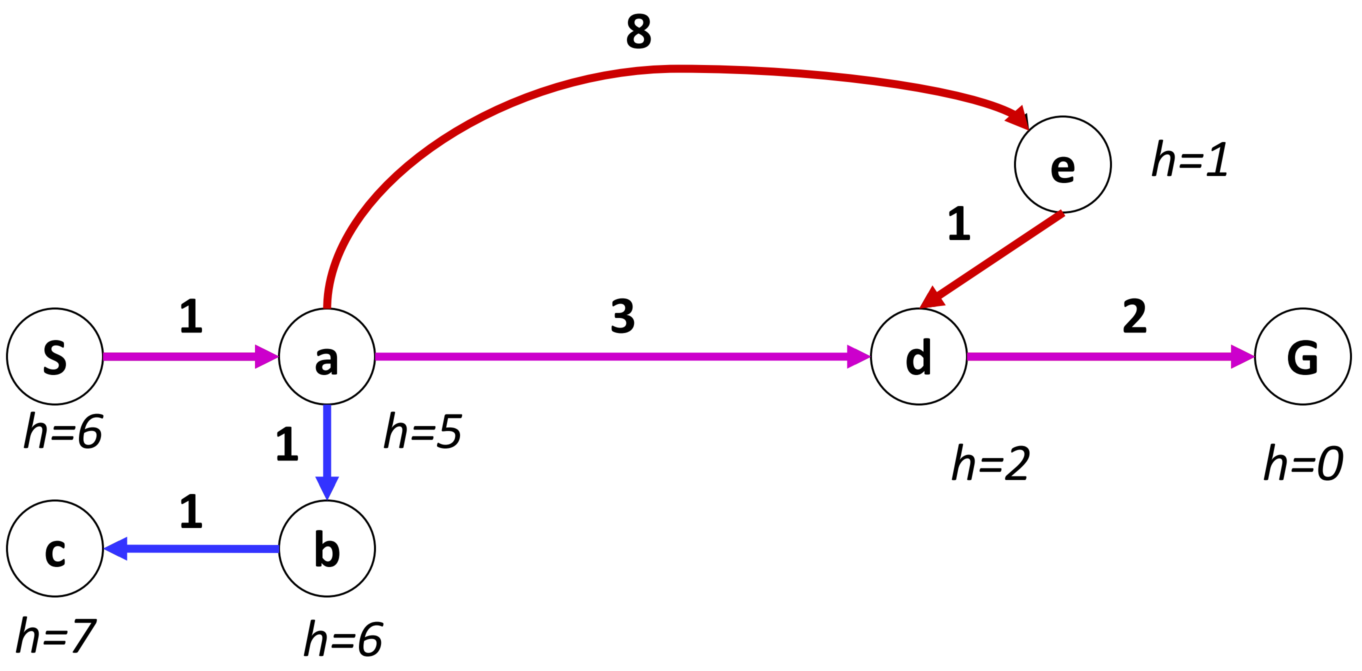
$f(n) = g(n) + h(n)$

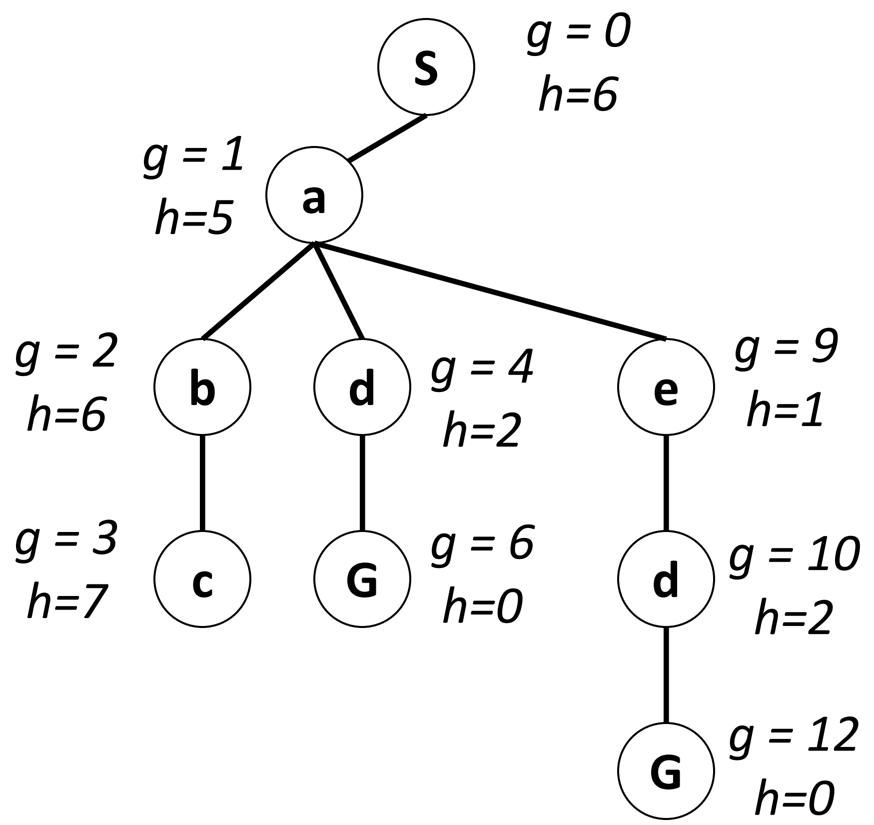


A* 搜索找到的解，是不是最优解？

课堂练习：UCS 、 Greedy、 A* 搜索

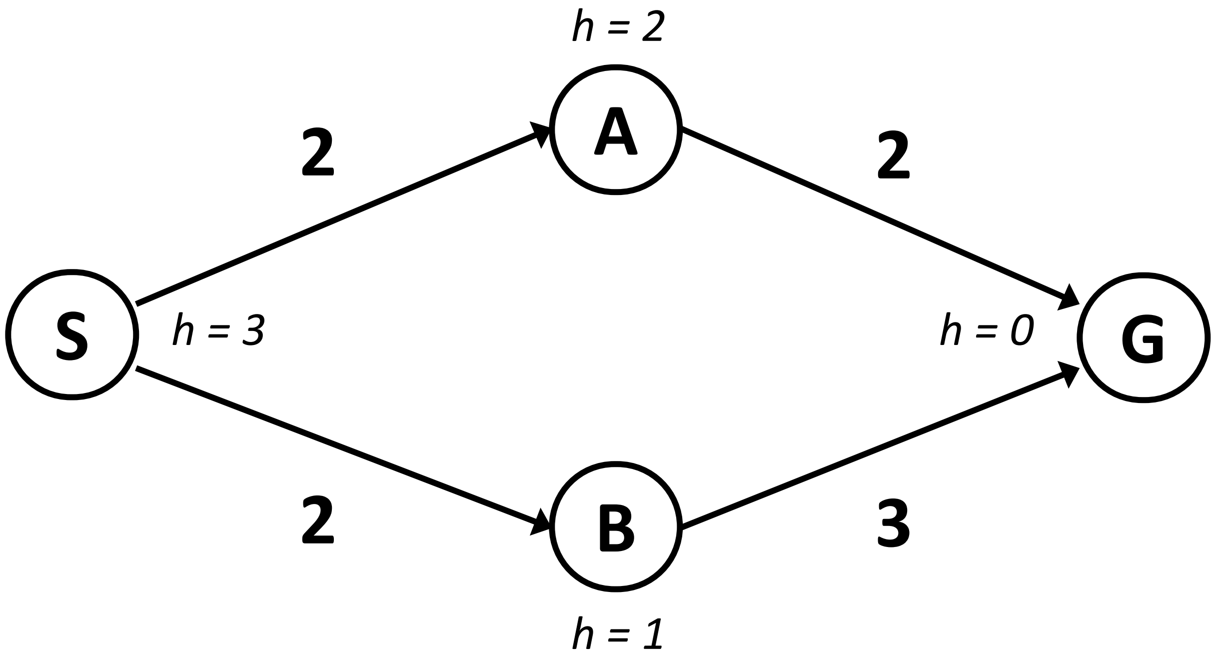
- UCS： *backward cost* g(n) Dijkstra算法1959
- Greedy： *forward cost* h(n)





- A* 搜索: f(n) = g(n) + h(n)

When should A* terminate?



Fringe

S(3)

B(3) A(4)

A(4)G(5)

G'(4)G(5)

- 当目标移除队列时算法终止

Is A* Optimal?

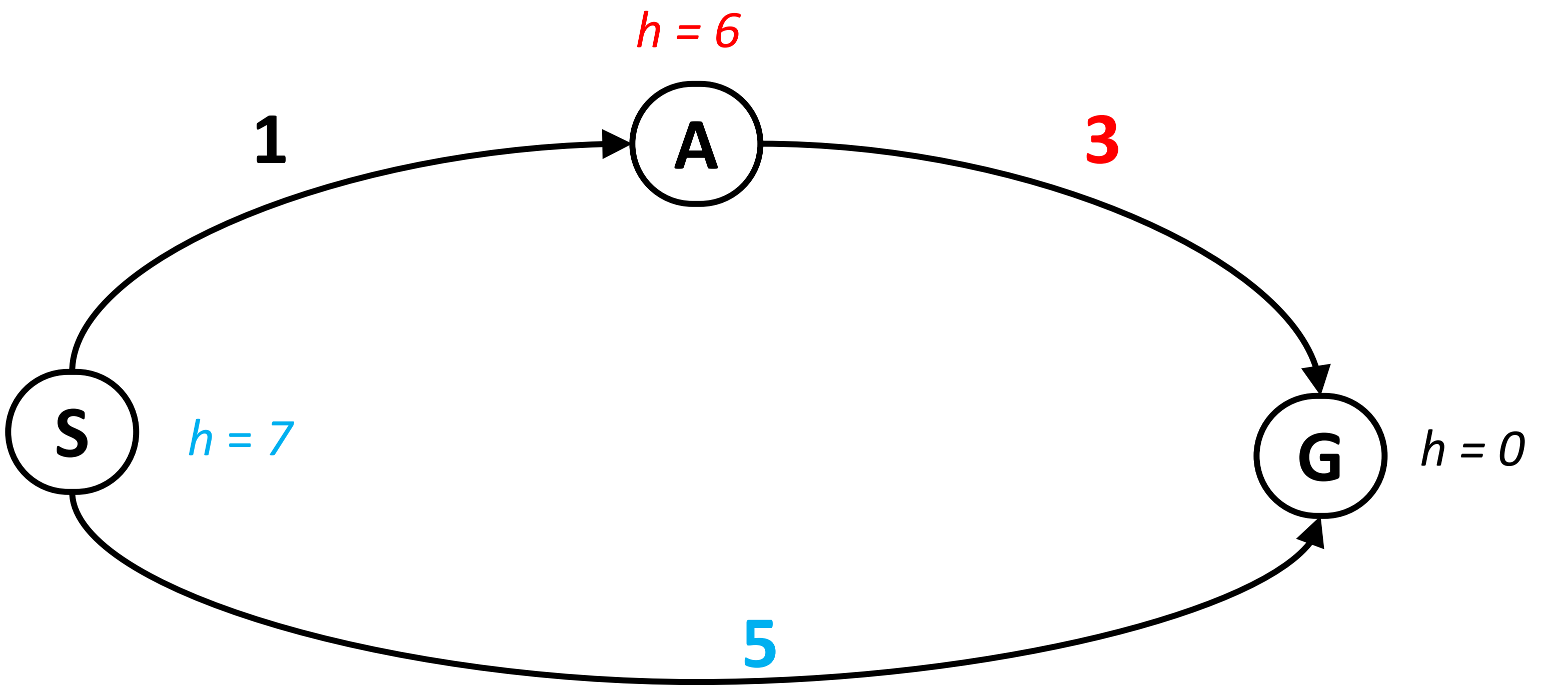
Fringe

S(7)

G(5) A(7)

解序列：S G， 耗散：5

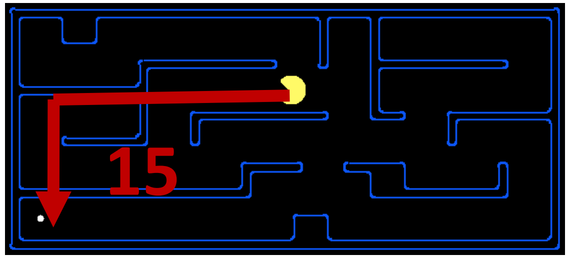
最优解：S A G， 耗散：4



- Yes, only if 估计的目标代价 h(n)<= 实际的目标代价h*(n)

可采纳启发式

- 若对每个结点n，满足$0 \le h(n) \le h^*(n)$,则 $h(n)$是**可采纳的**。
  - 其中 $h^*(n)$ 是从结点n到达目标结点的<u>真实代价</u>。
- 可采纳启发式不会过高估计到达目标的代价。
  - 例如: $h_{SLD}(n)$ (不会高估真实距离)



- **定理**: 如果$h(n)$是可采纳的, A*<u>搜索树搜索算法</u>是最优的。

A*搜索分析

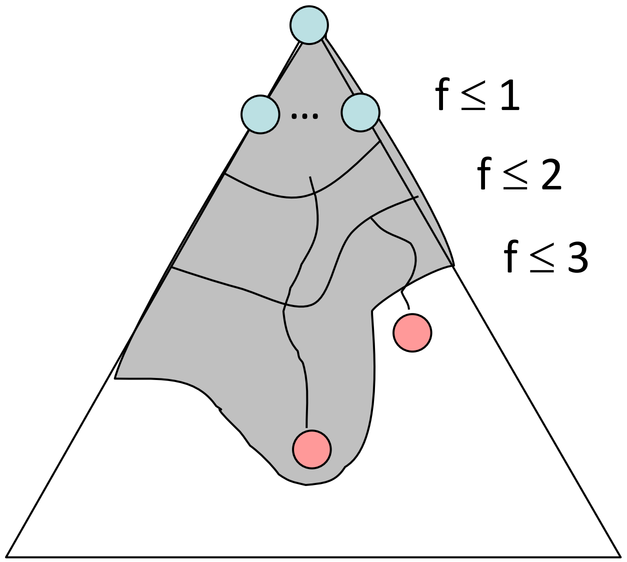
保证找到最短路径（最优解）的条件，关键在于评估函数f(n)的选取（或者说h(n)的选取）。

以h*(n)表达状态n到目标状态的真实代价，那么h(n)的选取有如下四种情况：

- 1. 如果h(n)=0, 一致代价搜索，能得到最优解
- 2.如果h(n)= h*(n)，搜索将严格沿着最优解路径进行， 此时的搜索效率是最高的。
- 3. 如果0<h(n)< h*(n)，搜索的点数多，搜索范围大，能得到最优解。
- 4.如果 h(n)> h*(n)，搜索的点数少，搜索范围小，但<u>不能保证得到最优解</u>。

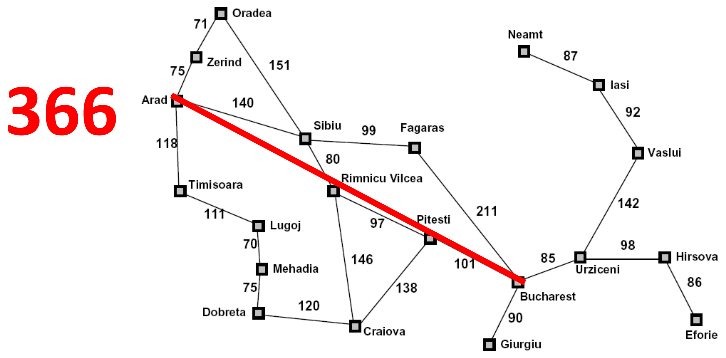
A*搜索的性能

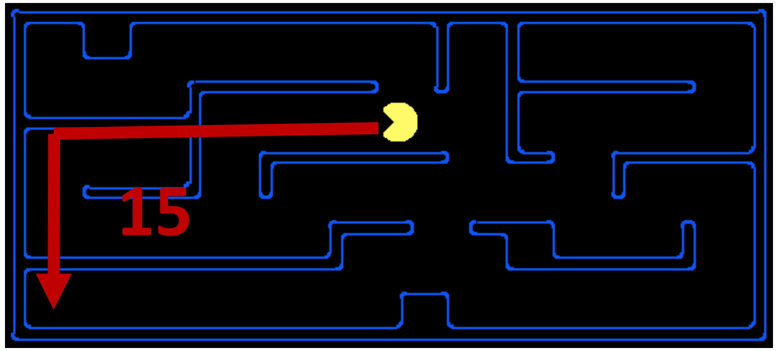
- 完备性? 是 (if finite nodes)
- 最佳性? 是 （ *if* $h(n)$是可采纳的: $0 \leq h(n) \leq h^*(n)$）
- 时间复杂度? 指数级
- 空间复杂度?指数级Keeps all nodes in memory



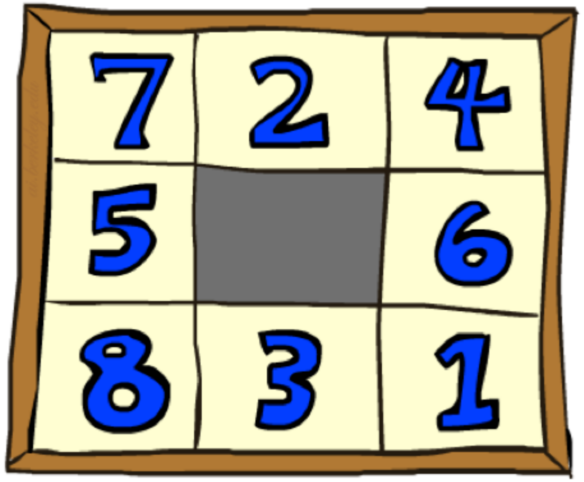
3.6 启发式函数

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics
- Often, admissible heuristics are solutions to *relaxed problems,* where new actions are available

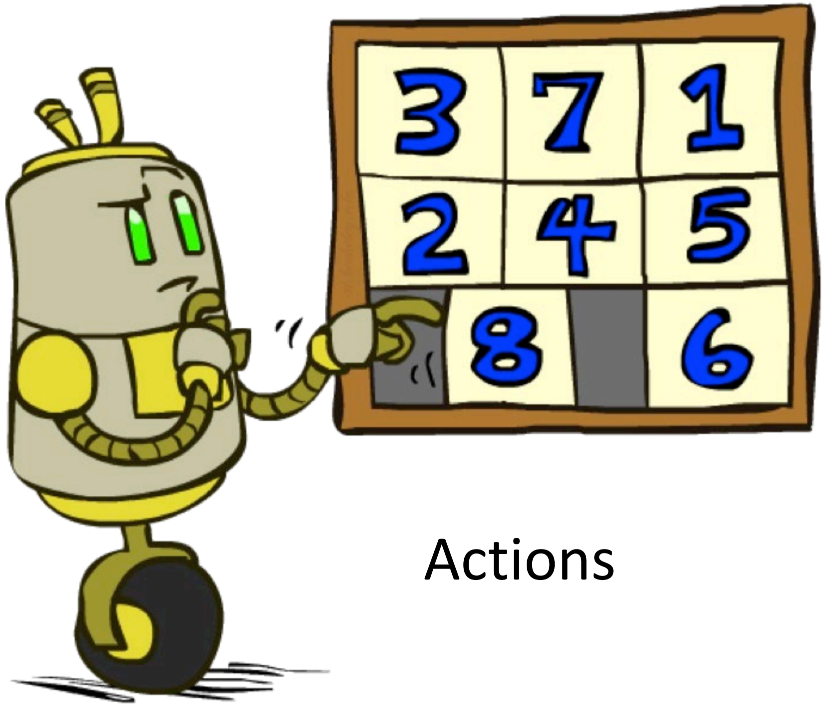




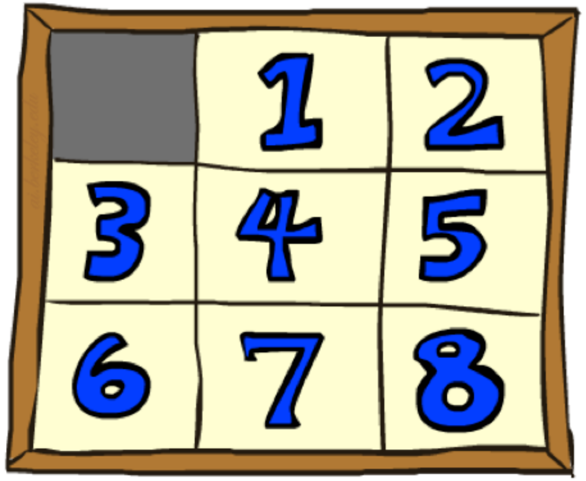
Example: 8 Puzzle



Start State



Actions



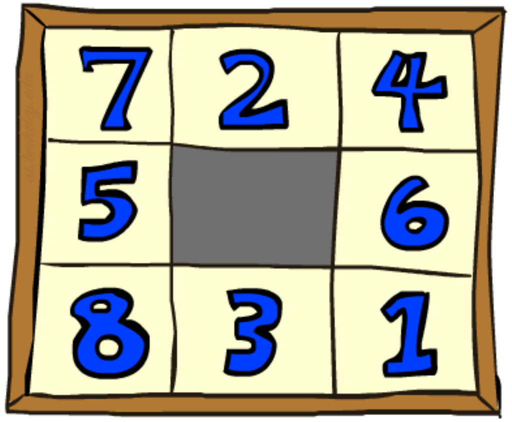
Goal State

- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?

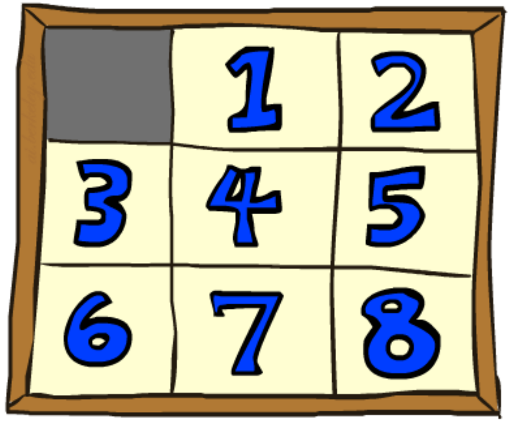
可采纳的启发式

E.g., 针对八数码问题，有两个常用的可采纳的启发式函数:

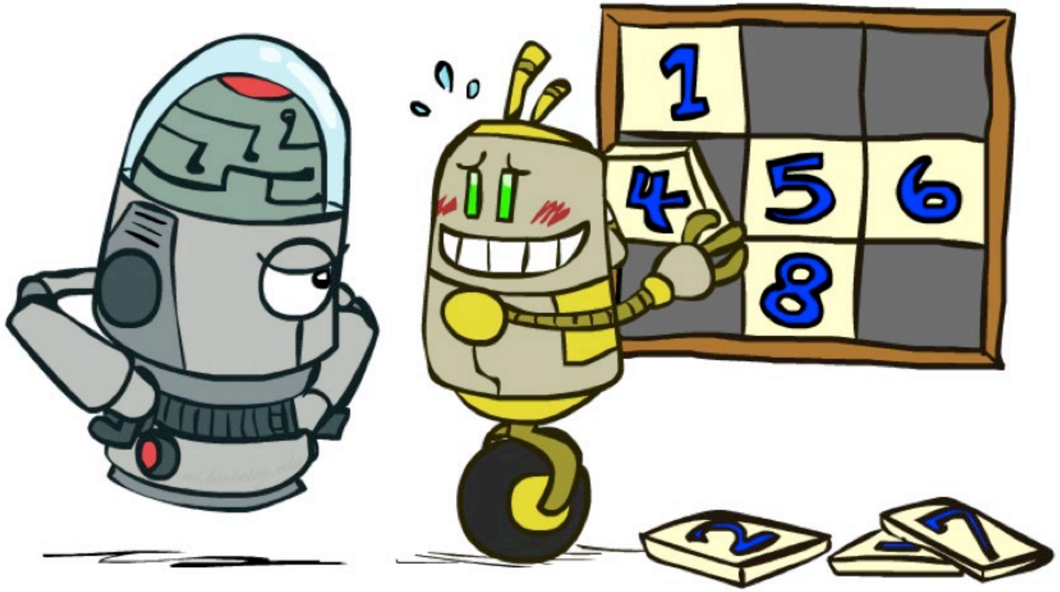
- $h_1(n)$ = 不在位的棋子数
- $h_2(n)$ =所有棋子到其目标位置的距离和



Start State

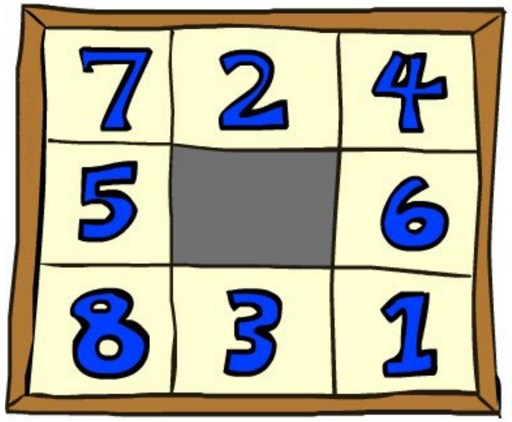


Goal State

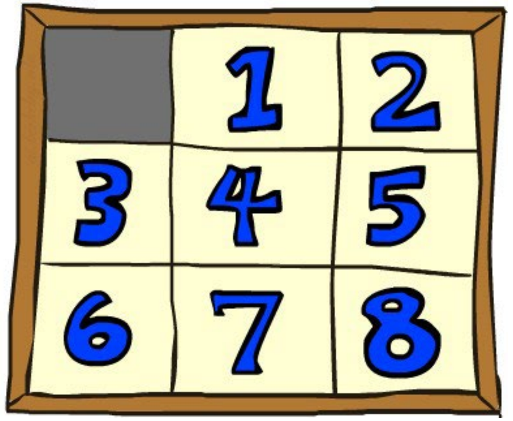


8 Puzzle I

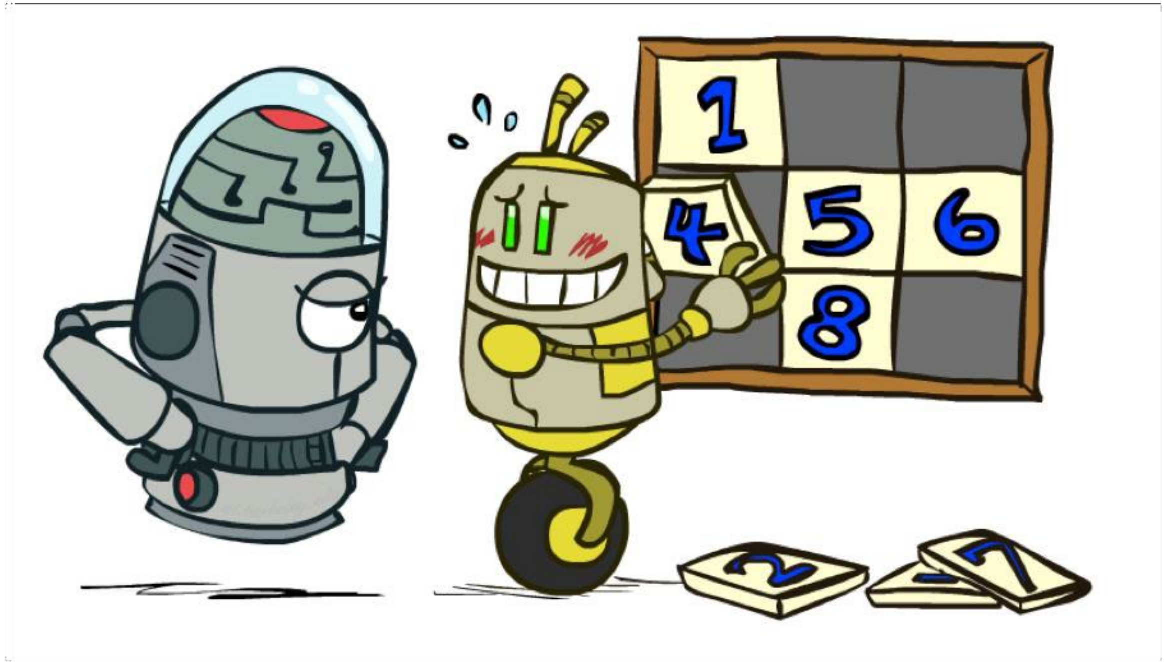
- Heuristic: $h_1(n)$ = 不在位的棋子数
- Why is it admissible?
- h(start) = 8



Start State

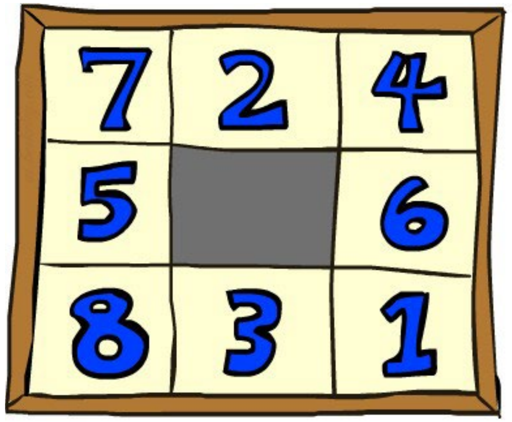


Goal State

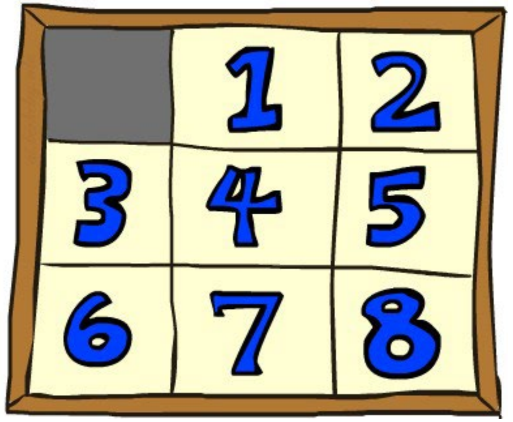


8 Puzzle II

- Heuristic:
- $h_2(n)$ =所有棋子到其目标位置的距离和
- Why is it admissible?
- h(start) =   3 + 1 + 2 + … = 18



Start State

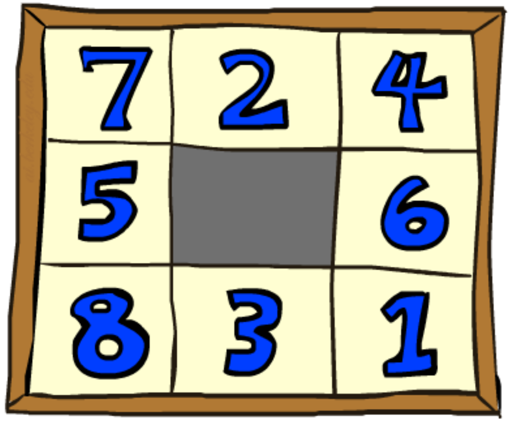


Goal State

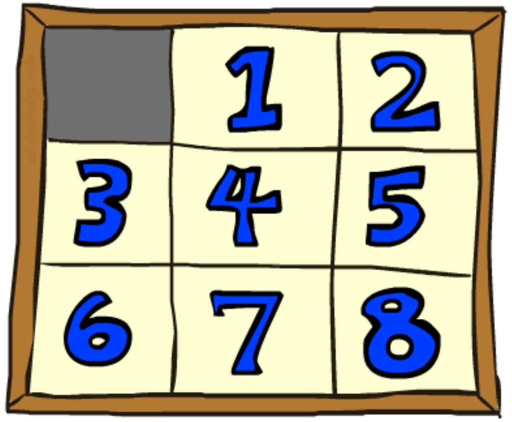
可采纳的启发式

E.g.,针对八数码问题，有两个常用的可采纳的启发式函数:

- $h_1(n)$ = 不在位的棋子数
- $h_2(n)$ =所有棋子到其目标位置的距离和



Start State



Goal State

- h1(start) = 8
- h2(start) = 3+1+2+2+2+3+3+2 = 18

- **h*(S) = 26 (上述两类启发式函数都没有高估到达目标的实际代价)**

启发式函数性能对比

- 对于任意结点*n*， 若$h_2(n) \geq h_1(n)$，称h_2比h_1占优势，h_2 更利于搜索。
- 搜索代价(扩展的平均结点数):
  - *d=12* IDS = 3,644,035 nodes
  - $A^*(h_1)$ = 227 nodes

    $A^*(h_2)$ = 73 nodes
  - *d=24* IDS = too many nodes

    $A^*(h_1)$ = 39,135 nodes

    $A^*(h_2)$ = 1,641 nodes

	搜索代价			有效分支因子		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

设计接近又总是小于等于$h^*(n)$的$h(n)$是应用A^*算法($h(n) \leq h^*(n)$)搜索问题解答的关键

A*搜索的应用

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …

总结：搜索策略

无信息搜索策略: 只使用问题定义中提供的状态信息

- 宽度优先搜索
- 一致代价搜索
- 深度优先搜索
- 深度受限搜索
- 迭代加深搜索

有信息搜索策略: 使用启发式信息指导搜索

- 贪婪最佳优先搜索
- A*搜索