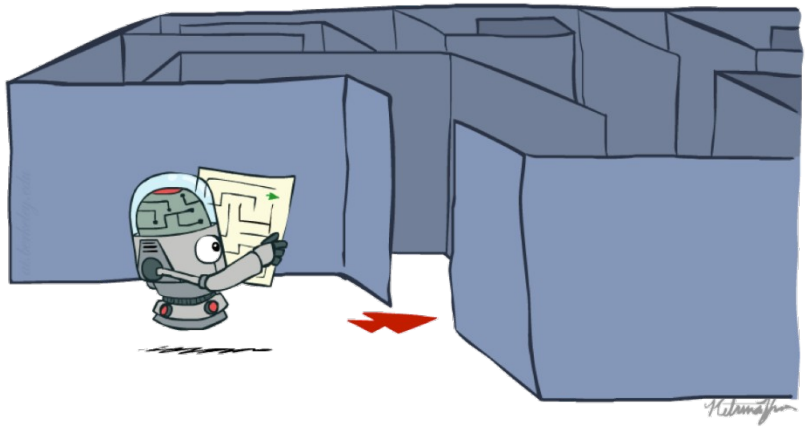
3.5 有信息（启发式）搜索策略

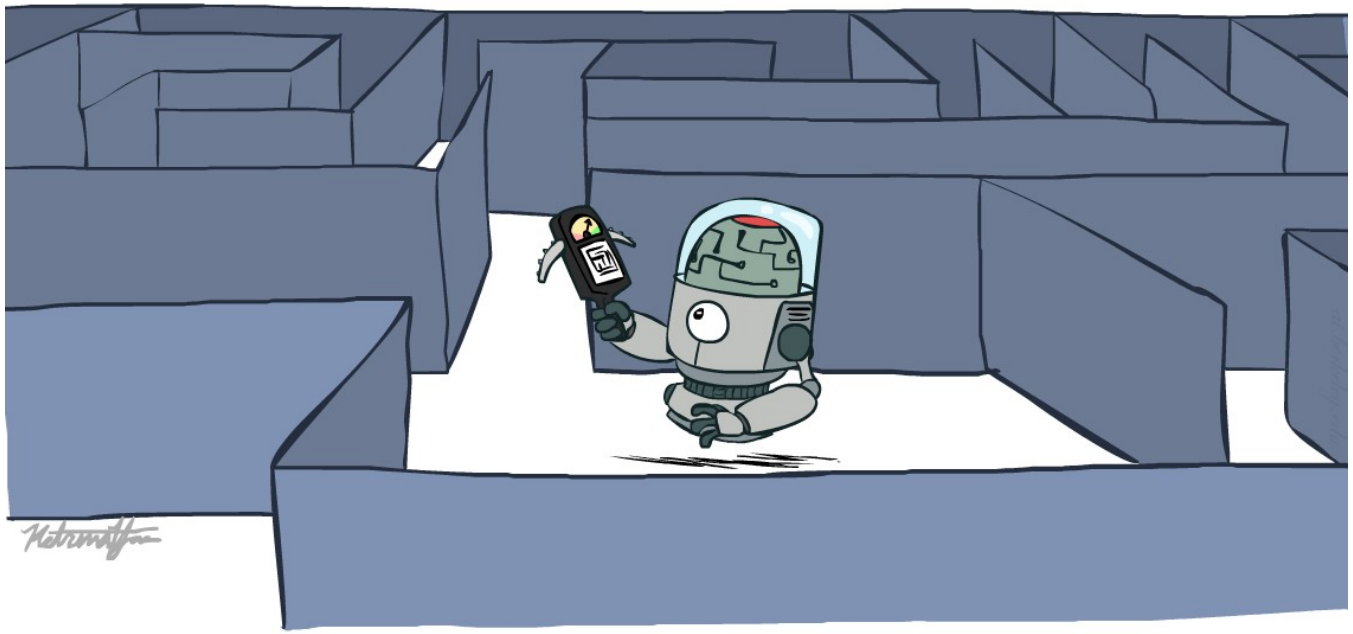


无信息搜索

（除了问题本身外，没有任何额外的信息）

有信息搜索

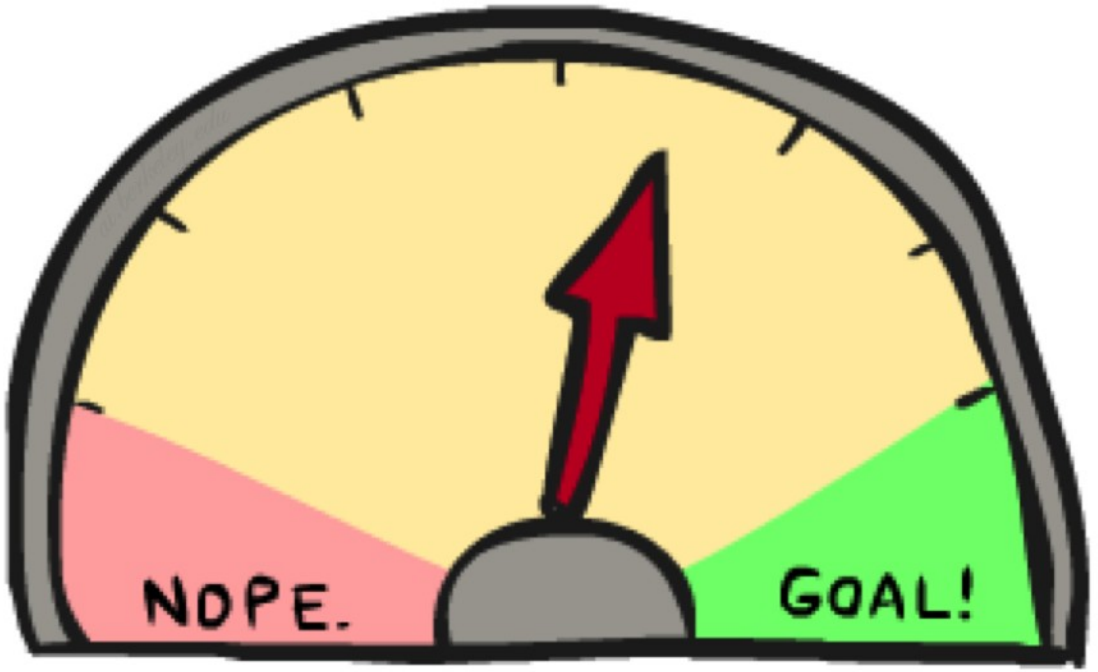
（除了问题本身外，还有启发式信息）



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

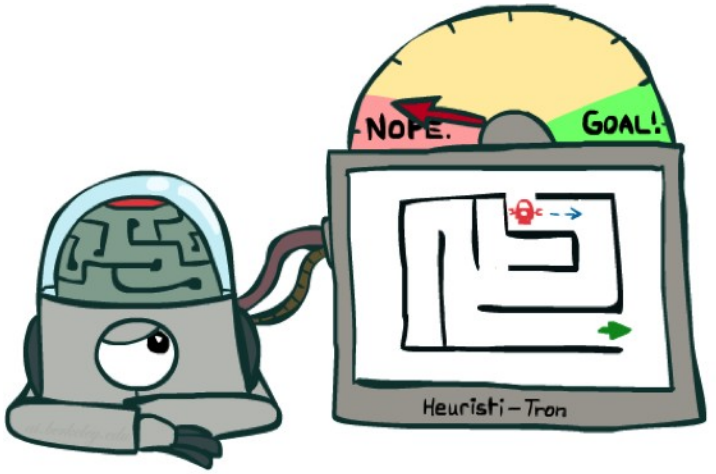
目录

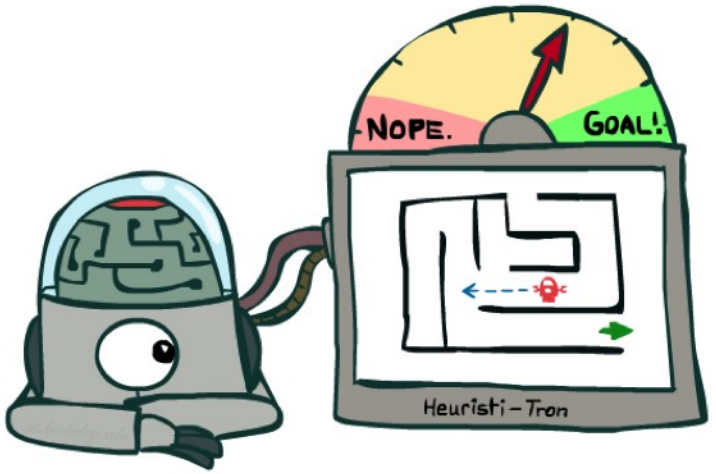
- 启发式函数
- 有信息搜索策略
  - 贪婪最佳优先搜索
  - A* 搜索



最佳优先搜索

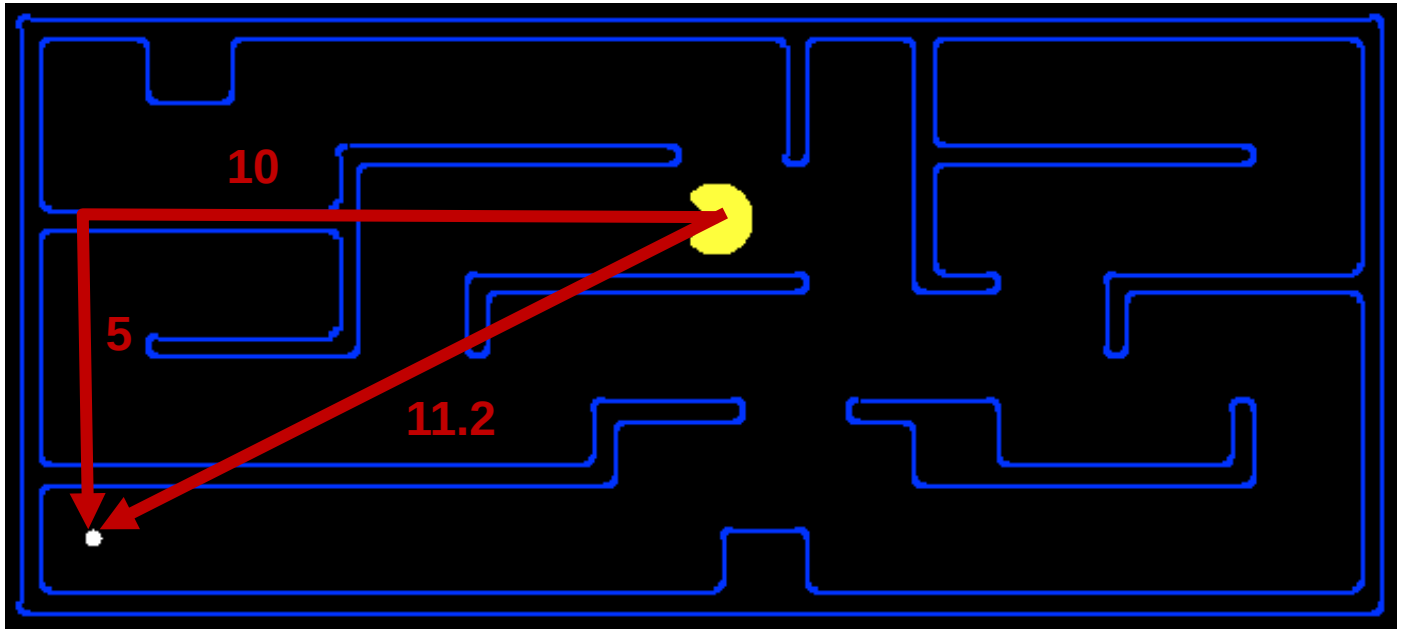
- 想法：创建一个评估函数 f(n) 用于每个结点
  - 评估“可取性”，确定哪个结点最有可能在通向目标的最佳路径上
  - 搜索策略：优先级队列
  - 扩展最可取的结点，总是选择“最有希望”的结点作为下一个被扩展的结点

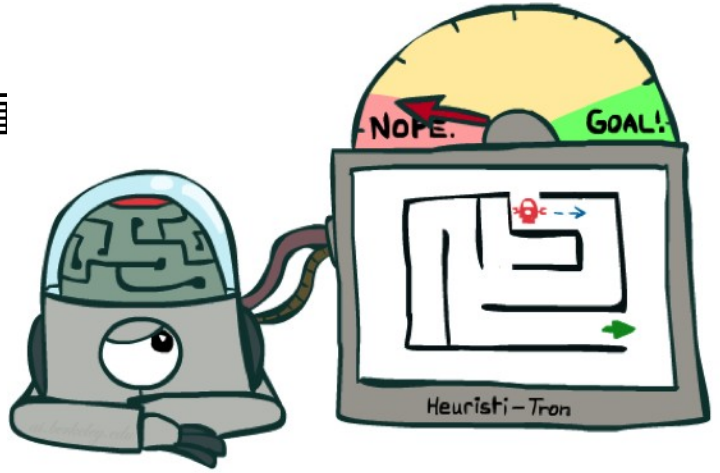


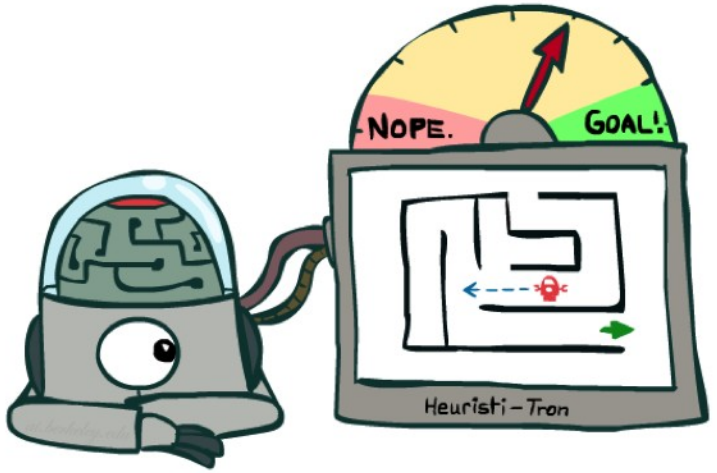


启发式函数

- 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







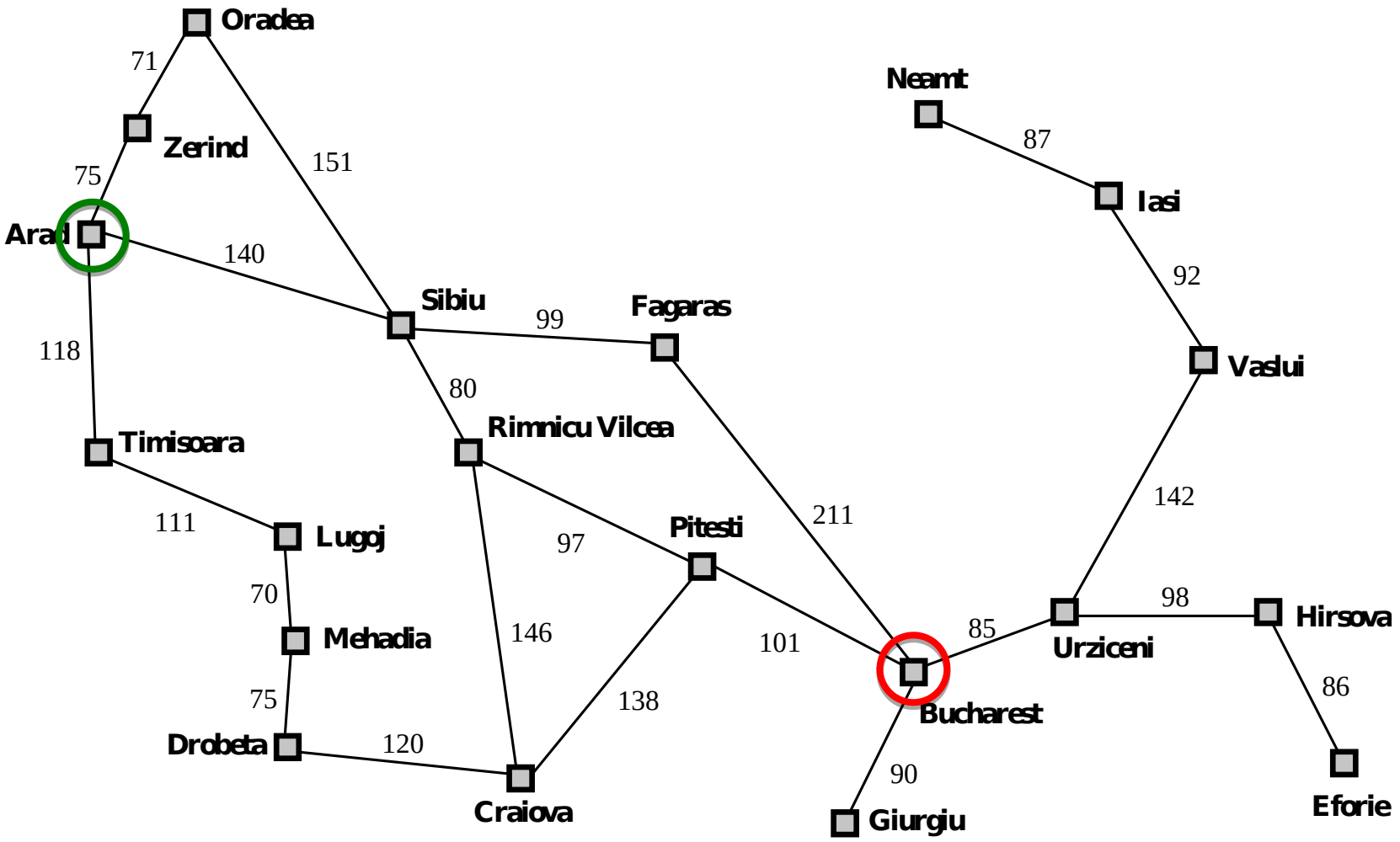
贪婪最佳优先搜索

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

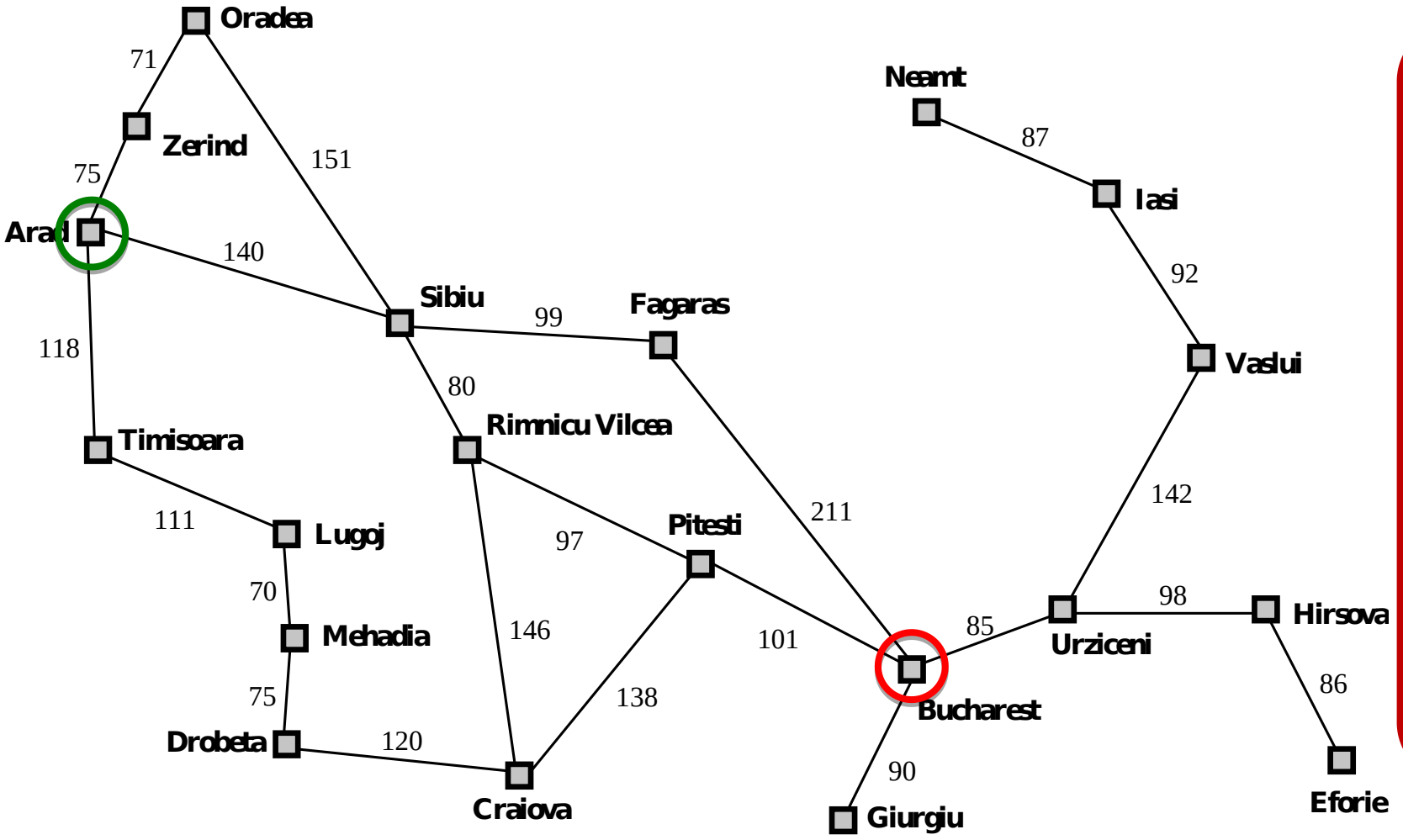
评估函数 f(n) = h(n)

Search Example: Romania



罗马尼亚问题

Search Example: Romania



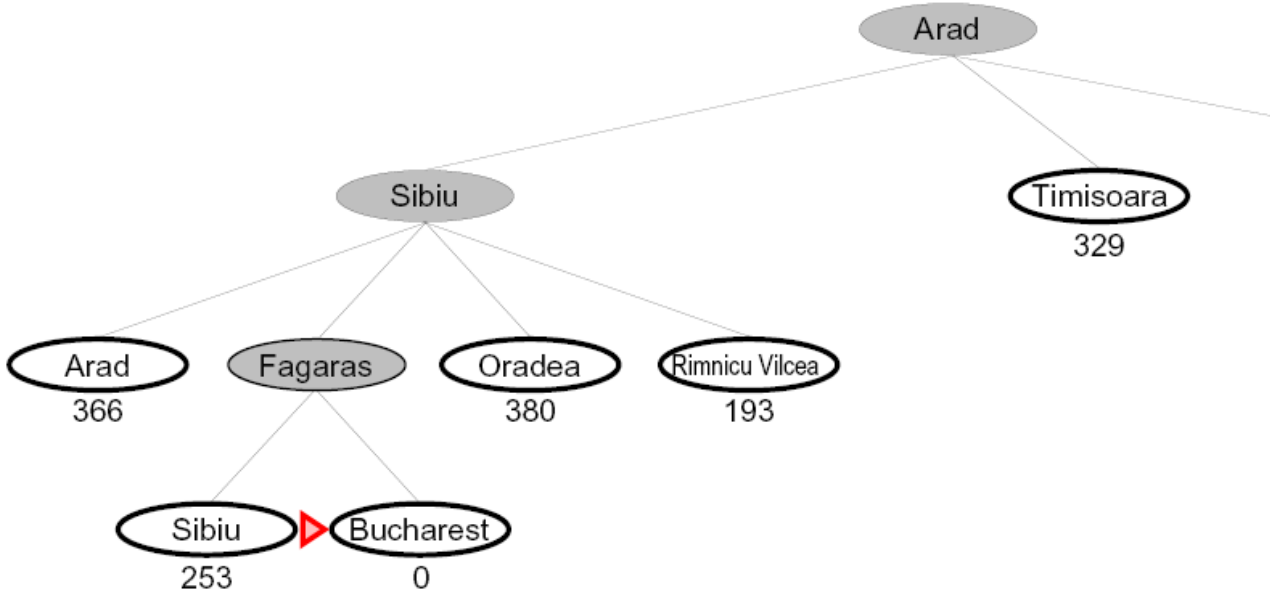
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

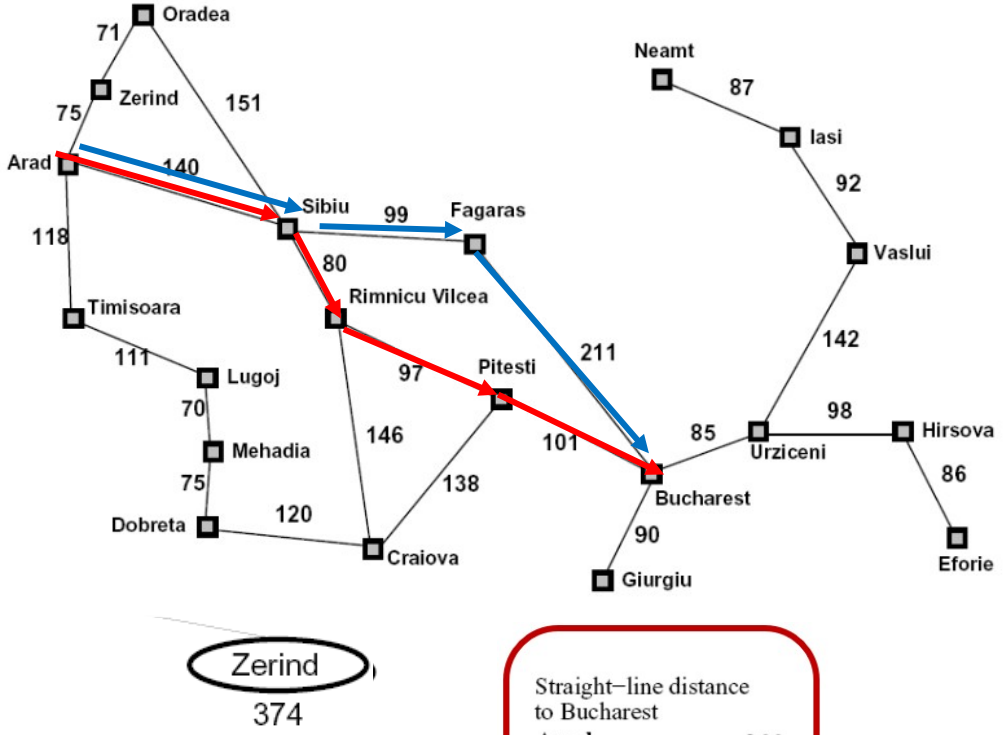
h(x)

罗马尼亚问题：启发式信息（直线距离）

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





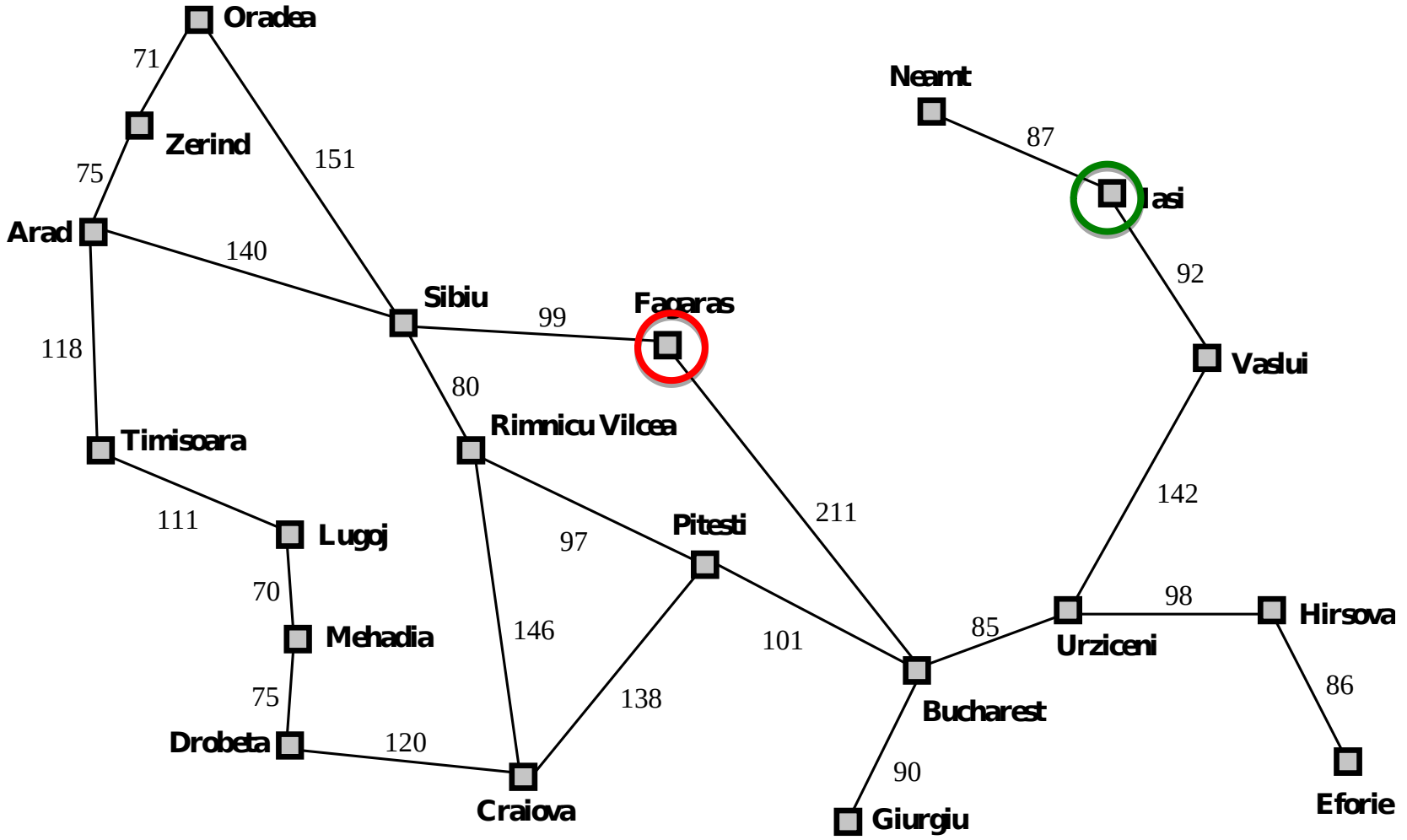
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

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

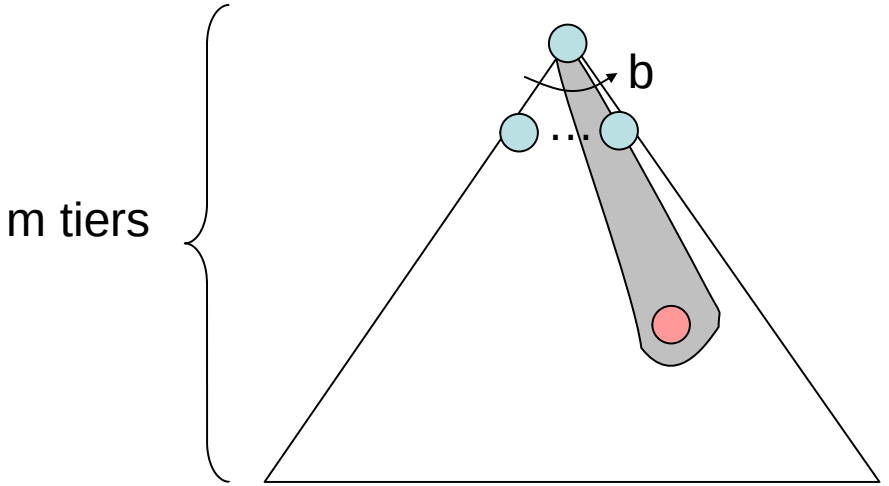
贪婪最佳优先搜索

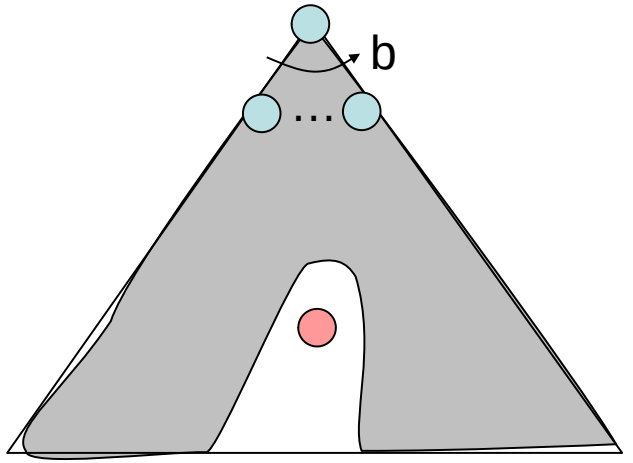
可能陷入死循环



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

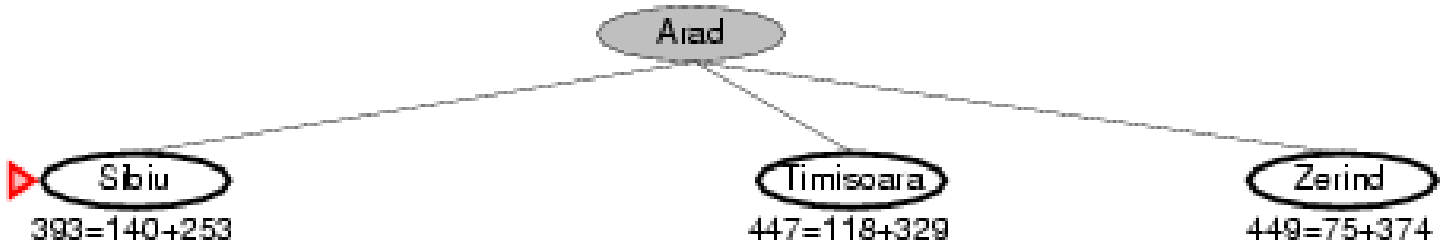
- 思路：避免扩展耗散值已经很大的路径
- 评估函数 f(n) = g(n) + h(n) ，经过结点 n 的最小代价解的估计代价
  - 代价函数 g(n) = 从初始结点 S 到达结点 n 已经花费的代价（实际代价）
  - 启发式函数 h(n) = 从结点 n 到目标结点 G 的最小代价路径的估计值
- 搜索策略：优先级队列，先扩展 f(n) 的值最小的结点

A* 搜索 - 案例

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

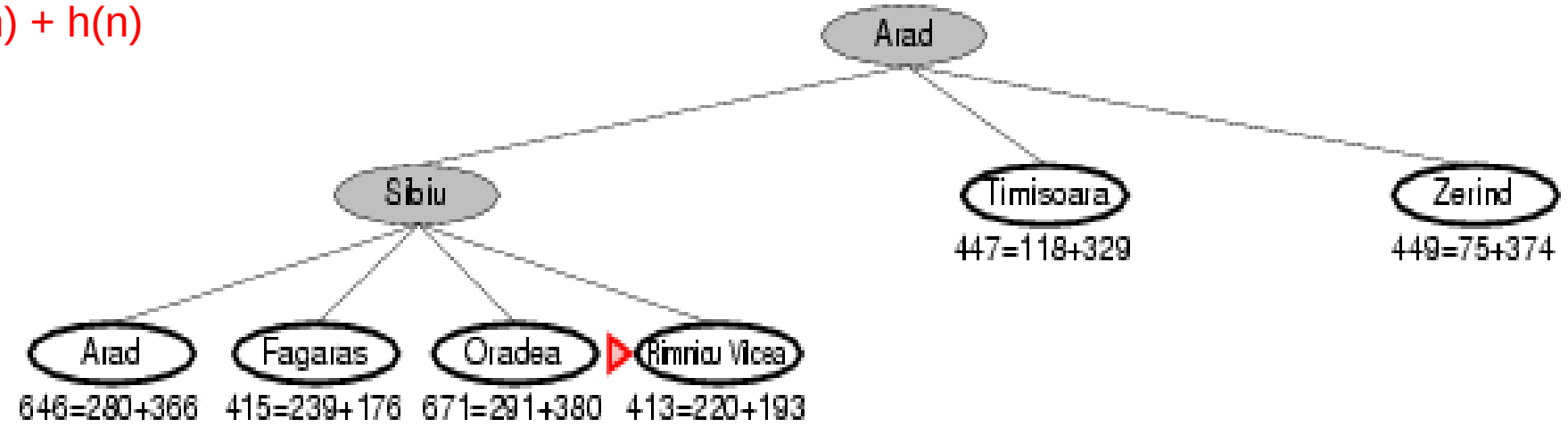
A* 搜索 - 案例

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



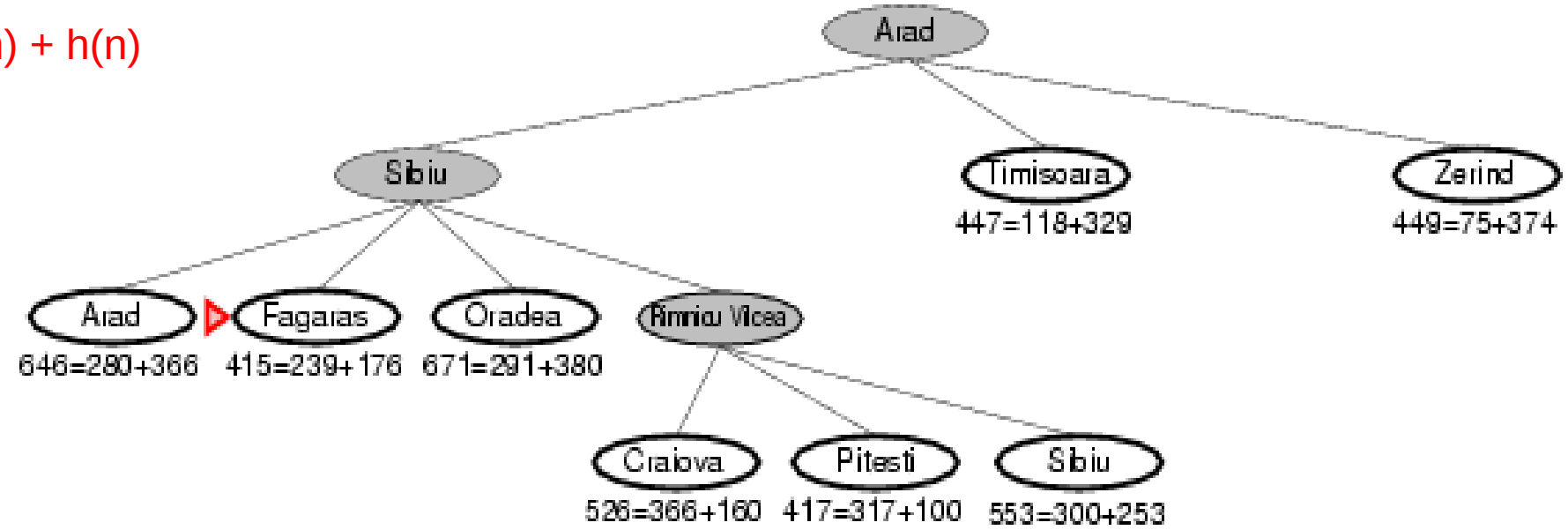
A* 搜索 - 案例

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



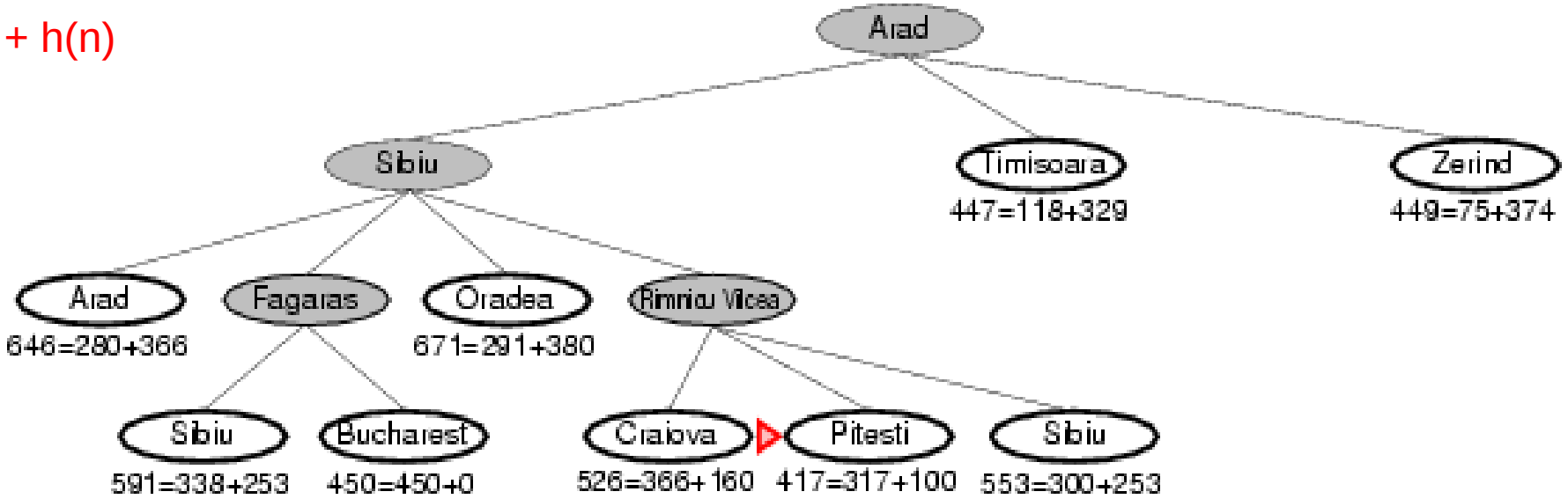
A* 搜索 - 案例

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



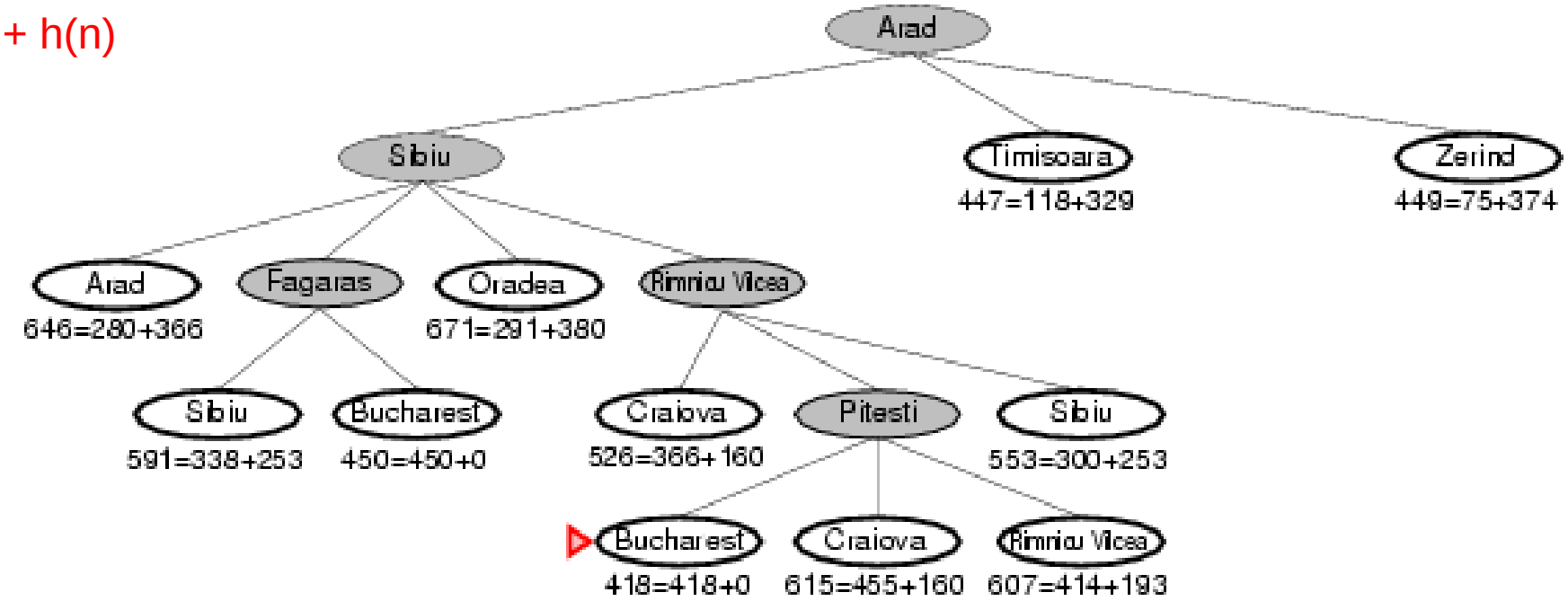
A* 搜索 - 案例

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



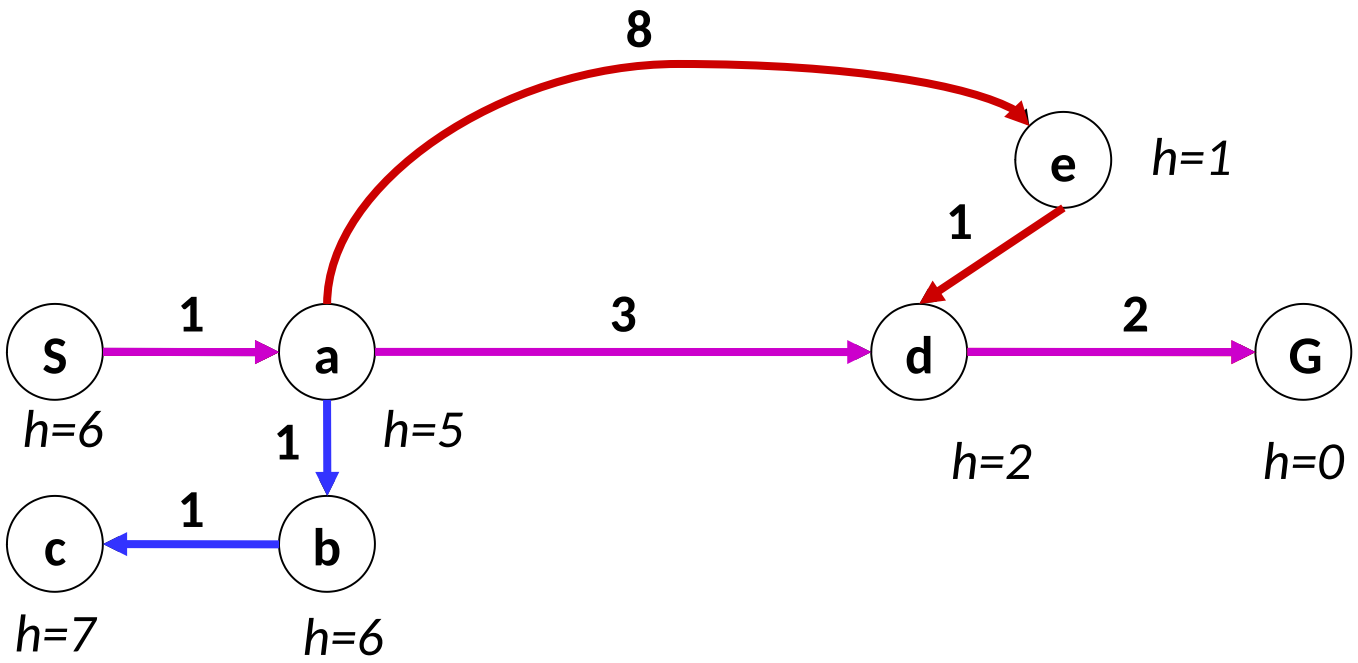
A* 搜索 - 案例

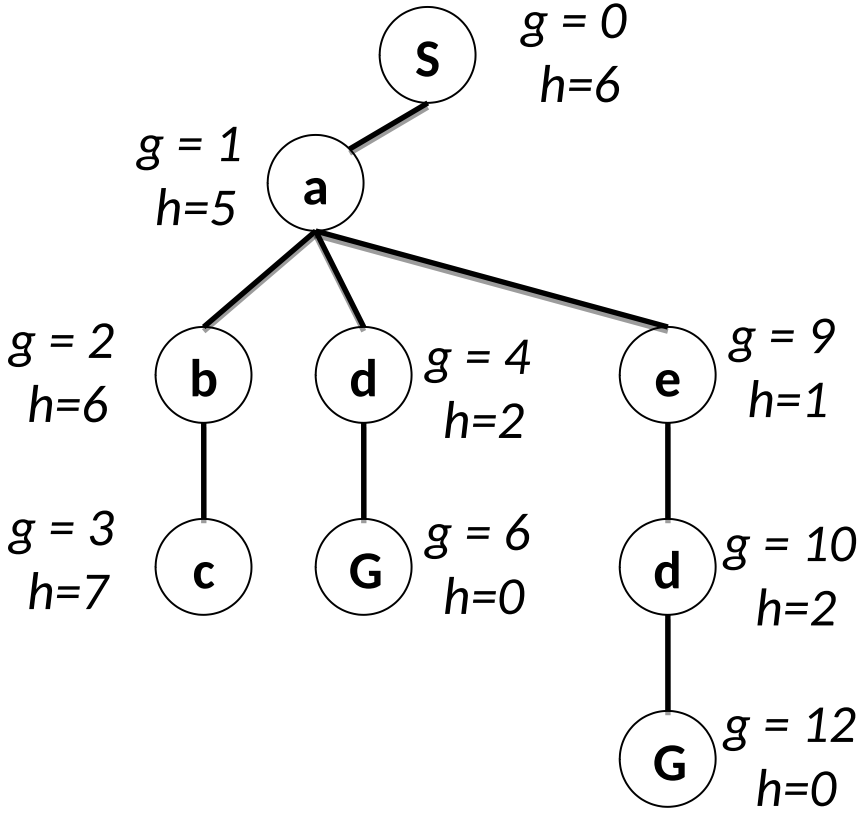
f(n) = g(n) + h(n)



课堂练习：UCS ，Greedy ，A* 搜索

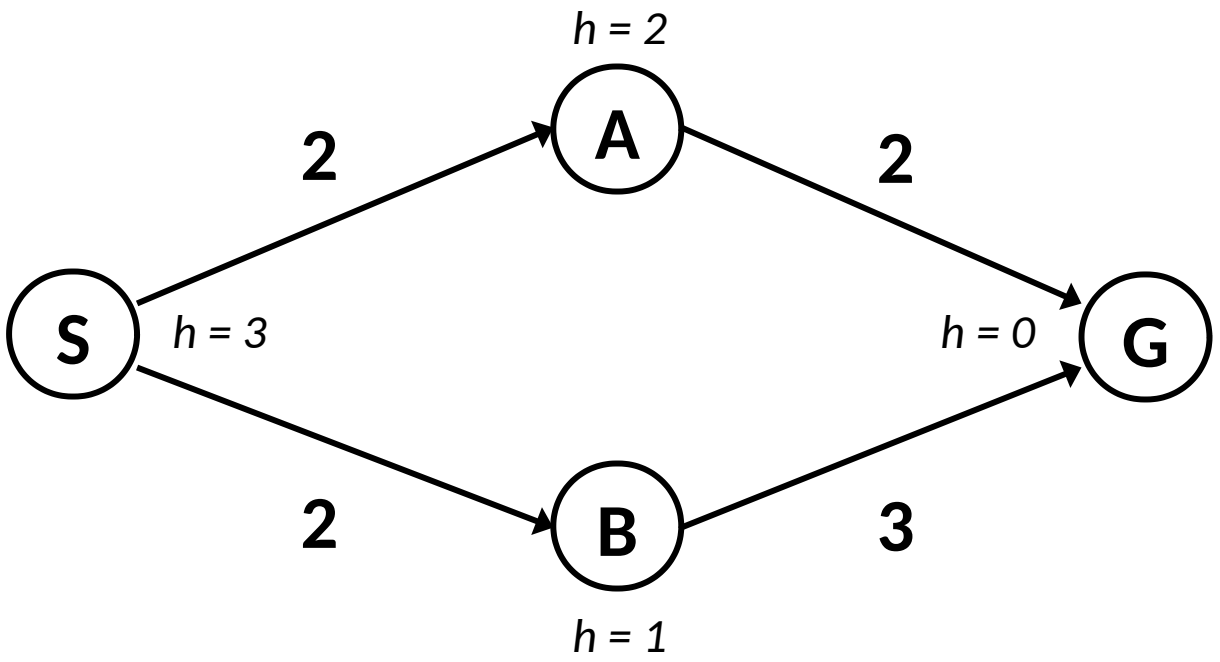
- UCS ： *backward cost* g(n)
- Greedy ： *forward cost* h(n)





- A* 搜索： f(n) = g(n) + h(n)

When should A* terminate?



- 当目标移除队列时算法终止

Is A* Optimal?

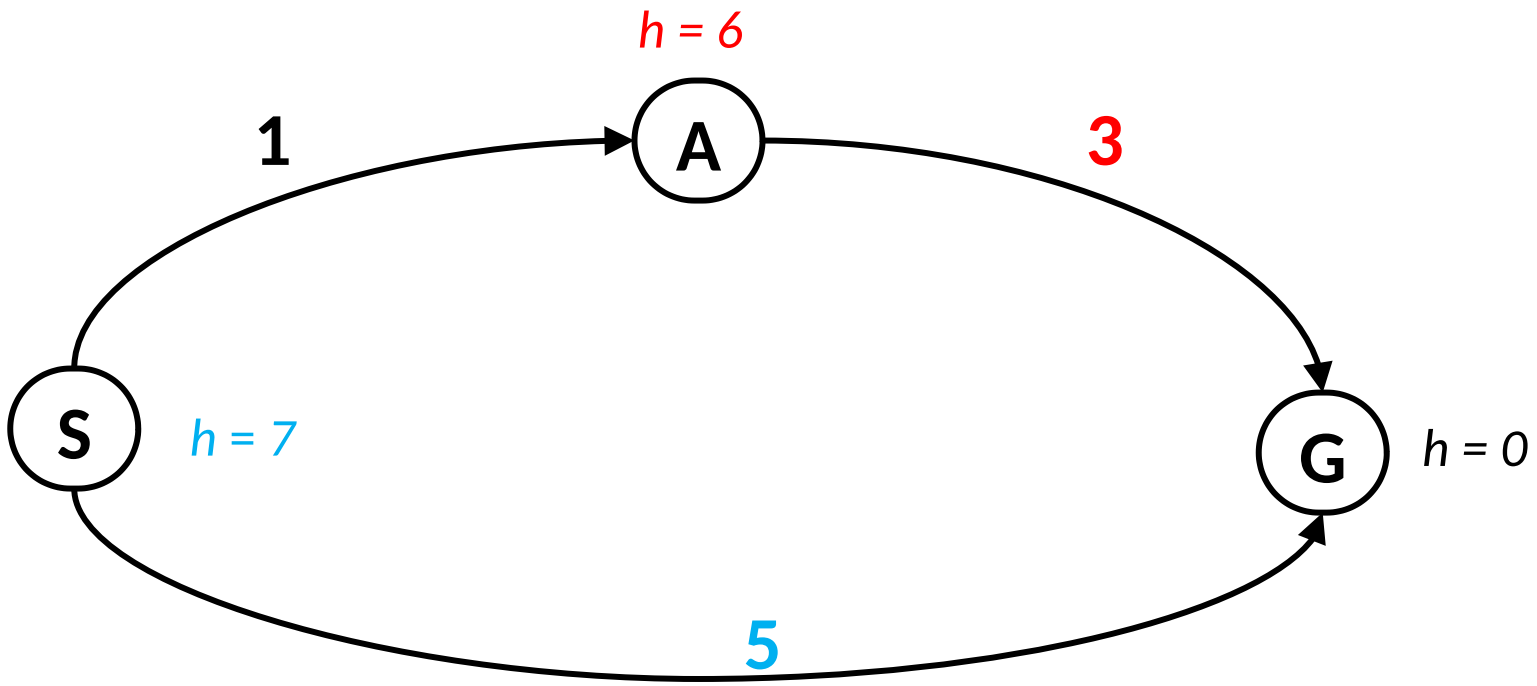
Fringe

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)