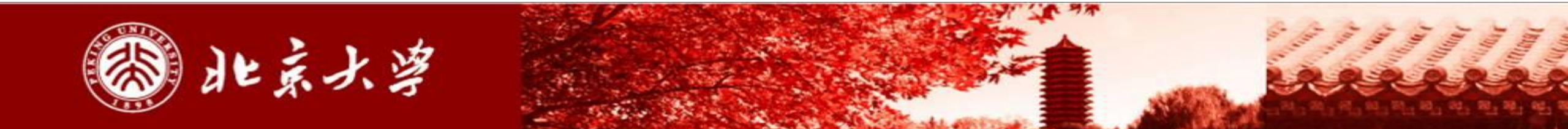
# 《物理与人工智能》

## 15. 搜索：UCS和A*

授课教师：马滟青

2025/10/27（第七周）

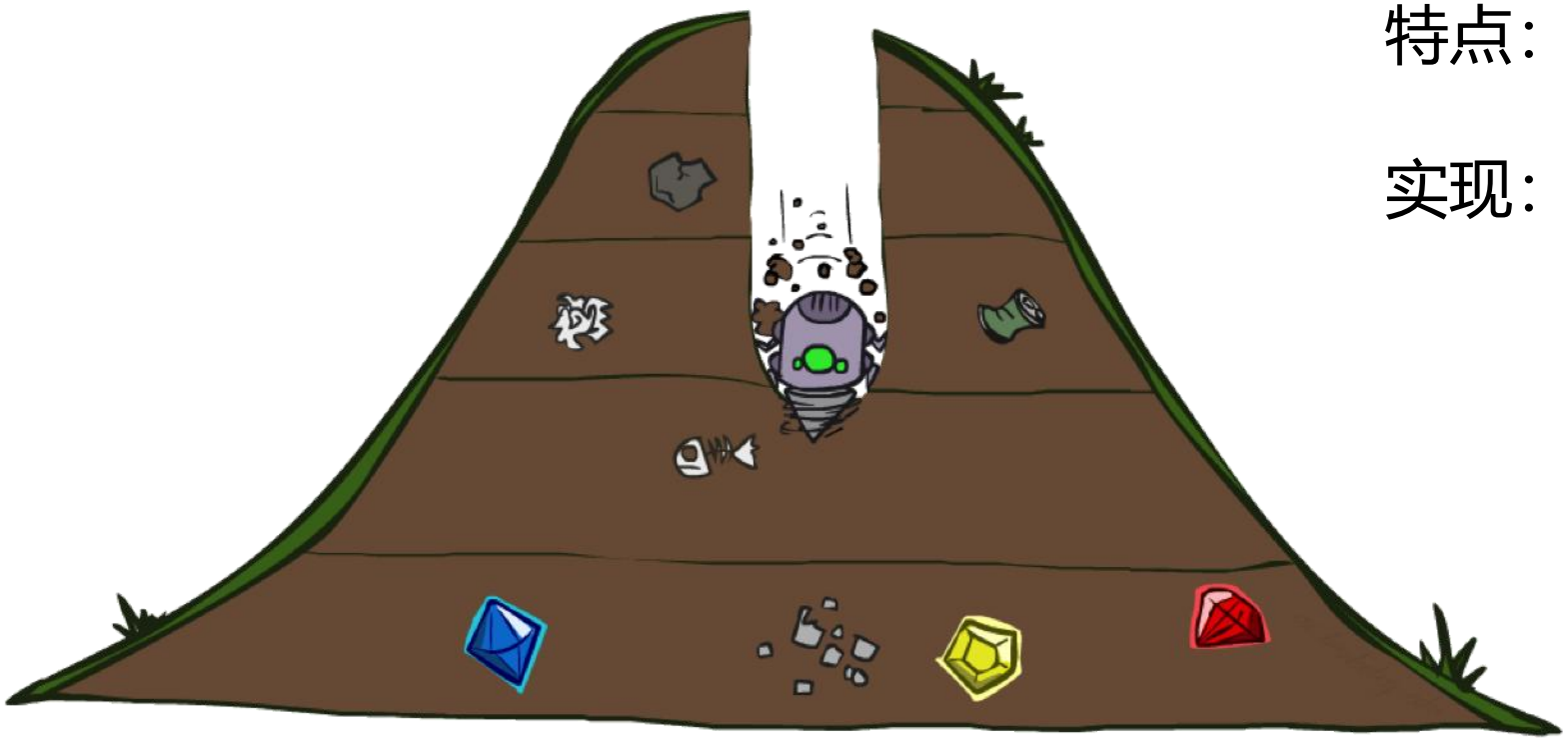
鸣谢：基于计算机学院《人工智能引论》课程组幻灯片



# 目录

- 目标：改进搜索
- 理解代价/成本函数会如何影响搜索：UCS
- 启发（heuristic)
- 启发式搜索：A*

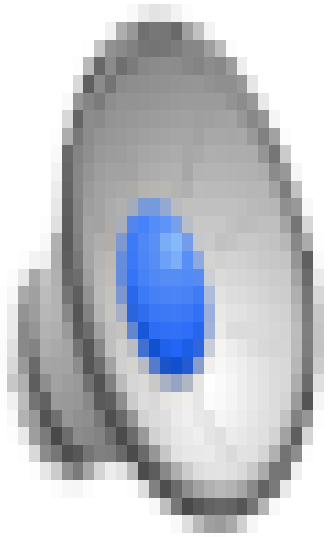
# 复习：深度优先搜索（depth first search）



特点：优先扩张最深的节点

实现：后进先出的栈（stack）

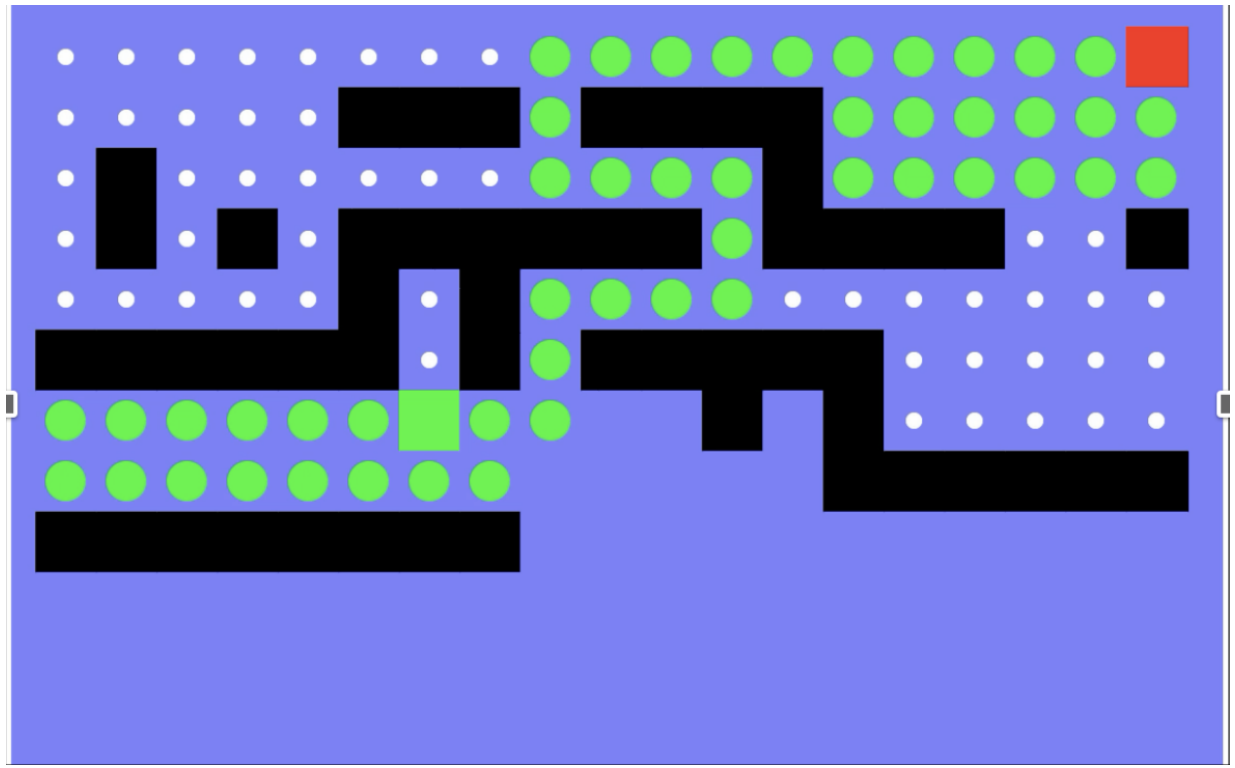
# 复习：深度优先搜索（depth first search）



特点：优先扩张最深的节点

实现：后进先出的栈（stack）

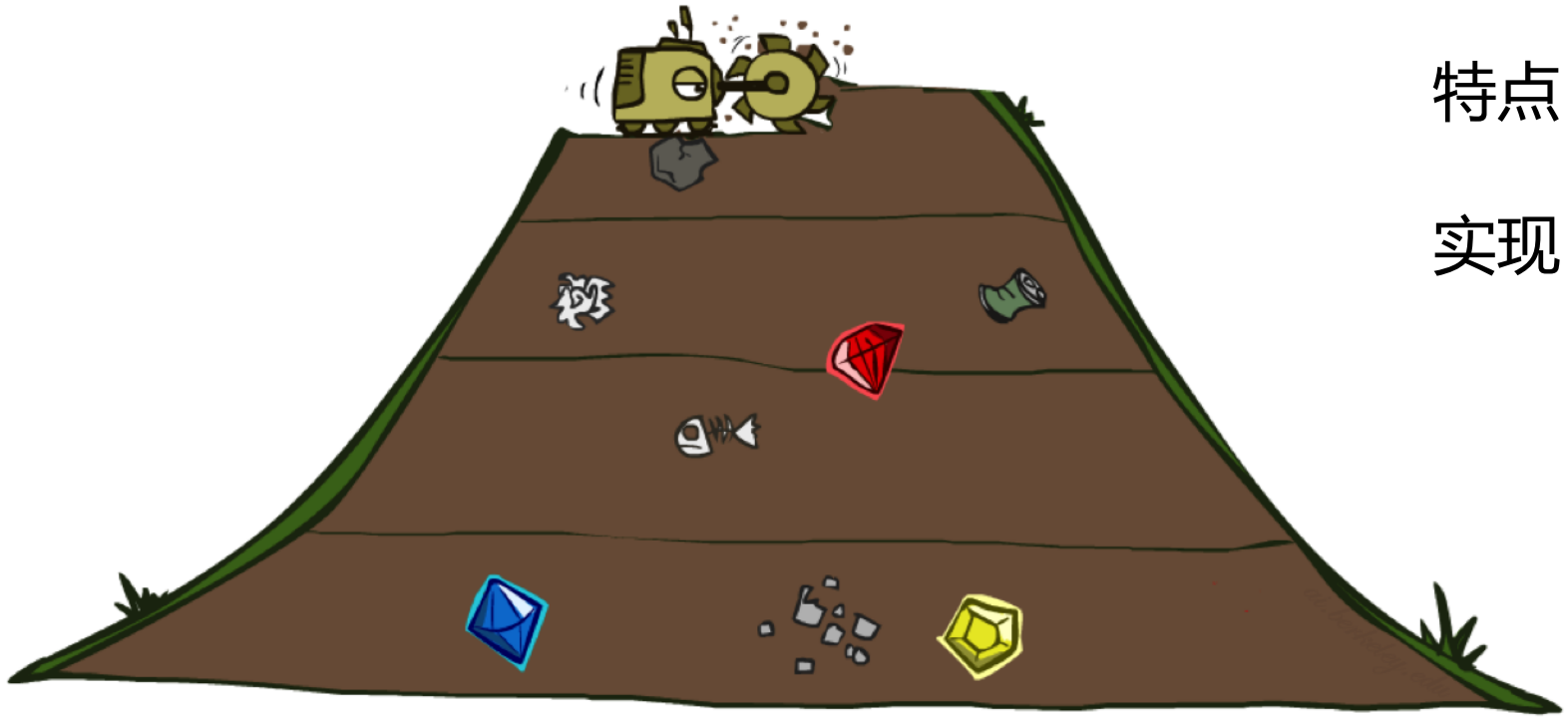
# 复习：深度优先搜索（depth first search）



特点：优先扩张最深的节点

实现：后进先出的栈（stack）

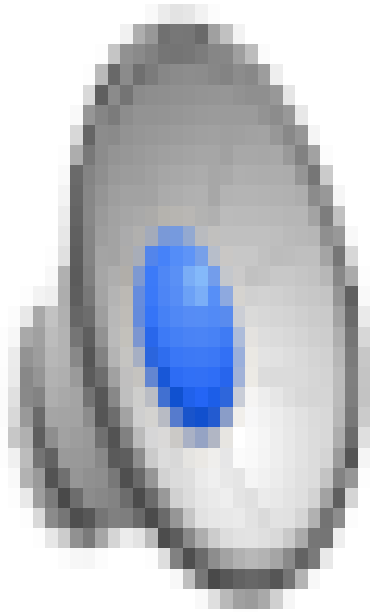
# 复习：广度优先搜索（breadth first search）



特点：优先扩张最浅的节点

实现：先进先出的队列（queue）

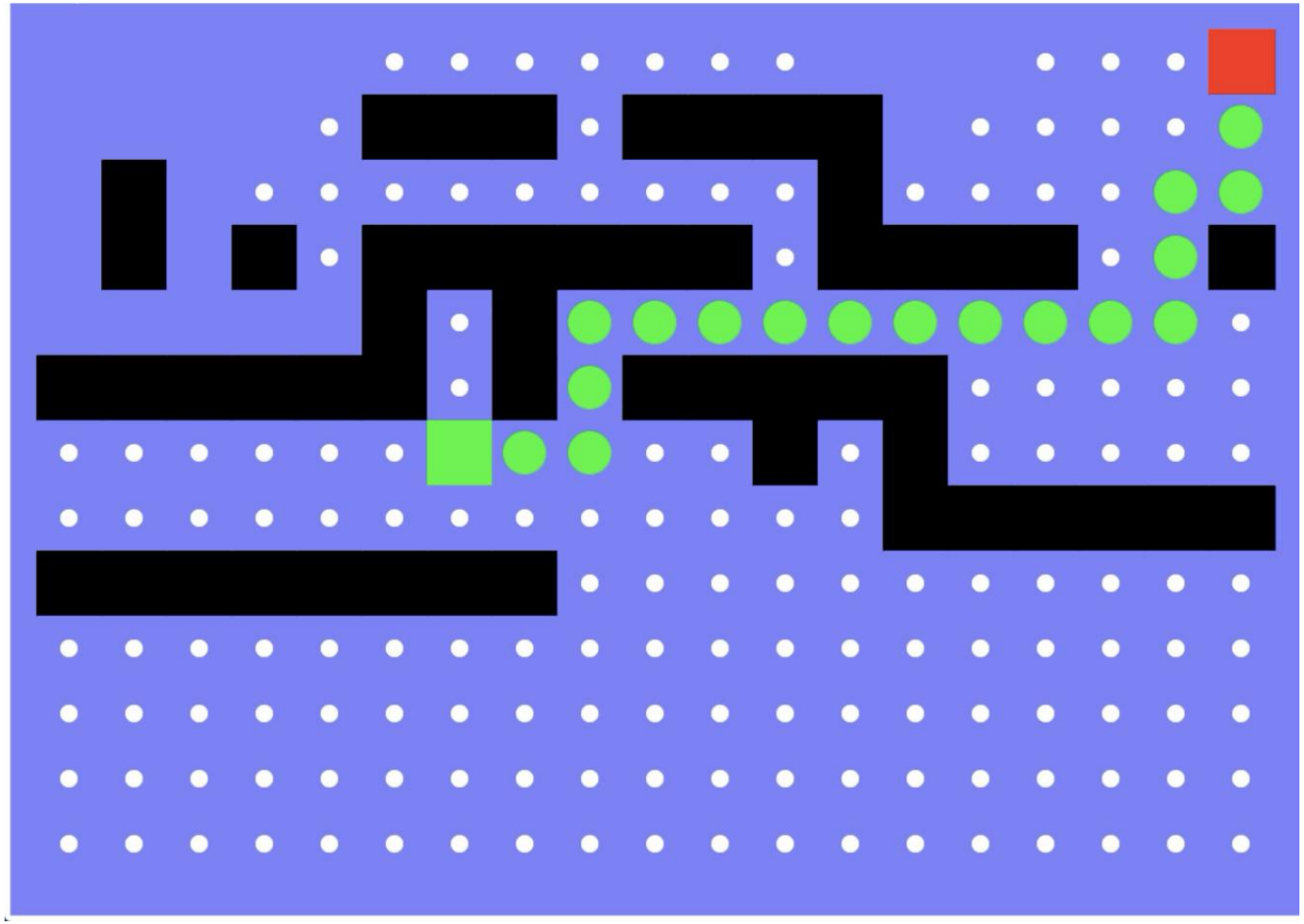
# 复习：广度优先搜索（breadth first search）



特点：优先扩张最浅的节点

实现：先进先出的队列（queue）

# 复习：广度优先搜索（breadth first search）



特点：优先扩张最浅的节点

实现：先进先出的队列（queue）

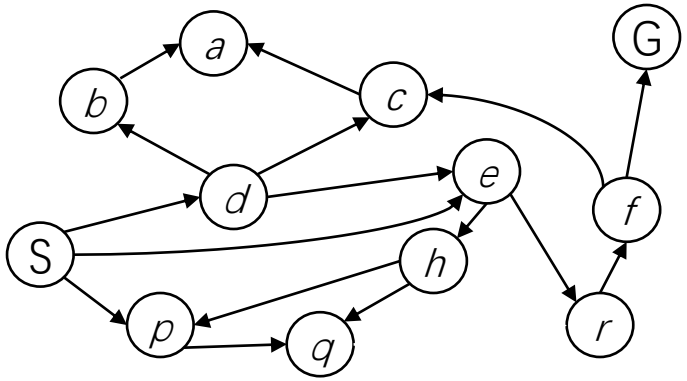
# 深搜，one more time

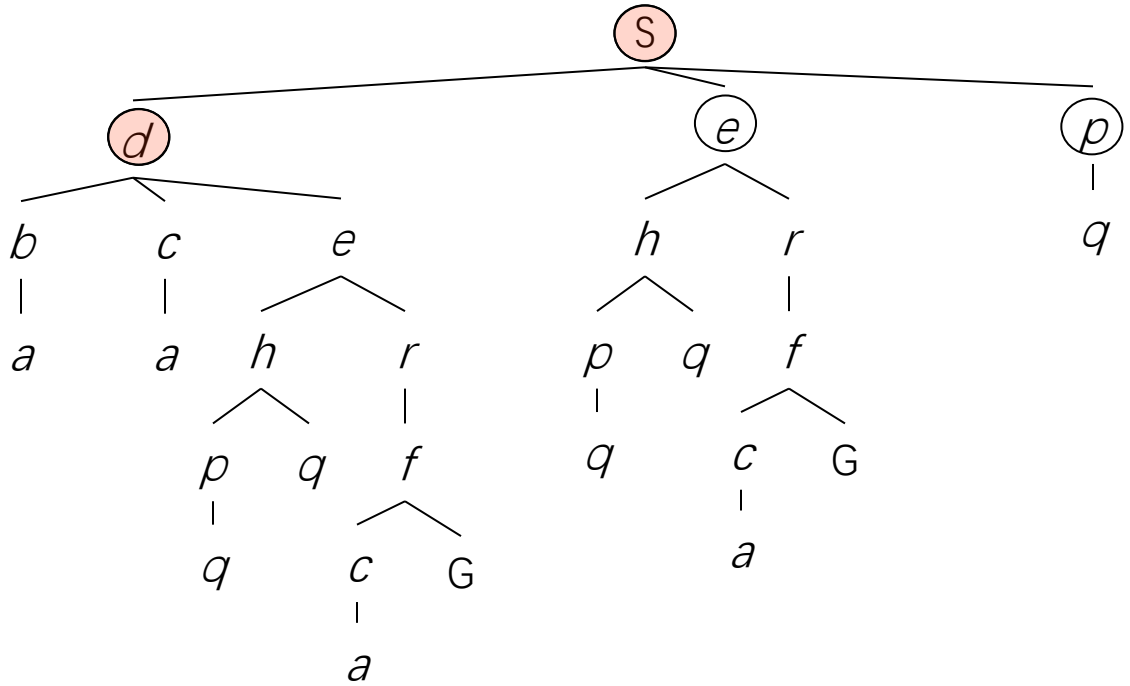
深搜找到的答案是最优的吗?

# 深搜，one more time

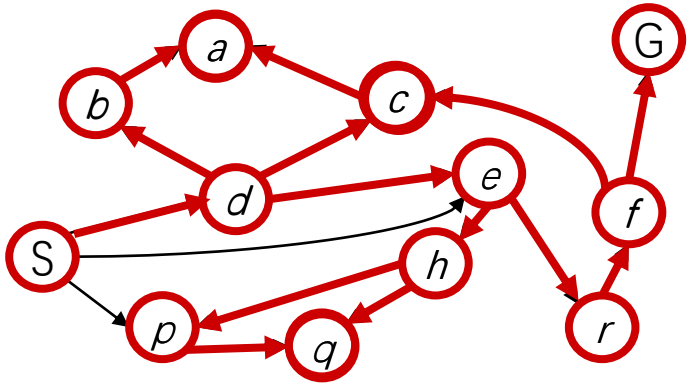
深搜找到的答案是最优的吗?
深搜找到的答案是 步数最少的？成本/代价最小的?

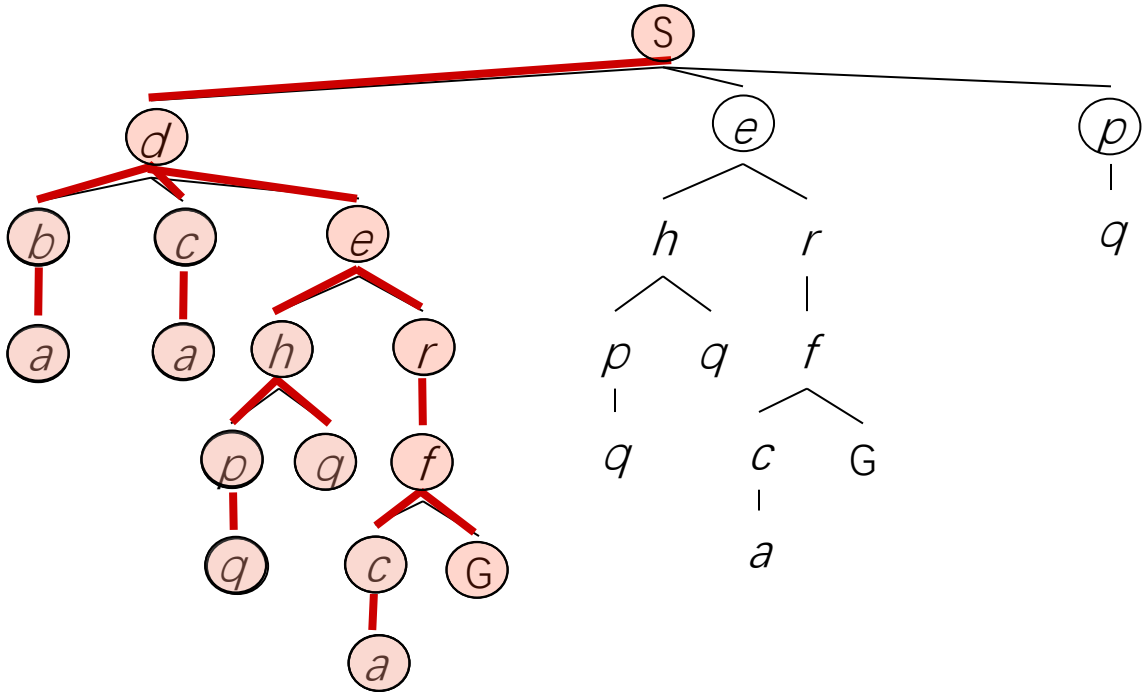
# 深搜，one more time



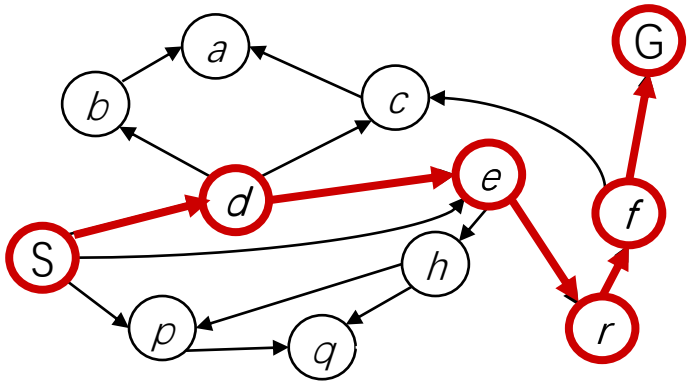



# 深搜，one more time



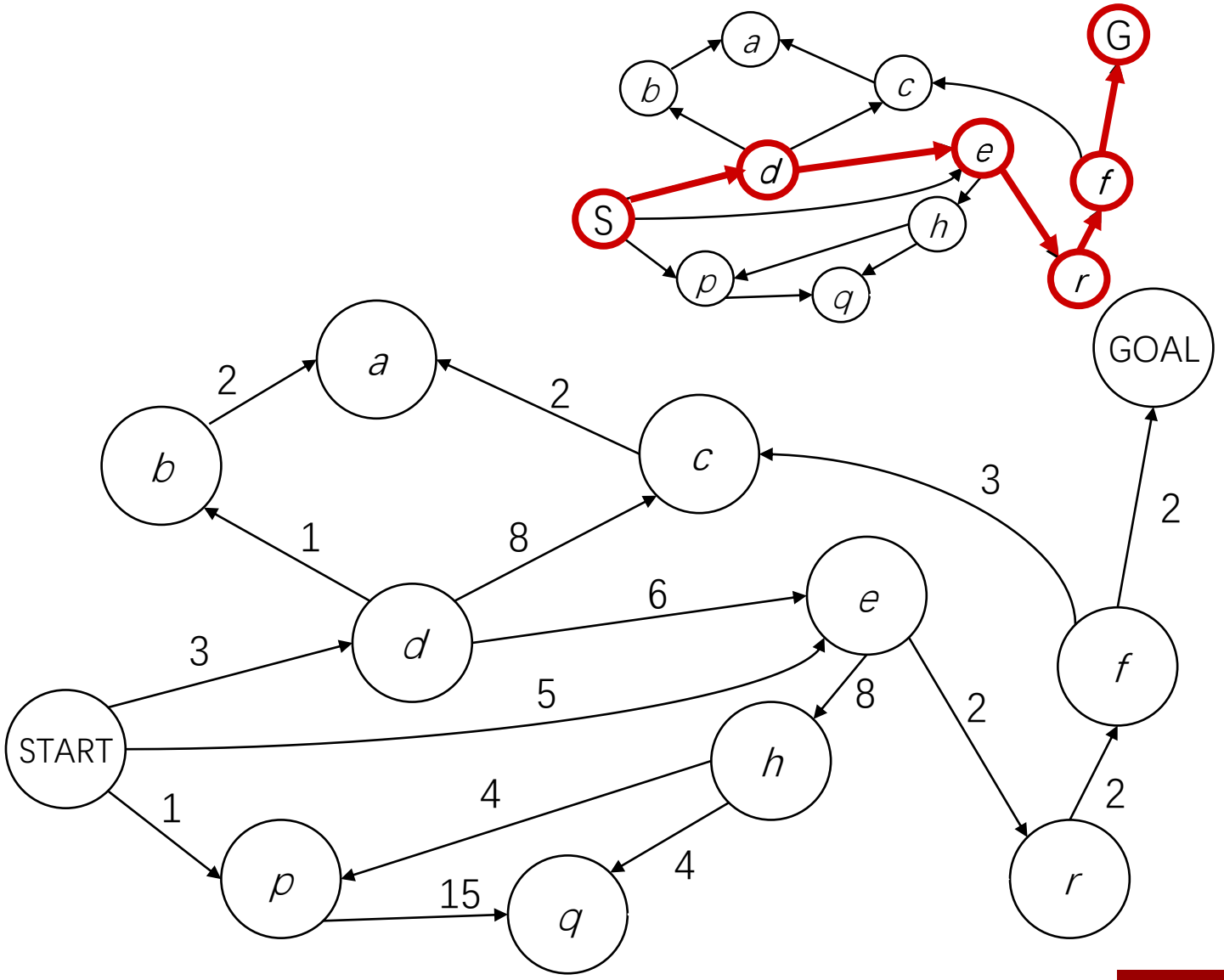



# 深搜，one more time




找到的答案是
S、d、e、r、f、G

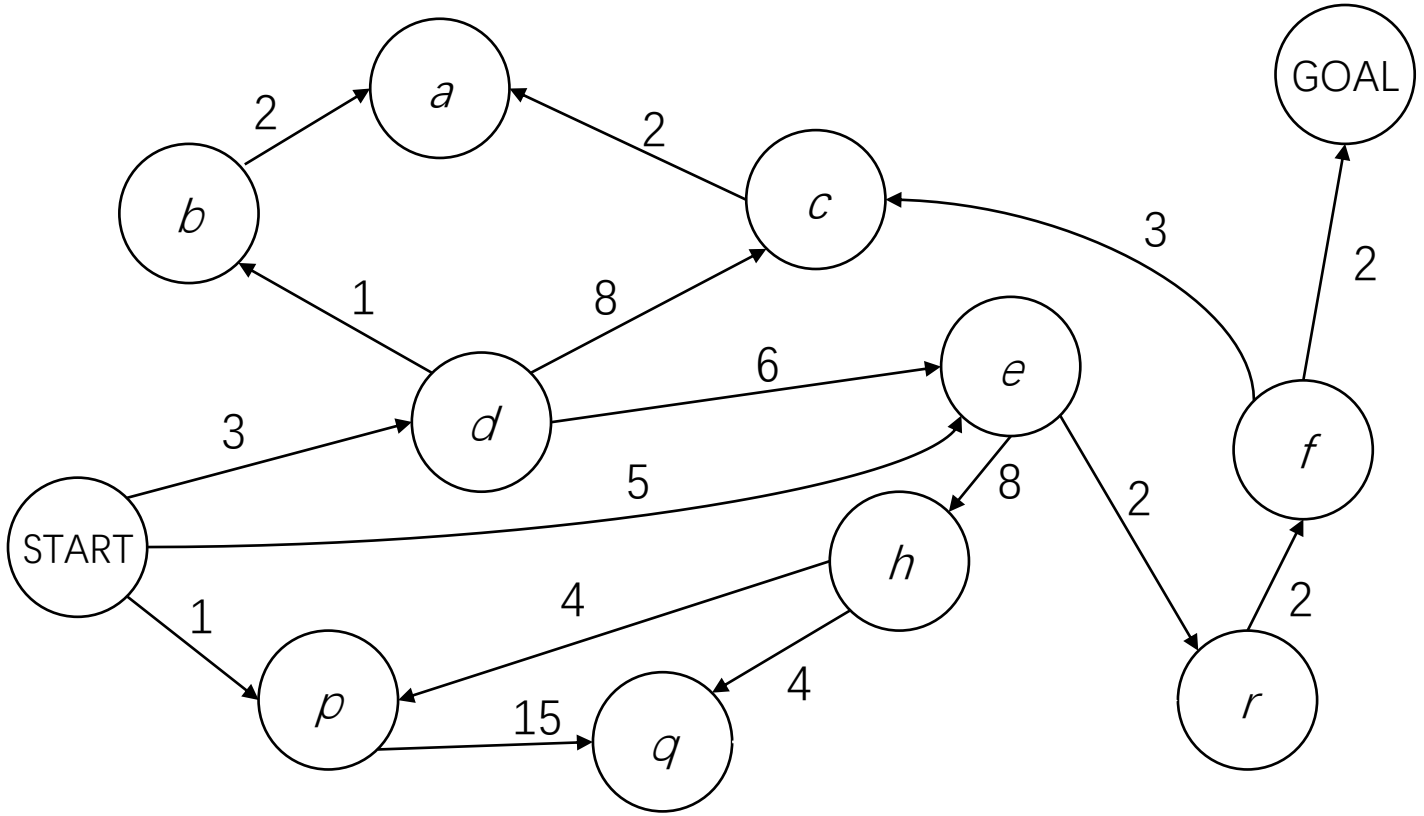
# 深搜，one more time




找到的答案是
S、d、e、r、f、G

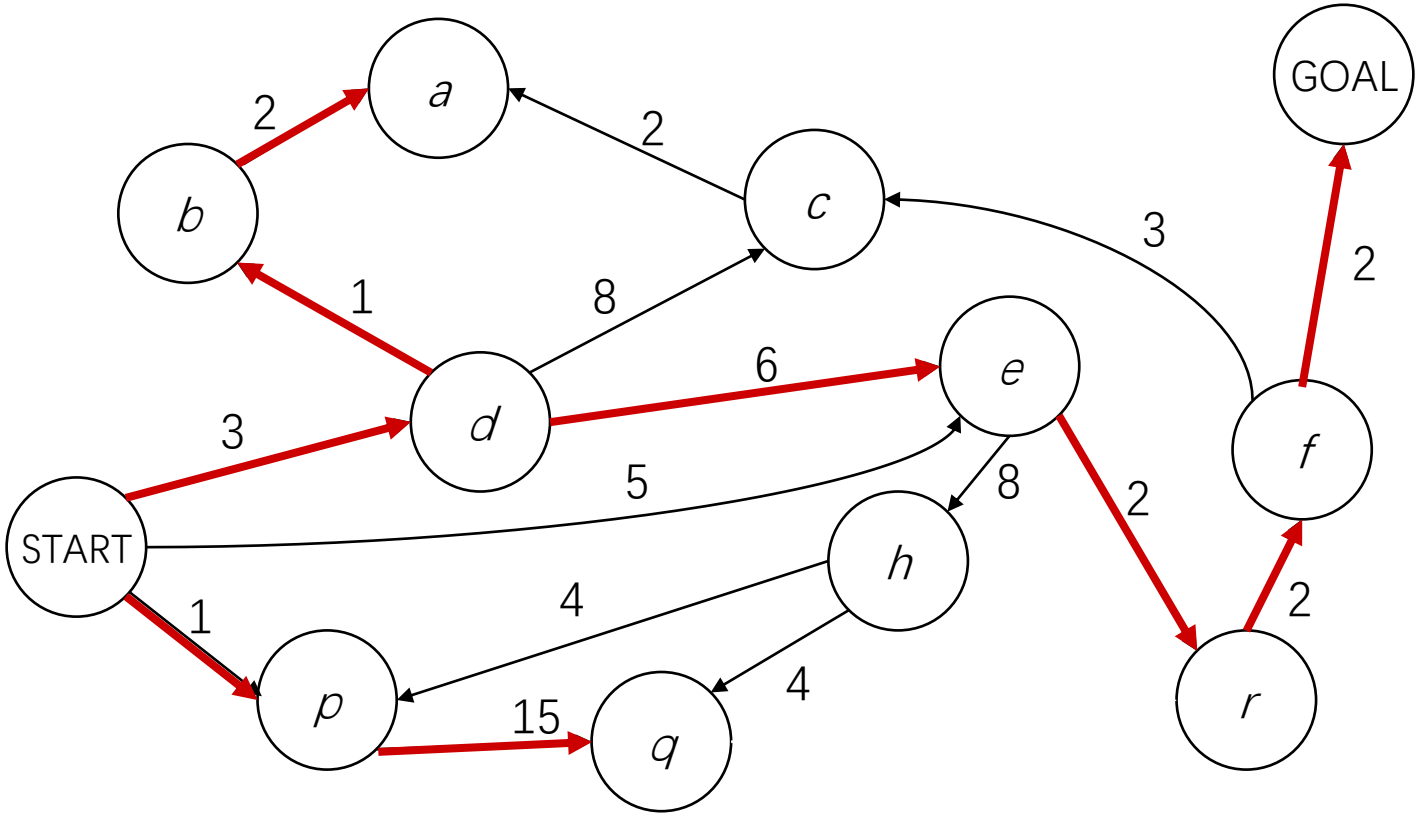
在有成本/代价的时候，
同样的答案是最优的吗？

# 广搜，one more time



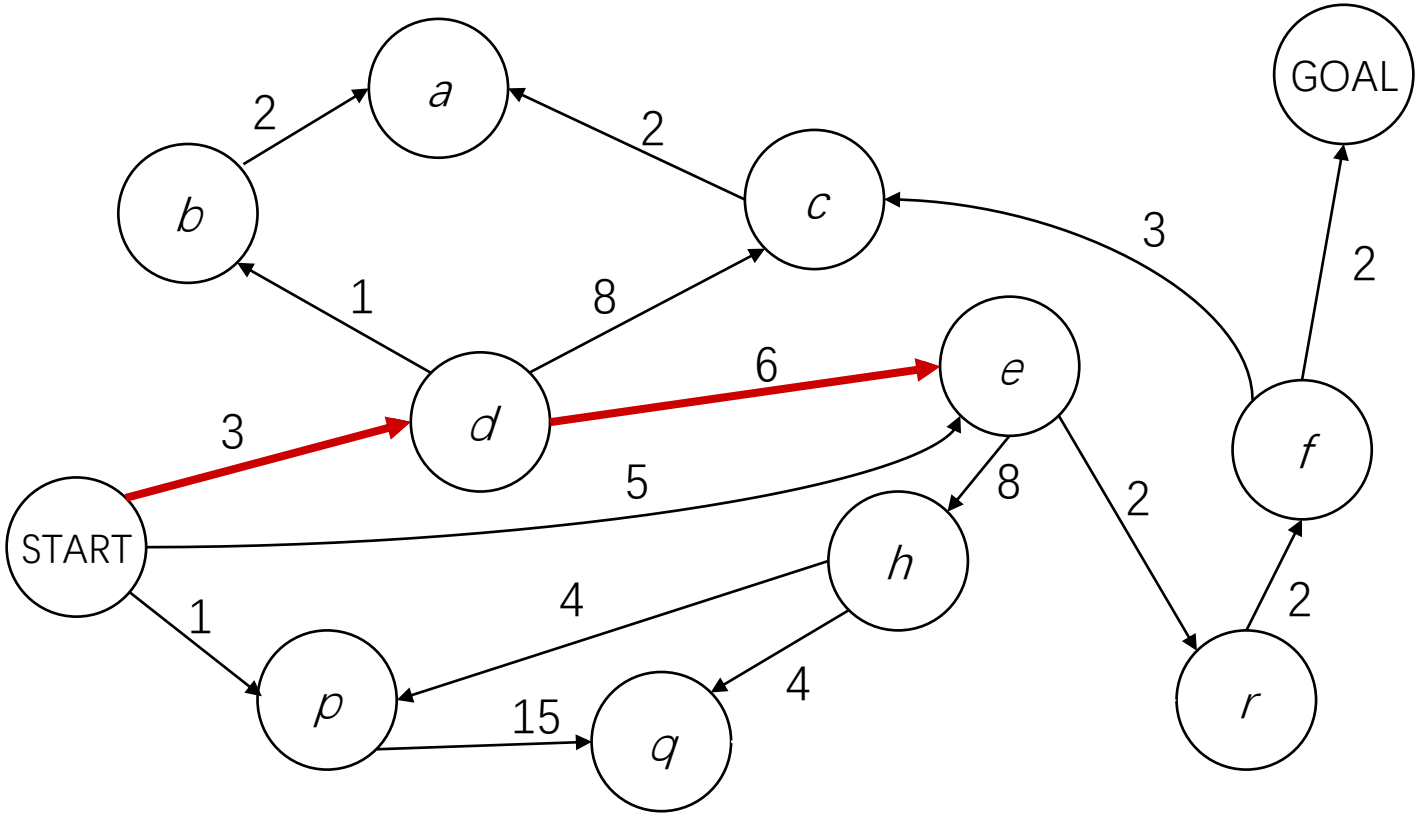

思考题：
广搜可以保证最优吗？

# 改进深搜



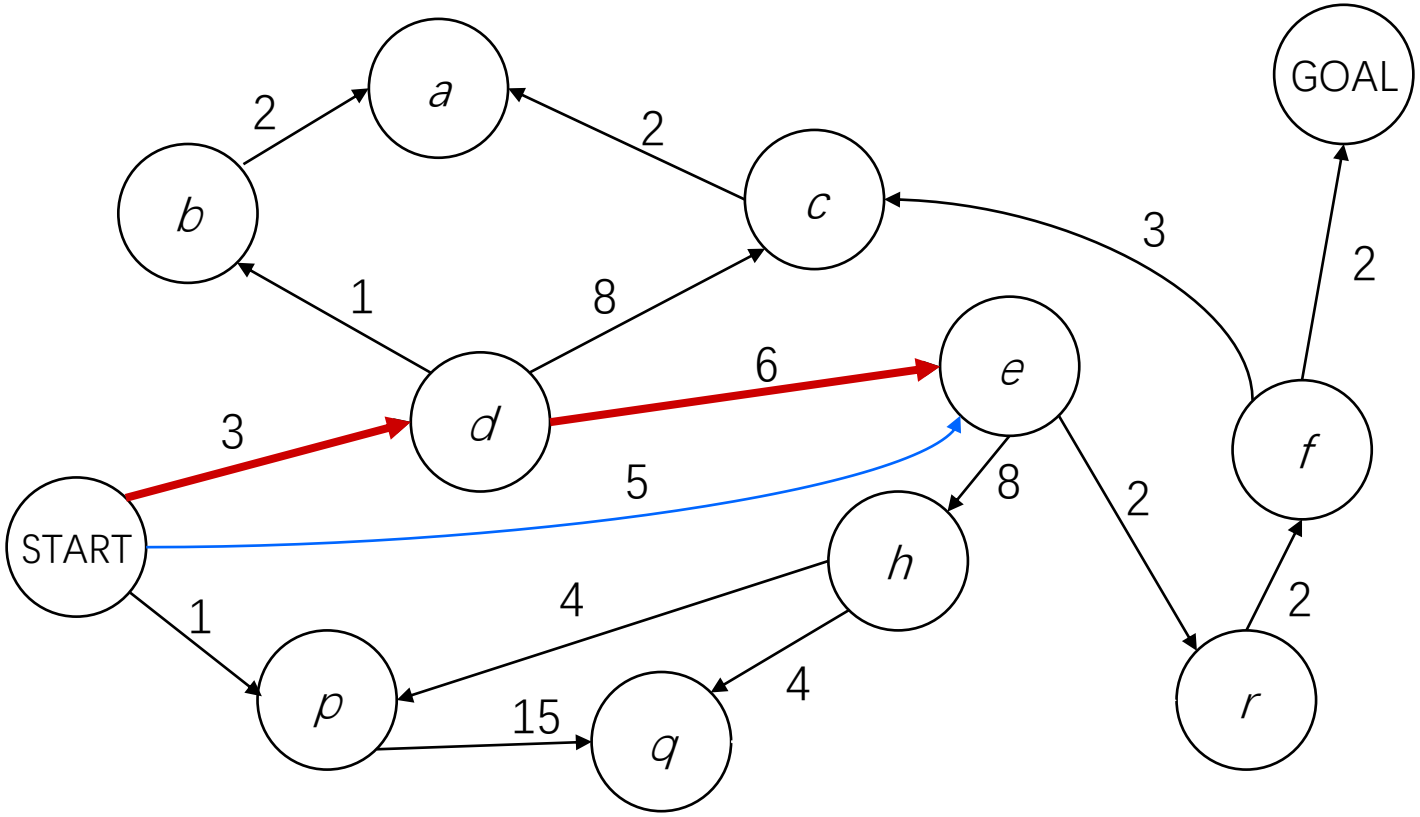

那我们在每步的时候考虑代价?

# 改进深搜



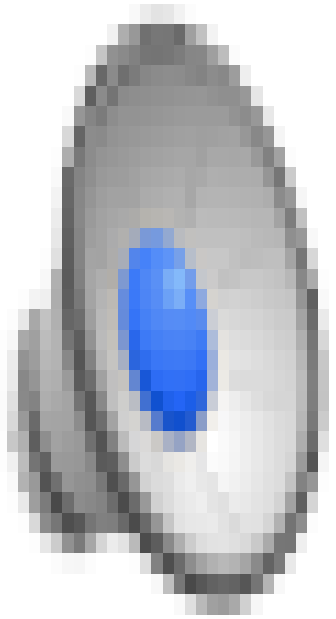

问题出在哪？为什么不直接选S->e?

# 统一代价搜索（uniform cost search）



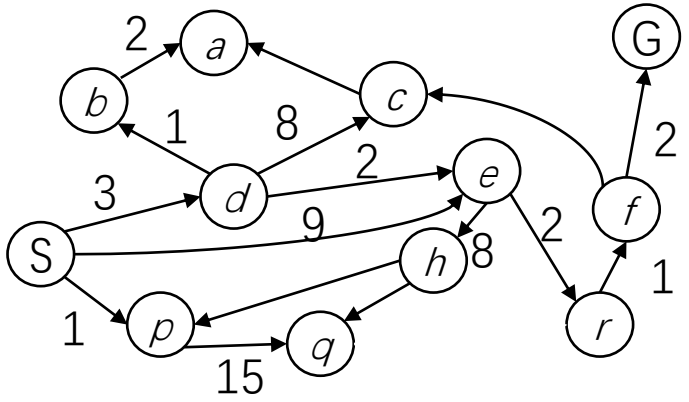

我们不应该只关注单边的成本，
而是要考虑总成本（从开始节点累积到当前节点）

# 这是什么搜索?



水深一样的情况
(还是要考虑直线距离是小于斜边距离的)

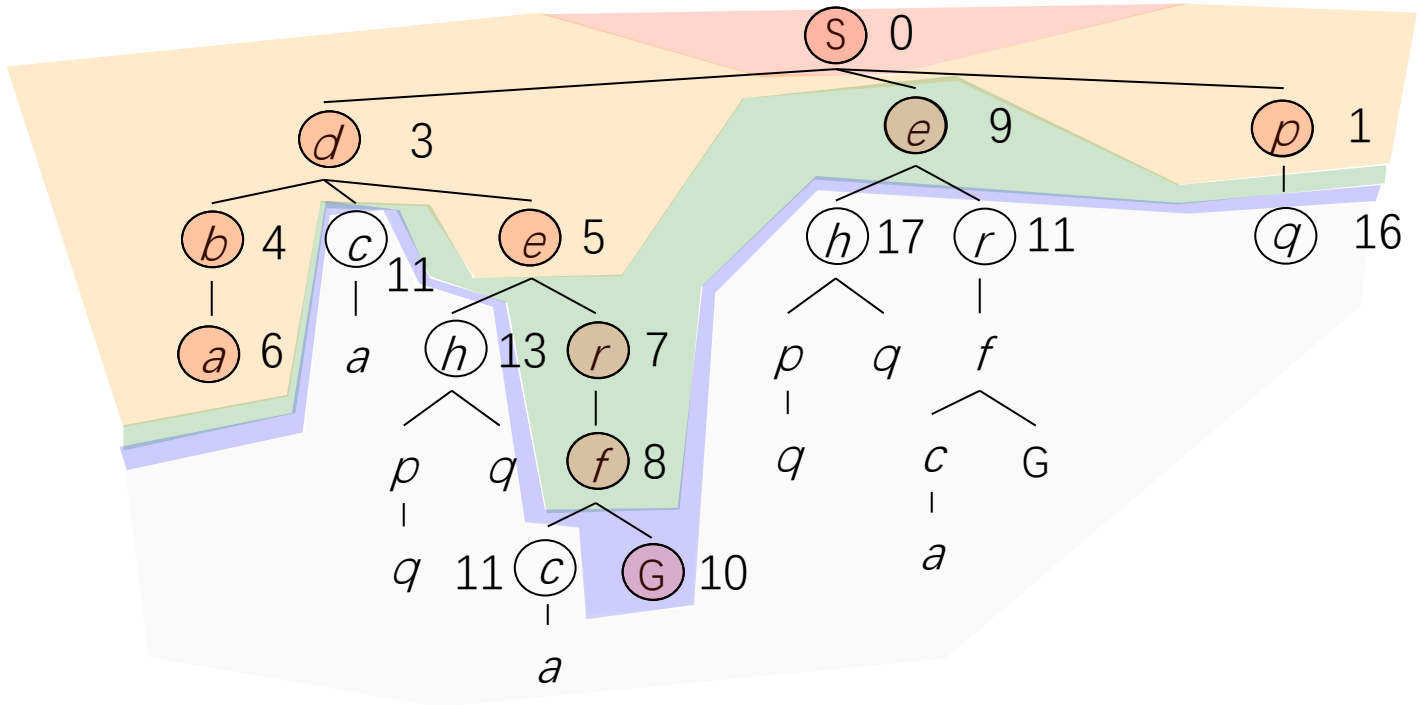
# 搜索树 -> 代价轮廓



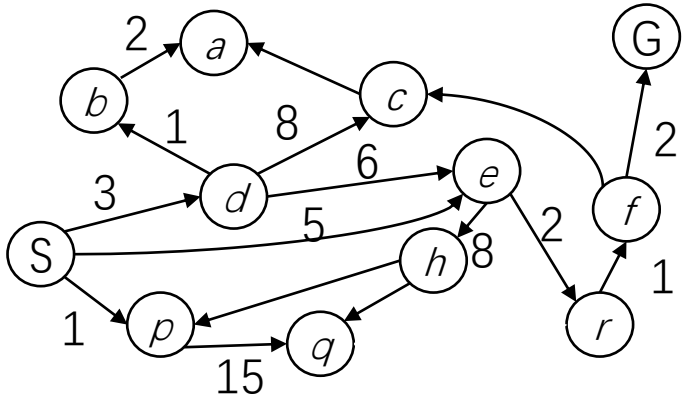

特点：优先扩张代价最小的节点

实现：总成本优先队列（priority queue）

代价轮廓



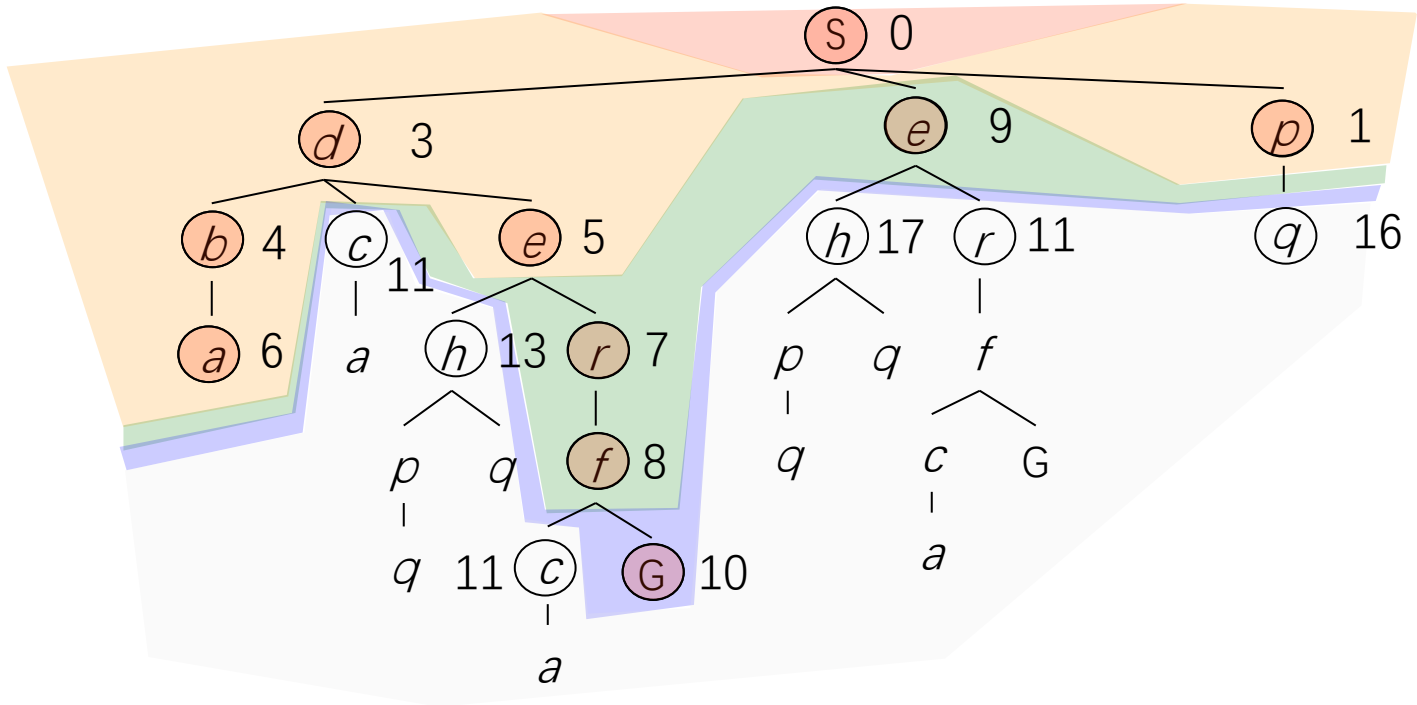
# 搜索树 -> 代价轮廓




特点：优先扩张代价最小的节点

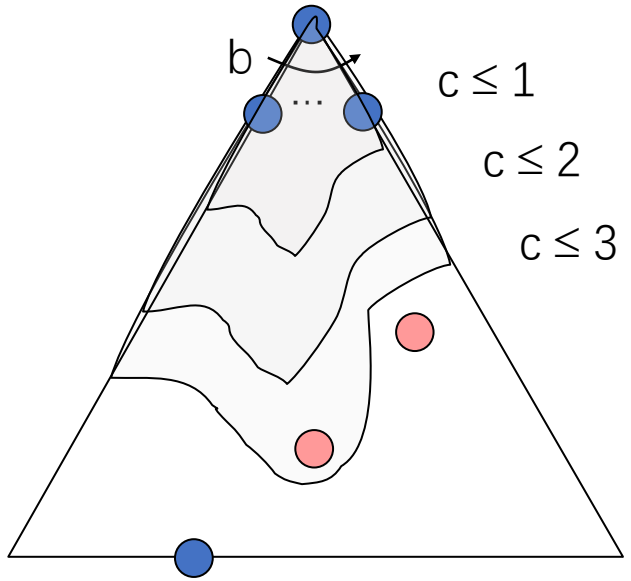
实现：总成本优先队列（priority queue）

代价轮廓



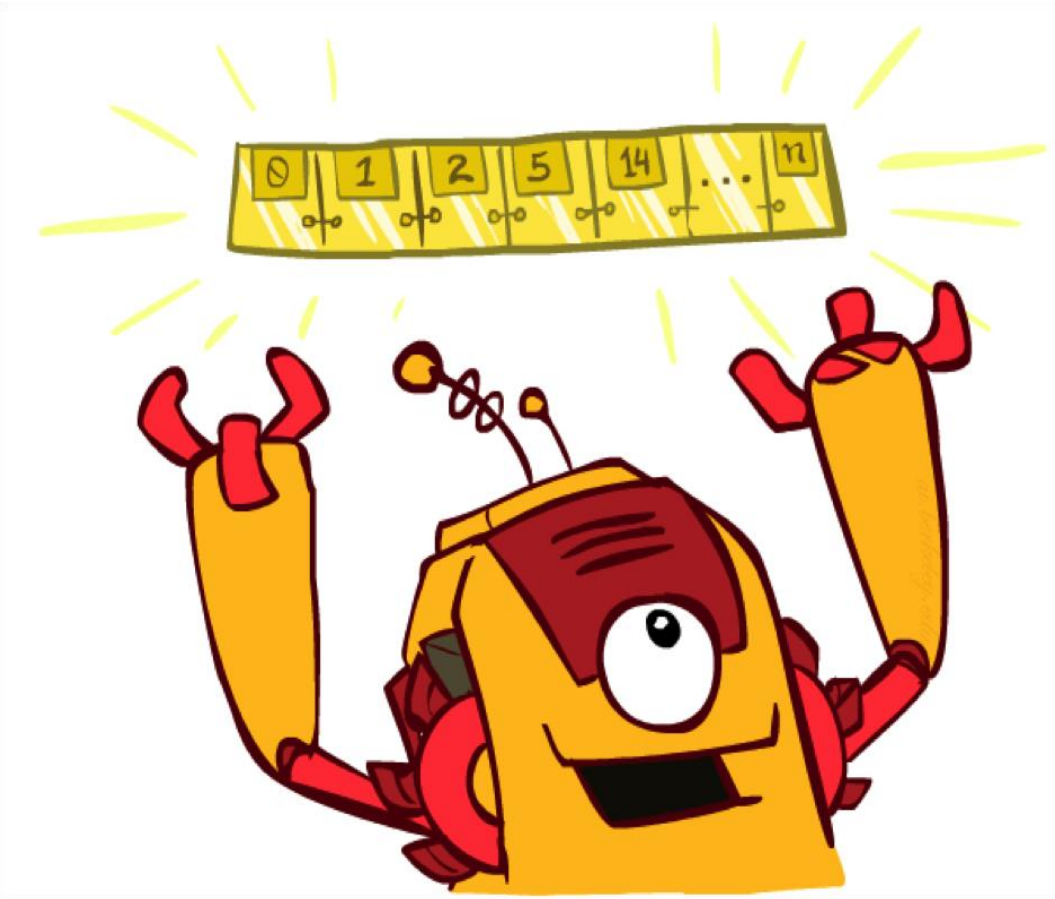
# UCS的性质

- UCS扩展什么节点?
  - 处理所有比现有解法代价要低的节点
  - 沿着代价轮廓，扩展越来越贵的节点
- 算法完整吗?
  - 假设最佳答案是有限代价，并且边的最低代价是正的，那是!
- 一定最优吗?
  - 是！ (通过A*我们能知道怎么证)
- 局限?
  - 没有考虑关于目标的信息



# 重新理解搜索：优先队列（priority queue）

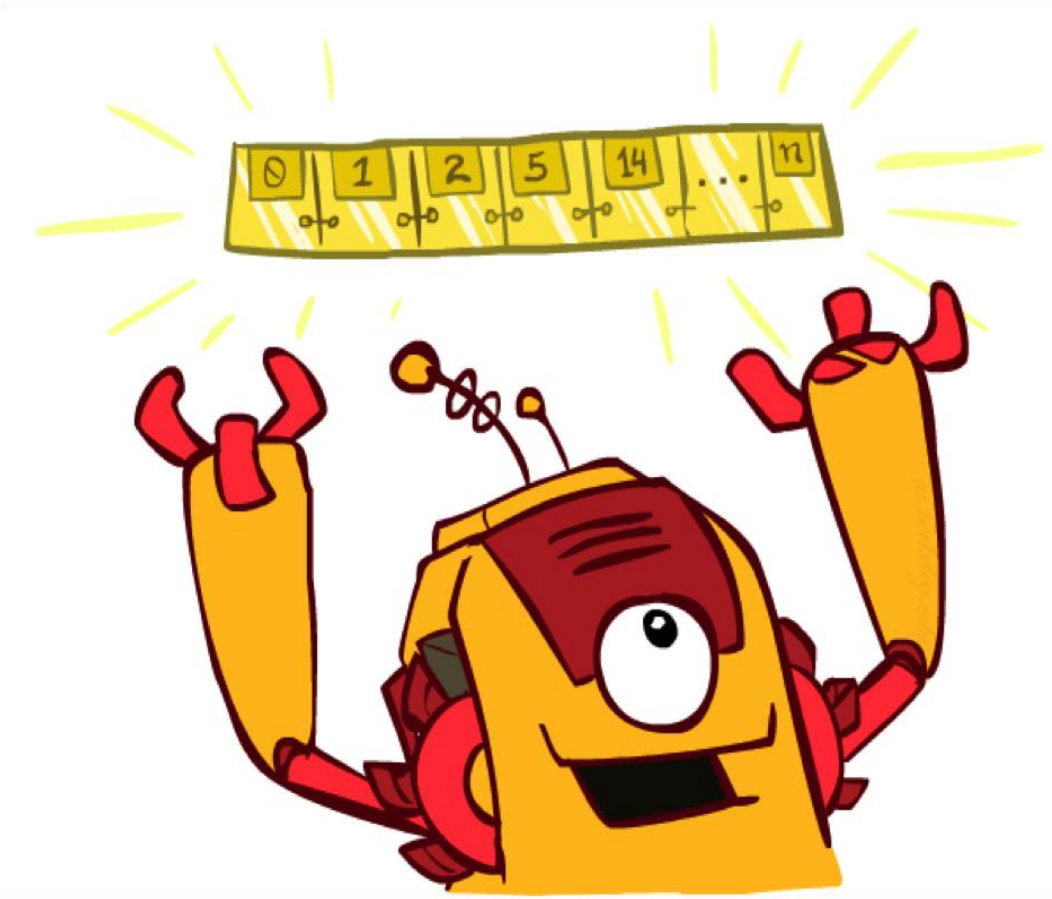
- 到现在，所有搜索的实现其实都是一样，除了“如何保存接下来可能要扩张的节点”方式不同
  - 所有的保存“列表”都是优先队列 (i.e. 保存所有节点和他们的优先值)



# 重新理解搜索：优先队列（priority queue）

北京大学 PEKING UNIVERSITY

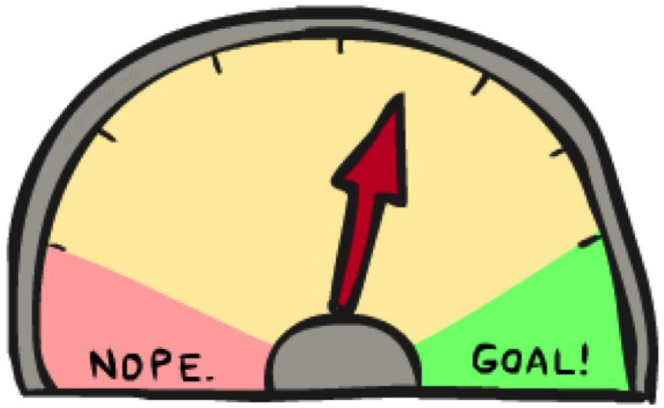
- 到现在，所有搜索的实现其实都是一样，除了“如何保存接下来可能要扩张的节点”方式不同
  - 所有的保存“列表”都是优先队列 (i.e. 保存所有节点和他们的优先值)
  - 但优先队列的维持，即每加入一个新的节点和其优先值，都需要O(log N)
  - 所以在深搜和广搜的实现上，我们使用栈（stack）和队列（queue），使我们的算法跑得更高效
  - 其实，我们也可以用同一个框架，就同时实现三个搜索算法



# 重新理解搜索

- 盲目搜索
  - 深搜（DFS）
  - 广搜（BFS）
  - 统一代价搜索（UCS）

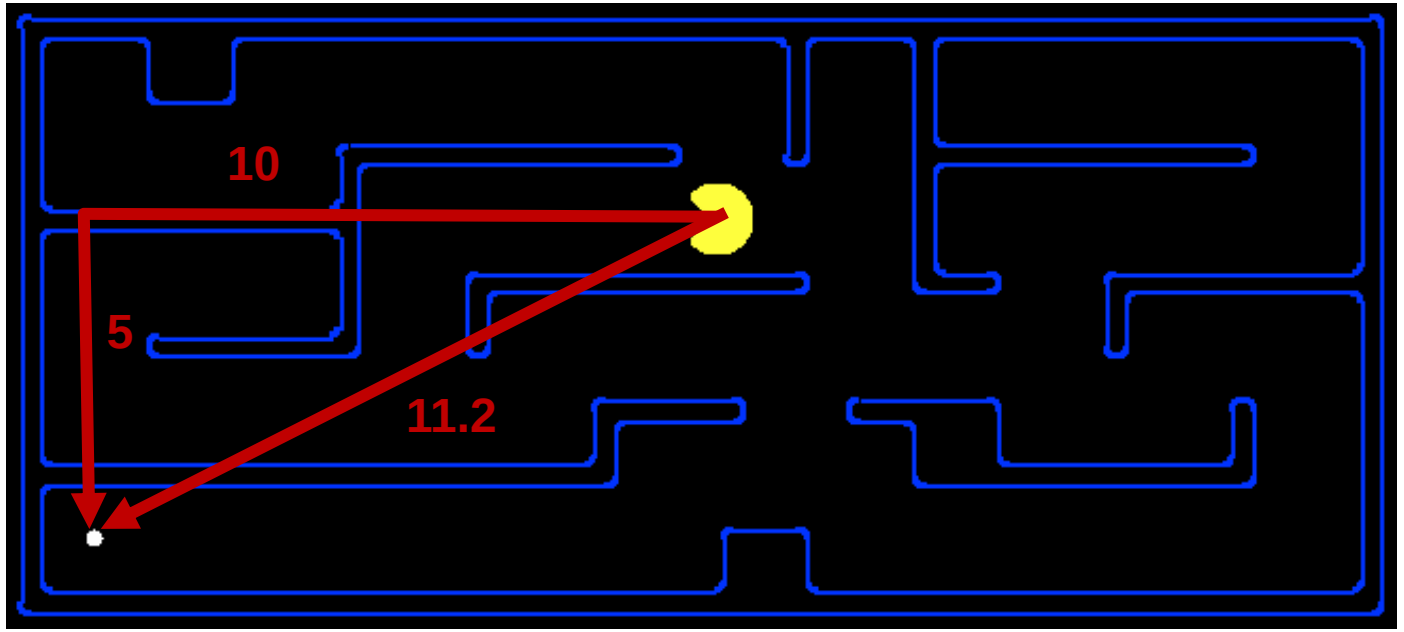
- 知情搜索
  - 启发 （heuristics）
  - 贪心搜索 （greedy）
  - A*

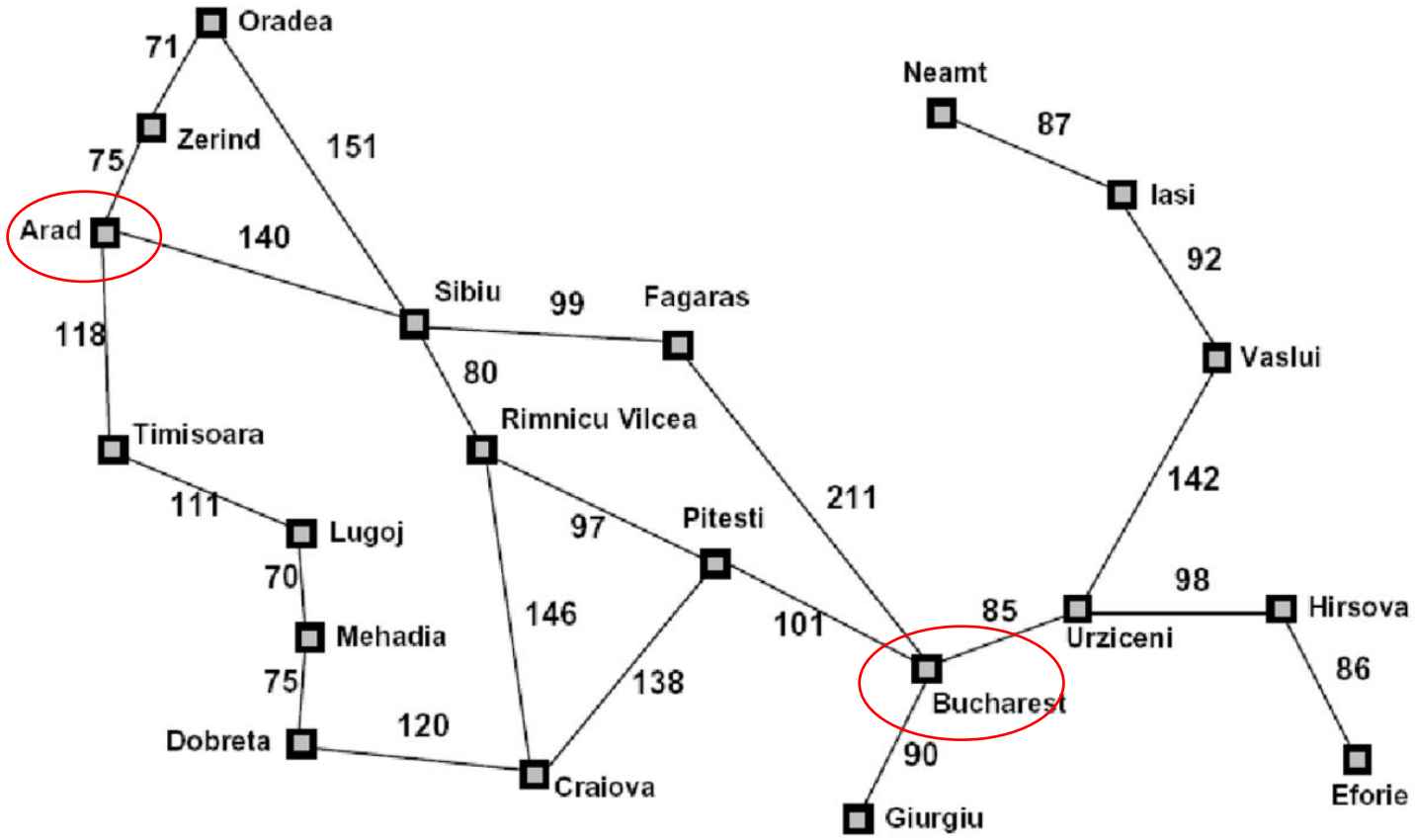
# 启发算法

北京大学 PEKING UNIVERSITY

- 启发是一个估计当前状态离目标状态还有多“远”的方程
- 经典的包括，曼哈顿距离等



# 例子：启发方程 h(x)



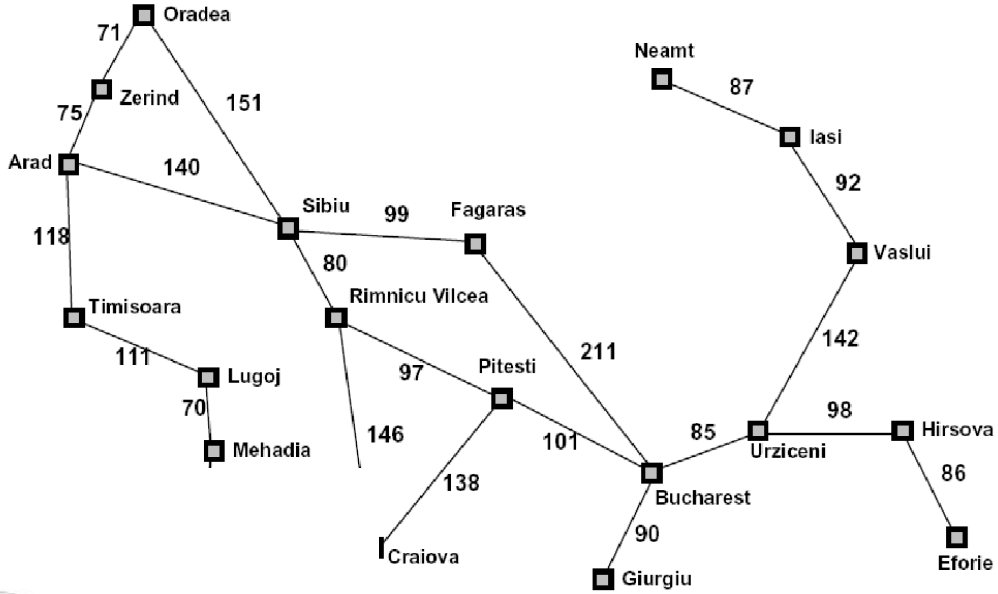

直线距离 h(x)

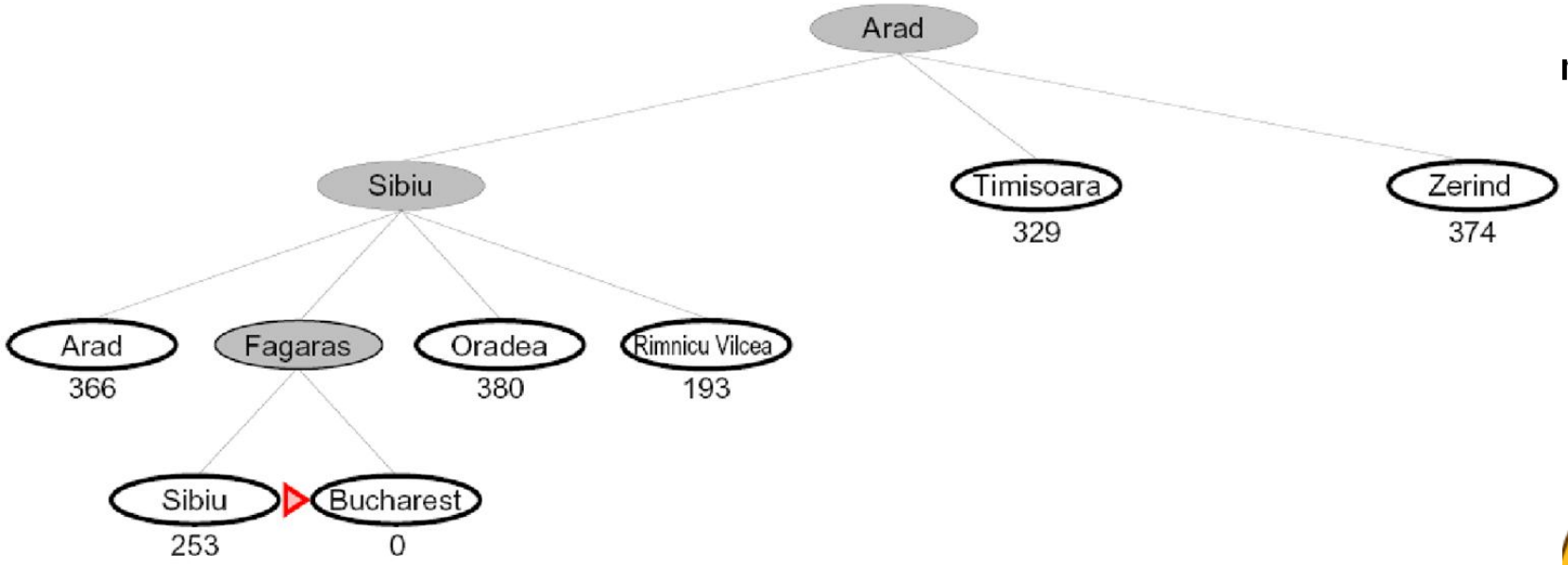
| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara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

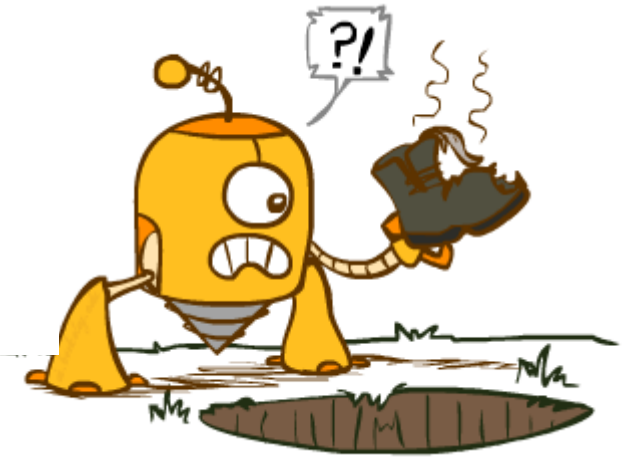
# 贪心搜索

# 贪心搜索

- 永远扩展看起来最近的…

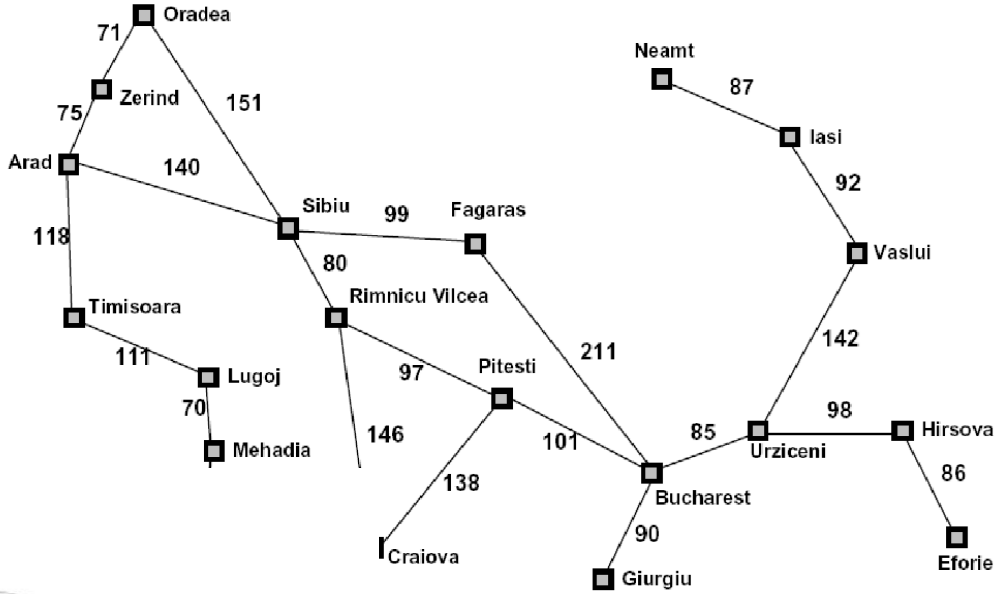


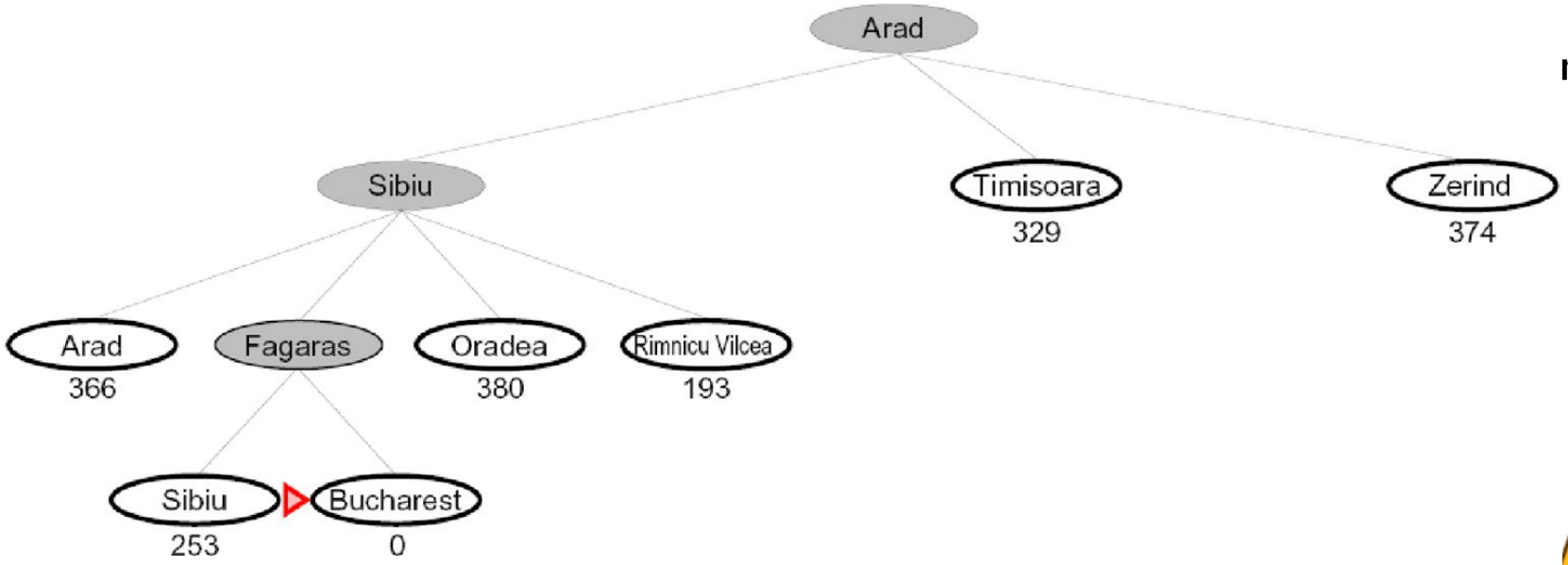



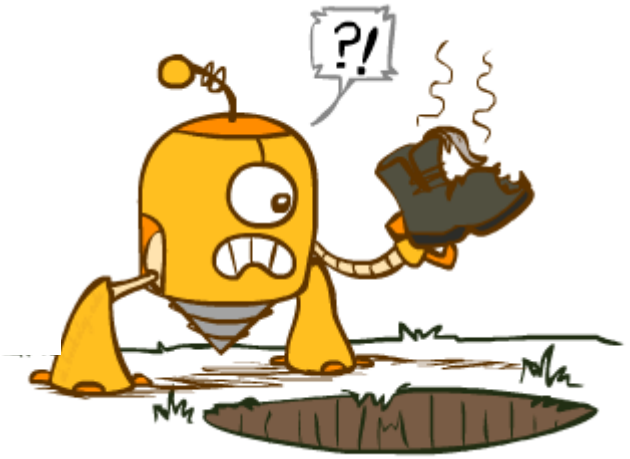



# 贪心搜索

- 永远扩展看起来最近的…



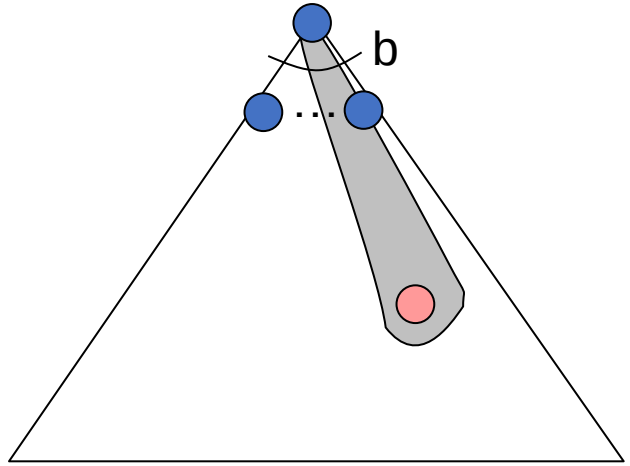



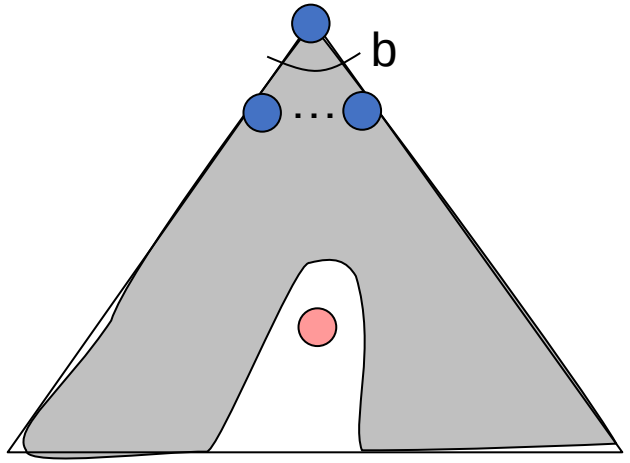




- 最优吗?
  - 不！（1）单次扩展最优并不一定导致全局最优 （2）启发估计不一定准确

# 贪心搜索

- 特点: 扩展一个根据预估离目标最近的点
  - 启发方程：估计到最近的目标的距离
- 经常会发生的情况:
  - 直接带去一个可能错的目标
- 最差情况: 像一个被错误引导的深搜，最后导致要扩展完整个搜索空间才能到目标



# 能不能更好地利用启发?

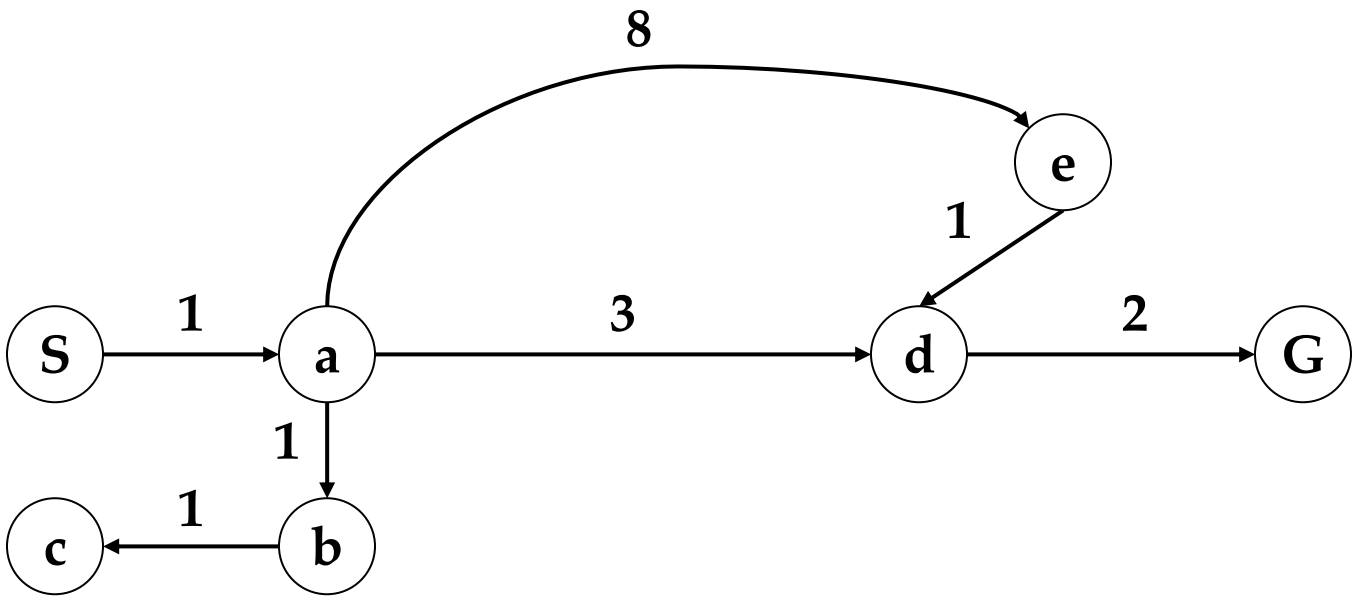
# A* 搜索

UCS

贪心

A*

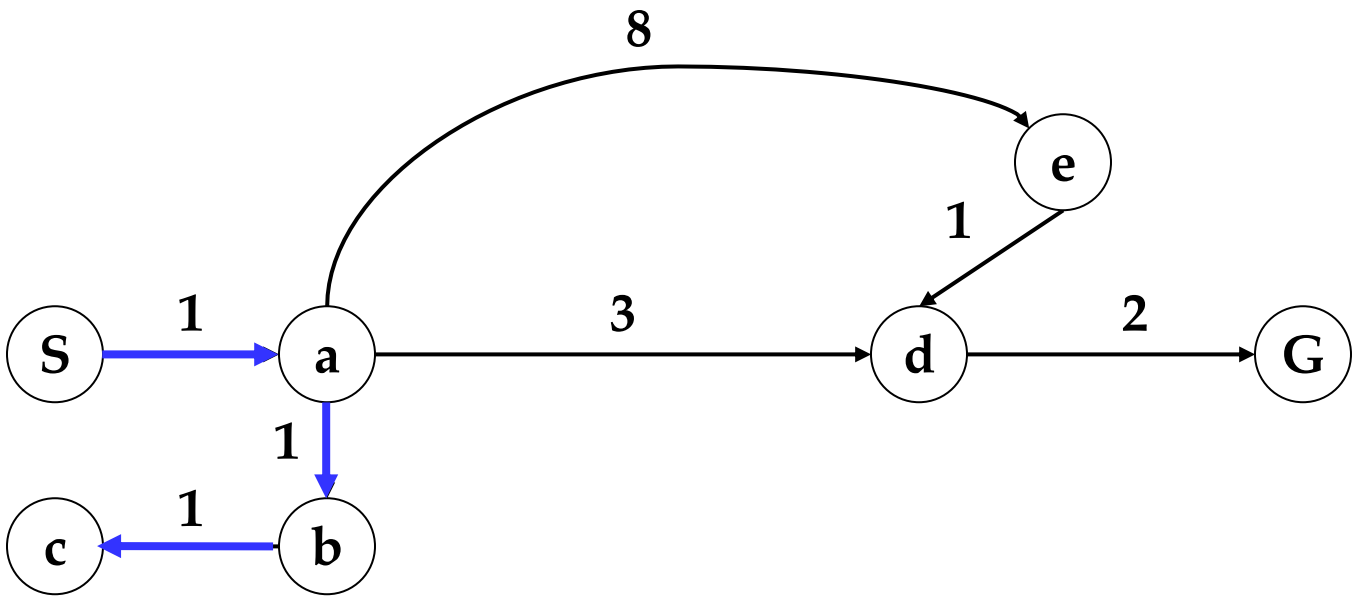
# A*




例子来源: Teg Grenager

# A*

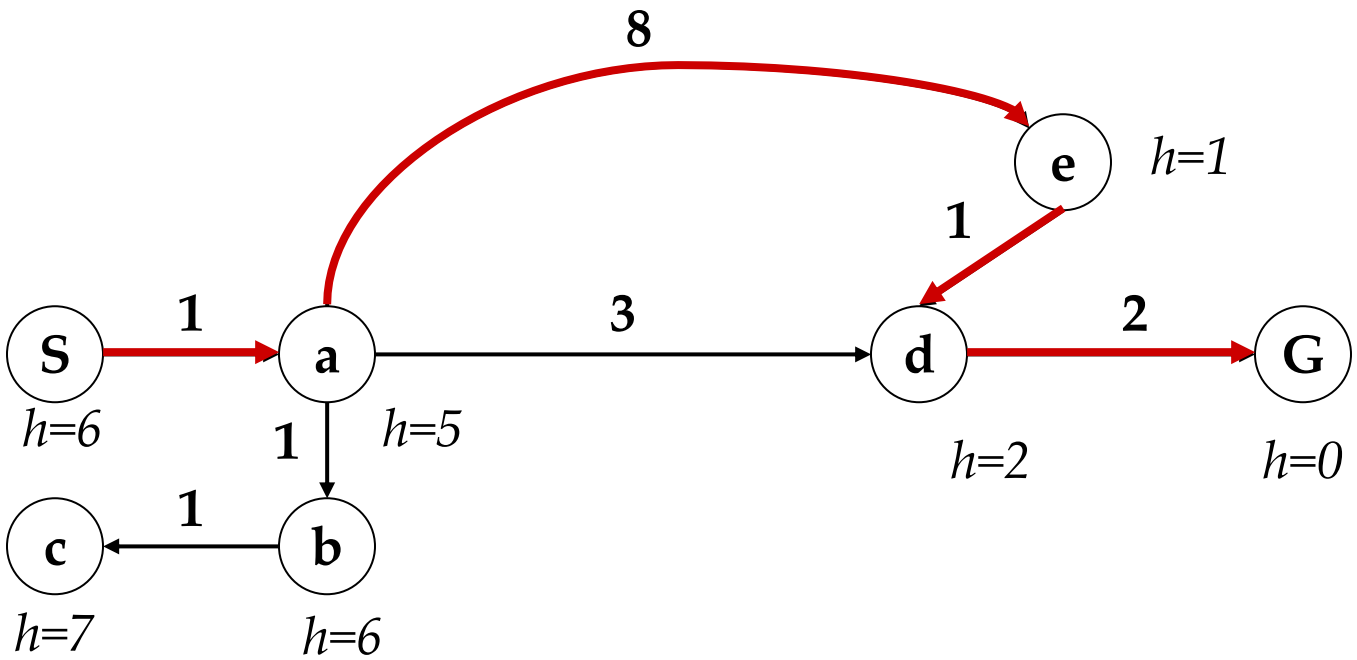
- UCS 向后看，根据至今积累的代价决定顺序 g(n)




例子来源: Teg Grenager

# A*

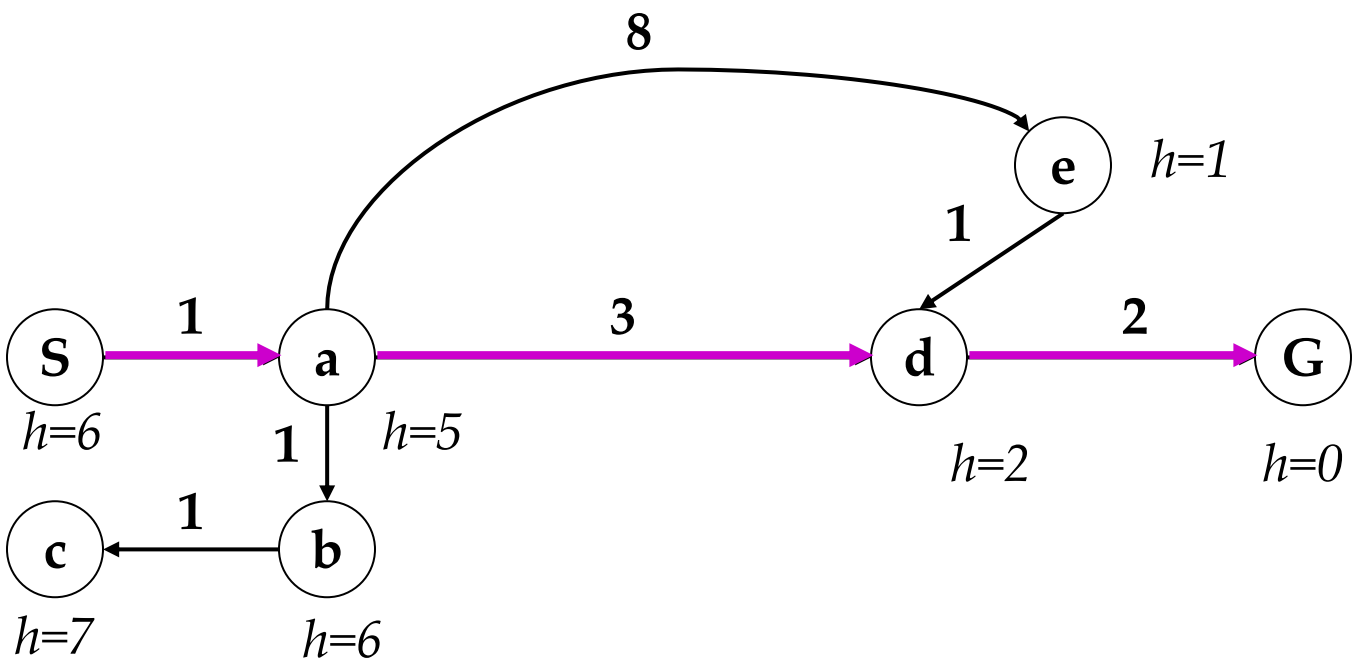
- UCS 向后看，根据至今积累的代价决定顺序 g(n)
- 贪心 向前看，根据离目标还有多远的估计决定顺序 h(n)

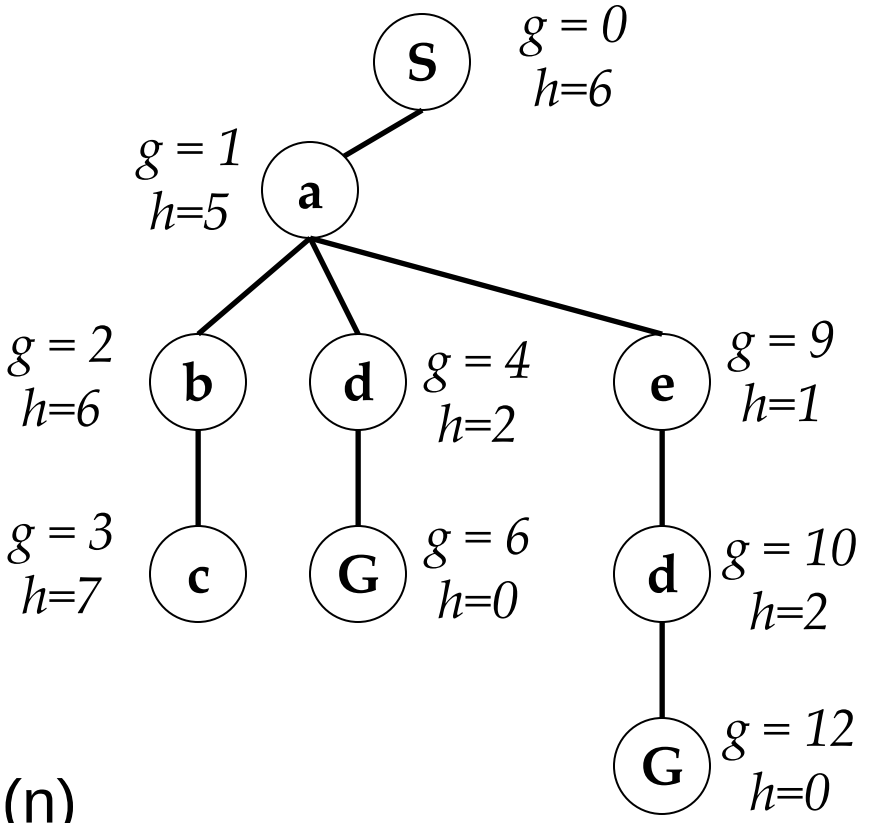



例子来源: Teg Grenager

# A*

- UCS 向后看，根据至今积累的代价决定顺序 g(n)
- 贪心 向前看，根据离目标还有多远的估计决定顺序 h(n)



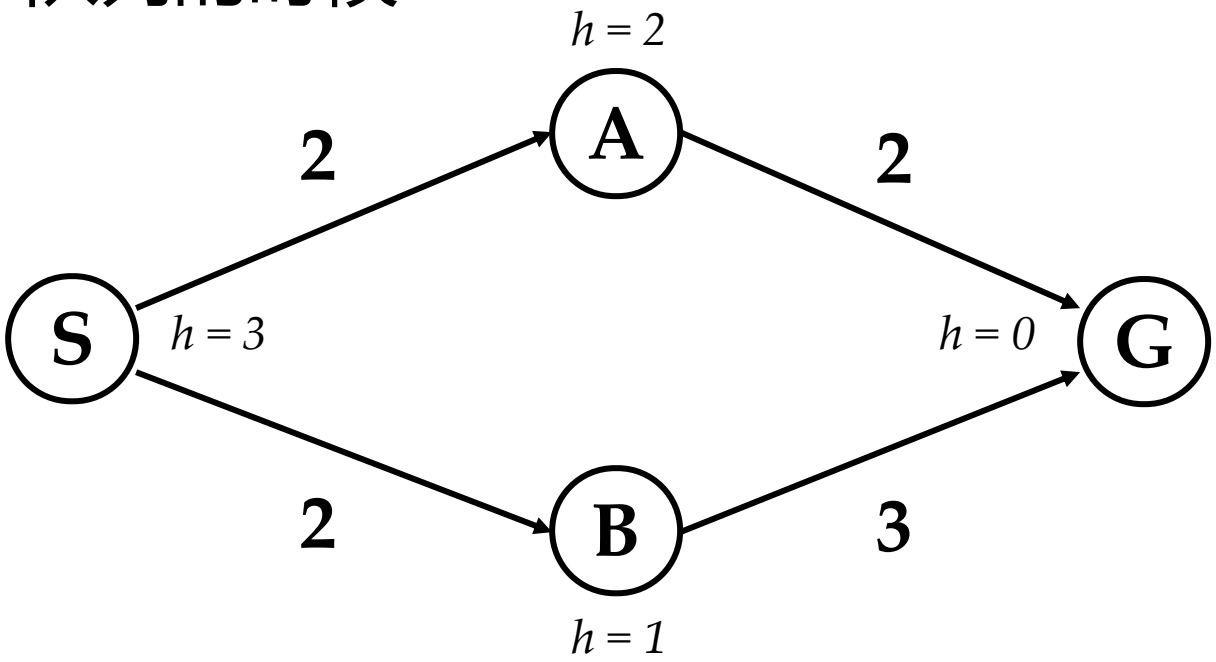




- A* 根据两者的和决定顺序 : f(n) = g(n) + h(n)

例子来源: Teg Grenager

# A*

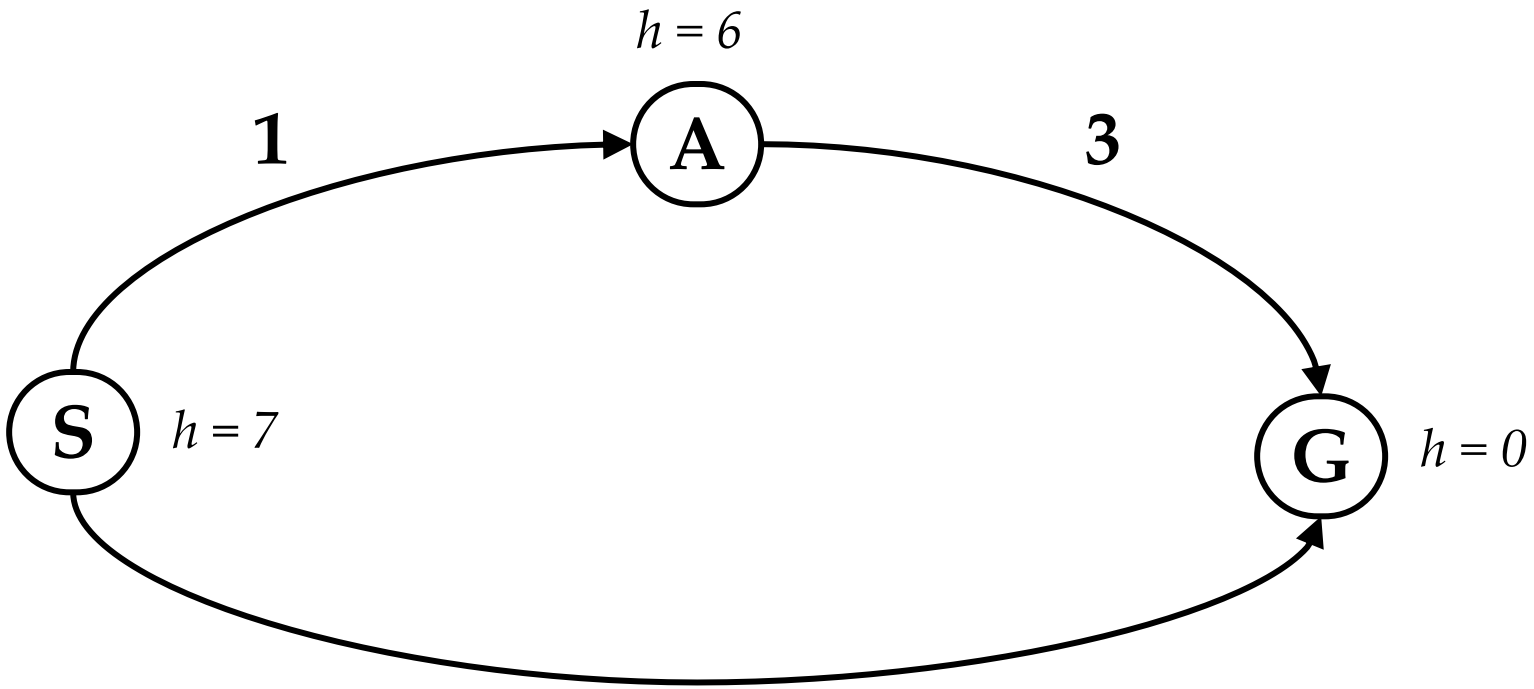
- 我们什么时候该停止?当把一个目标状态放到（优先）队列的时候?




| | g | h | + |
|---|---|---|---|
| ~~S~~ | ~~0~~ | ~~3~~ | ~~3~~ |
| ~~S->A~~ | ~~2~~ | ~~2~~ | ~~4~~ |
| ~~S->B~~ | ~~2~~ | ~~1~~ | ~~3~~ |
| S->B->G | 5 | 0 | 5 |
| S->A->G | 4 | 0 | 4 |

- 不！ 在从（优先）队列里移出一个目标状态时停。

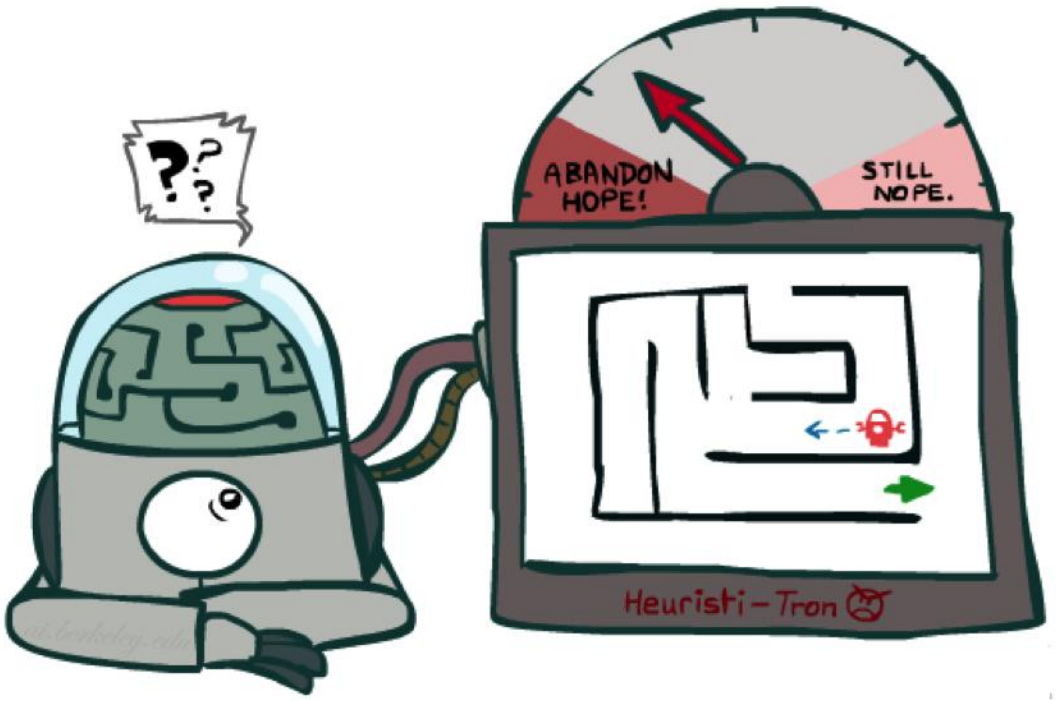
# 任何A*都是最优的吗?



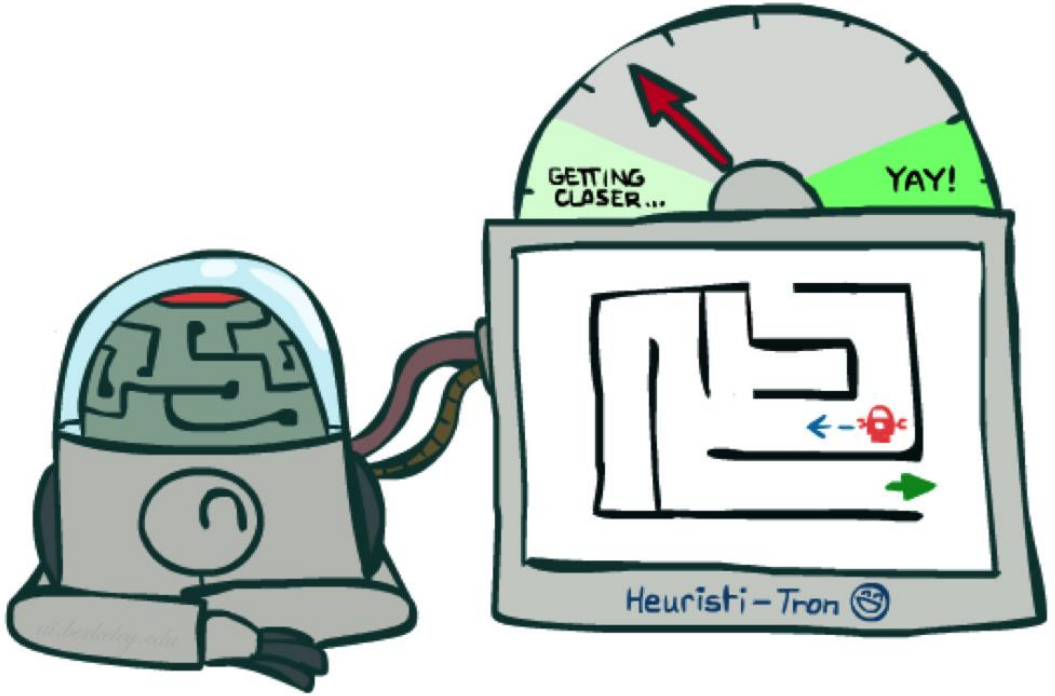

| | g | h | + |
|---|---|---|---|
| S | 0 | 7 | 7 |
| S->A | 1 | 6 | 7 |
| S->G | 5 | 0 | 5 |

- 哪里出错了?
- 实际的代价 < 预估的代价!

# 更“严格”的启发：可接受启发（admissible）




不可接受 (悲观的) 启发
破坏最优性：把一个好的规划困在了队列




可接受 (乐观的) 启发
拖慢不好的规划，但从来不会比真实代价还慢

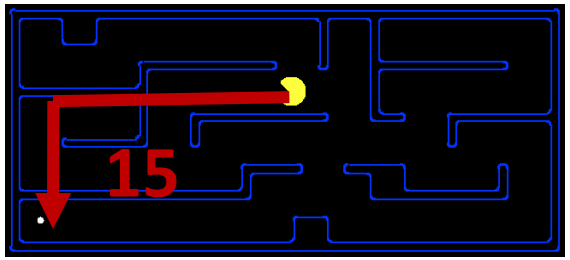
# 可接受启发

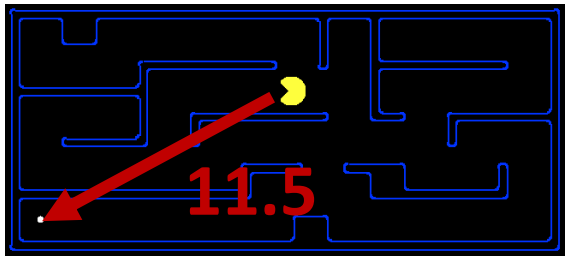
- 一个启发是 $h$ 是 *可接受* (乐观的)，需满足:

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

$h^*(n)$ 是到最近目标的真实代价

- Examples:




0.0

- 在设计A*算法时，最重要的就是设计可接受启发

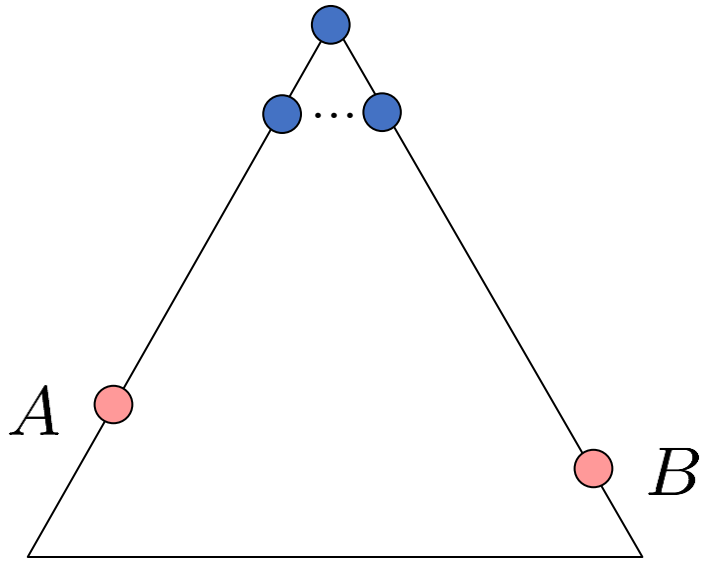
# A* 为什么是最优的（对于搜索树）？

假设:

- A 是一个最优的目标节点
- B 是一个次优的目标节点
- h 是可接受的

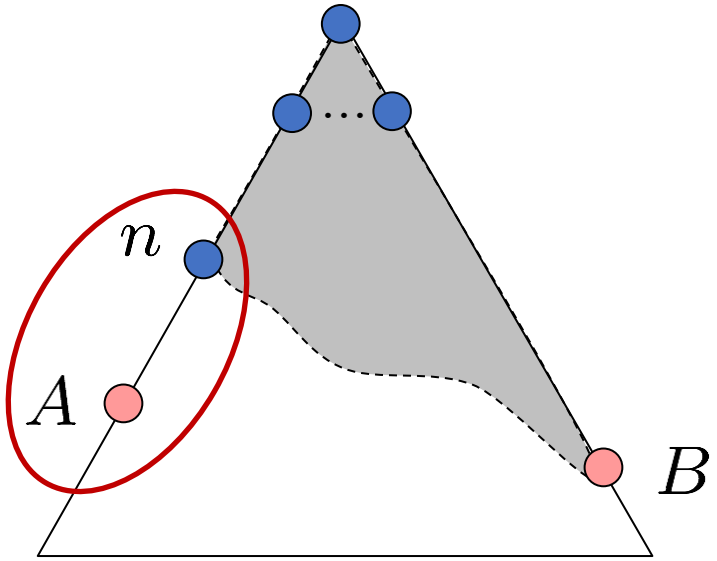
我们需要证明:

- A 一定比 B 更早离开（优先）队列



# A*为什么是最优的（对于搜索树）？

- 假设 B 已经在（优先）队列里了
- A的一些前序节点n也已经在（优先）队列里了 (当然 A也是自己的前序!)
- 需要证明: n比B先扩展
  1. f(n) 小于等于 f(A)




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

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

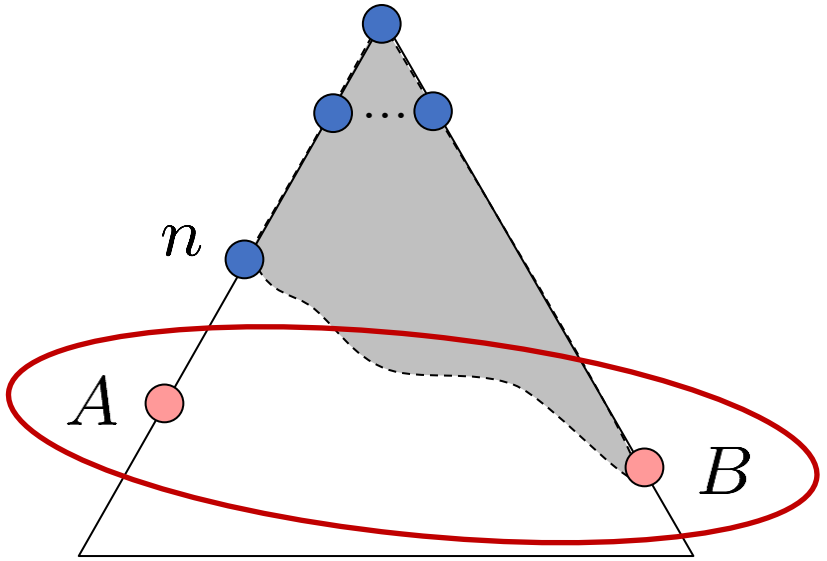
$$g(A) = f(A)$$

h是可接受的

对于目标h = 0

# A*为什么是最优的（对于搜索树）？

- 假设 B 已经在（优先）队列里了
- A的一些前序节点n也已经在（优先）队列里了 (当然 A也是自己的前序!)
- 需要证明: n比B先扩展
  1. f(n) 小于等于 f(A)
  2. f(A) 小于 f(B)




$g(A) < g(B)$

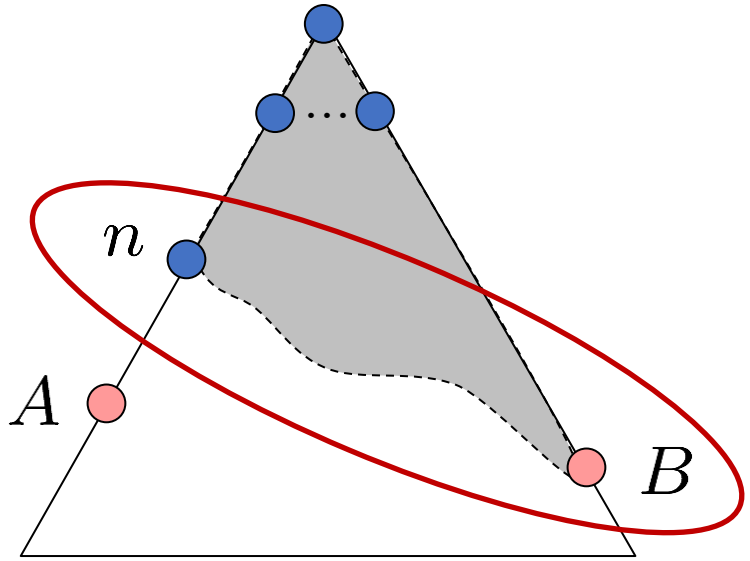
$f(A) < f(B)$

B 是次优的

对于目标 h = 0

# A*为什么是最优的（对于搜索树）？

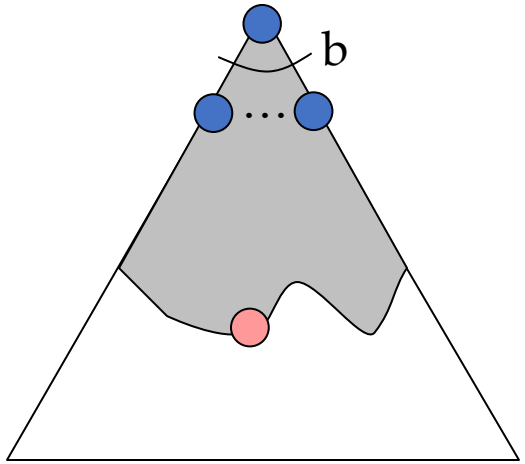
- 假设 B 已经在（优先）队列里了
- A的一些前序节点n也已经在（优先）队列里了 (当然 A也是自己的前序!)
- 需要证明: n比B先扩展
  1. f(n) 小于等于 f(A)
  2. f(A) 小于 f(B)
  3. n 比 B 先扩展
- 所有 A 的前序都比 B 先扩展
- A 比 B 先扩展
- A* 是最优的
- (取h=0，得到UCS最优)



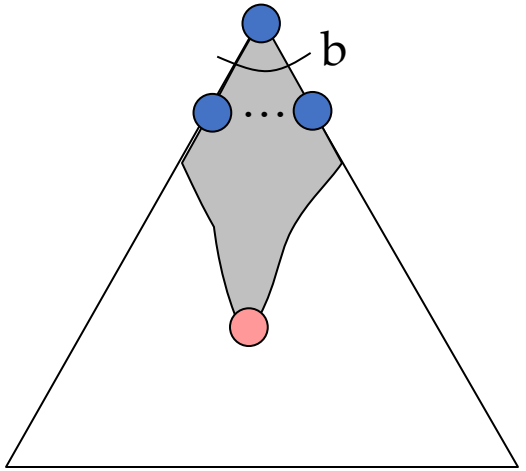

$$f(n) \leq f(A) < f(B)$$

# UCS vs. A*

UCS

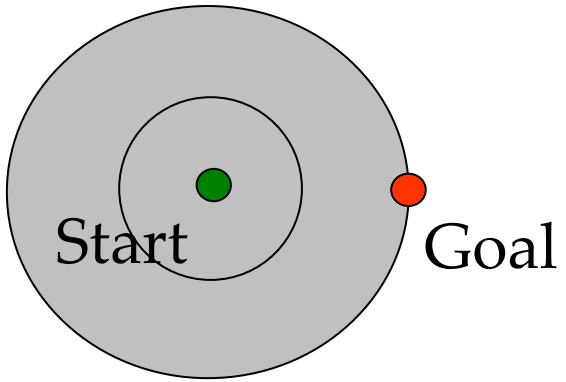



A*

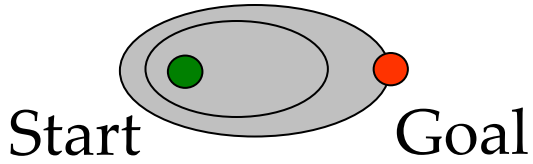


# UCS vs. A*

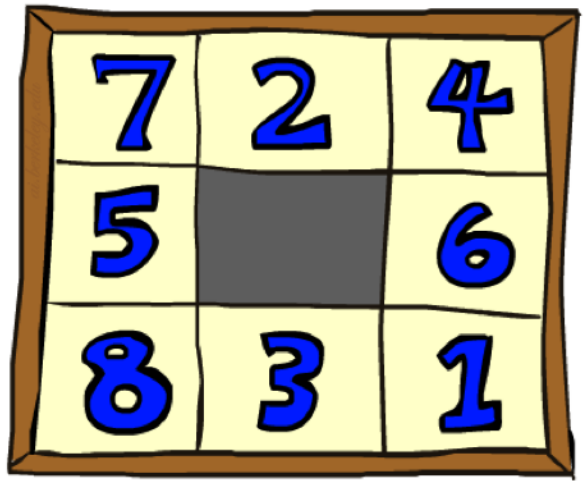
- UCS 平等对待每一个方向（在同样的轮廓上的话）



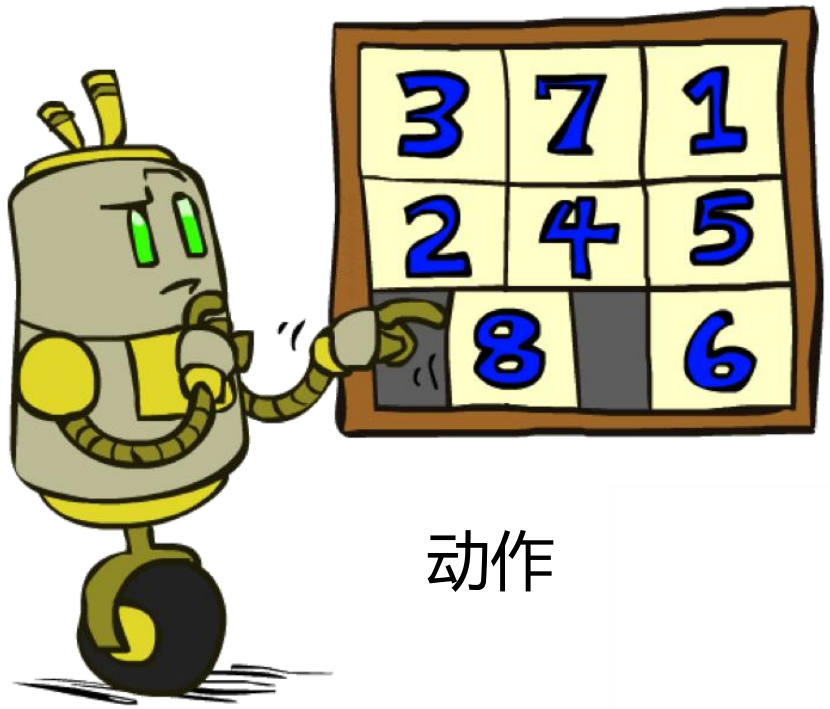

- A* 会比较倾向往目标去靠近，但是也会考虑考虑别的方向保证自己的最优性



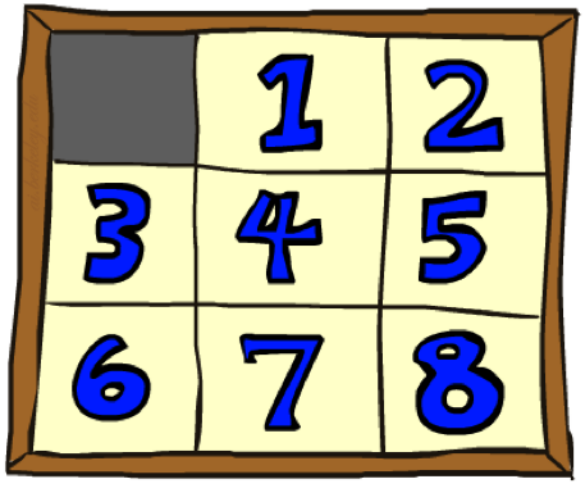
# 例子：8方块




开始状态




动作

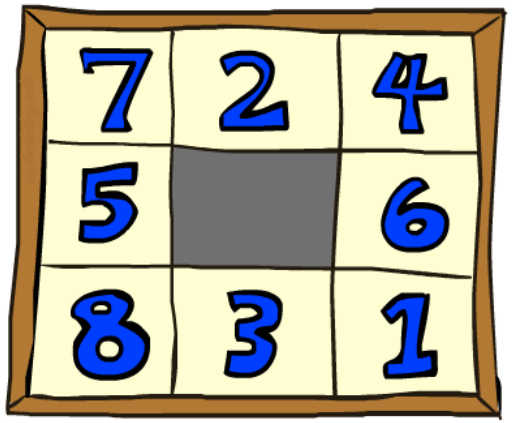



目标状态

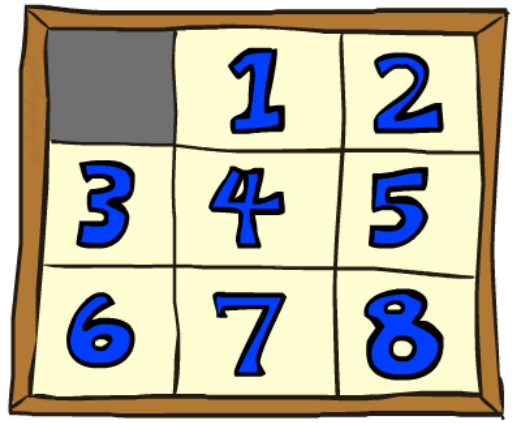
可接受启发?

# 例子：8方块

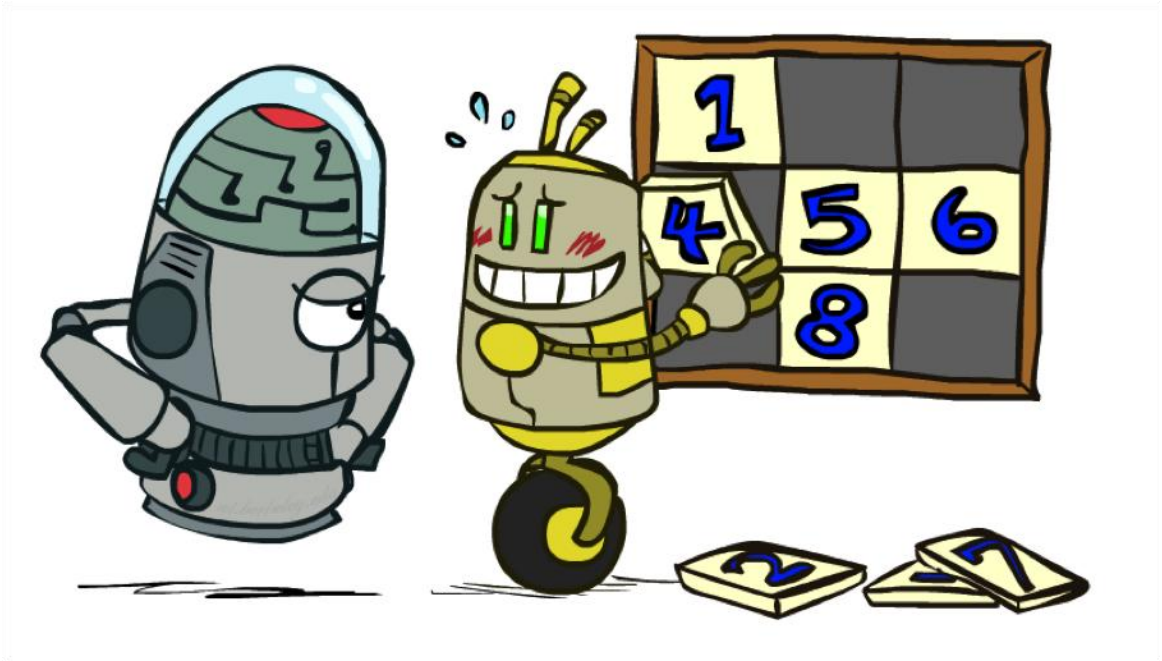
- 启发： 有多少方块错放了?
- 为什么这是可以接受的?
- h(开始) = 8




开始状态




目标状态

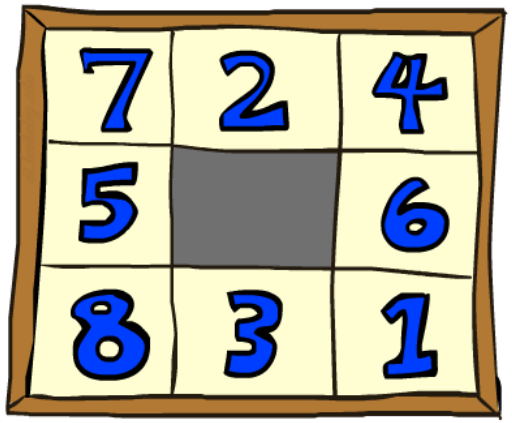



| | 平均扩展的节点数，当最优路线需要 | | |
|---|---|---|---|
| | …4 步 | …8 步 | …12 步 |
| UCS | 112 | 6,300 | $3.6 \times 10^6$ |
| TILES | 13 | 39 | 227 |

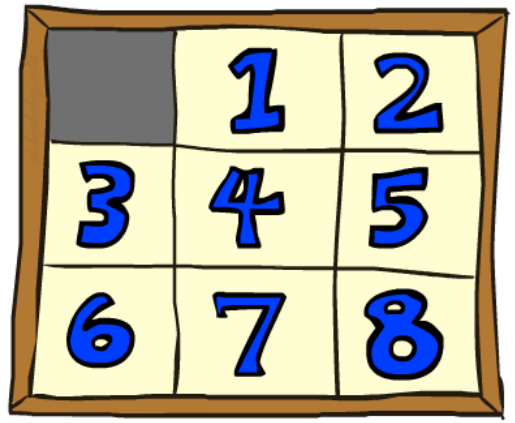
统计数字来自 Andrew Moore

# 例子：8方块

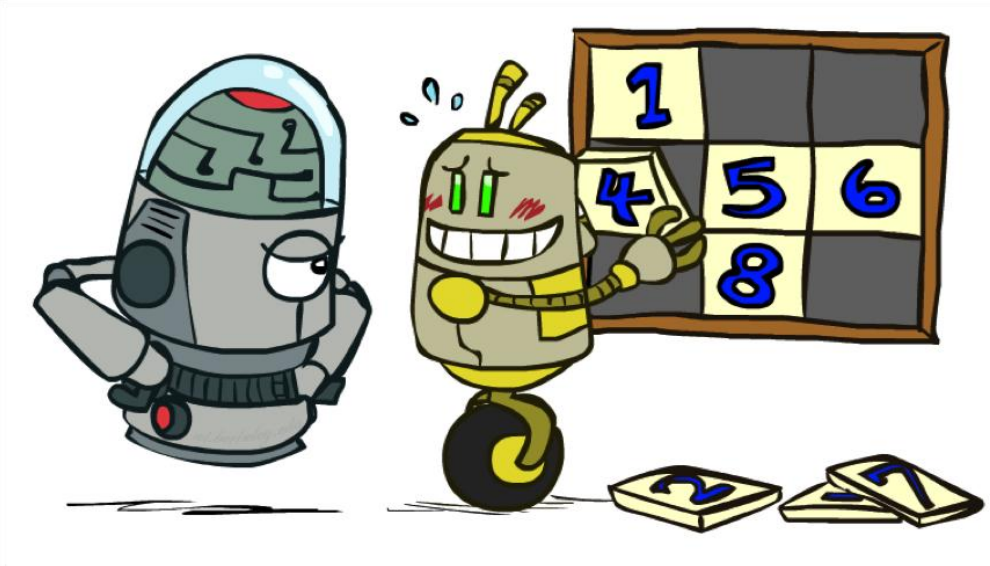
- 如果我们考虑一个简单点的问题？如果我们允许方块重叠
- 总共的曼哈顿距离？为什么这是可接受的？
- h(开始) = 3 + 1 + 2 + … = 18



开始



目标状态




| | 平均扩展的节点书，当最优路线需要 | | |
|---|---|---|---|
| | …4 步 | …8 步 | …12 步 |
| UCS | 112 | 6,300 | $3.6 \times 10^6$ |
| TILES | 13 | 39 | 227 |
| MANHATTAN | 12 | 25 | 73 |

统计数字来自 Andrew Moore

# 例子：8方块

北京大学 PEKING UNIVERSITY

- 我们能不能使用真实代价作为启发?
  - 可以被接受吗（admissible）？
  - 这有什么问题?

- A*: 平衡估计的质量和每一个节点需要计算的量?
  - 当启发越来越接近真实代价的时候，我们可能可以扩展更少的节点，但很有可能我们需要在每一个节点做更多的计算从而达到这个启发

# 小结

- 如何保证搜索的最优性?
- 什么是知情搜索?
- 一个可接受启发需要满足什么条件?
- A*为什么是最优的?
- 如何设计一个好的A*?

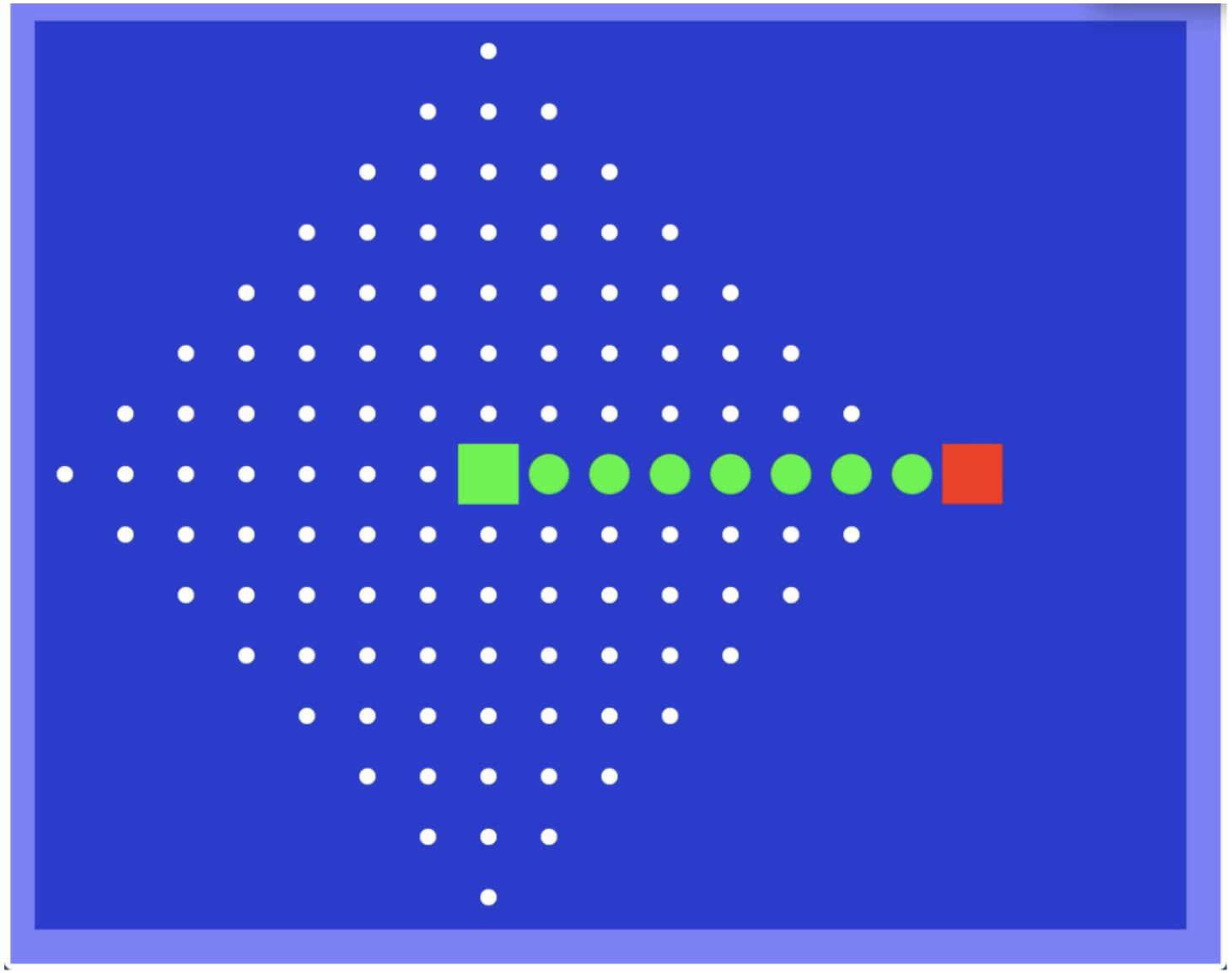
# 预告

- 为什么我们平等对待每一个动作？ -> 启发搜索
- 当有一个对手和我们博弈的时候？ -> minimax和剪枝
- 如果环境的转变是不确定性的？ -> 不确定性和概率推理
- 搜索空间太大了？ -> 蒙特卡洛搜索

# 谢谢

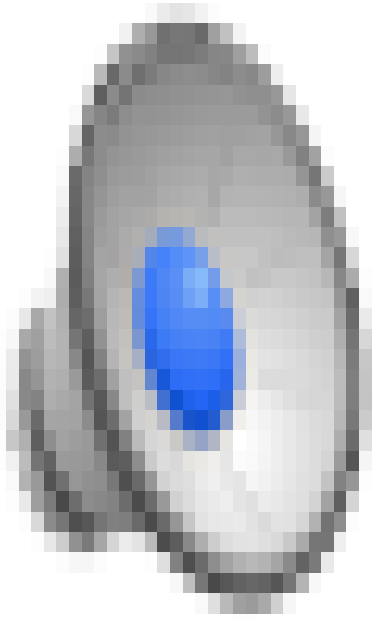
北京大学
PEKING UNIVERSITY

# 这是什么搜索?



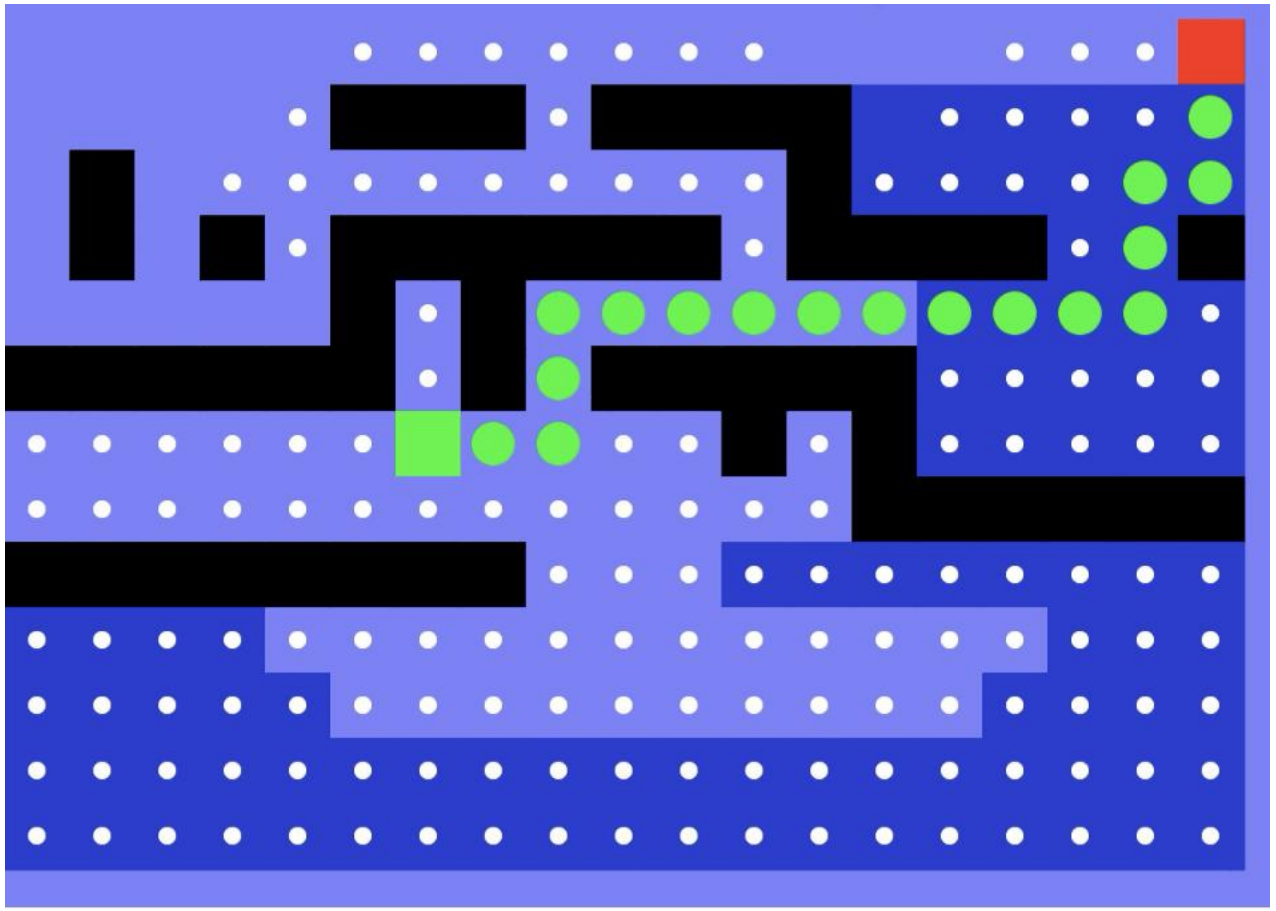
水深一样的情况
(还是要考虑直线距离是小于斜边距离的)

# 这是什么搜索?



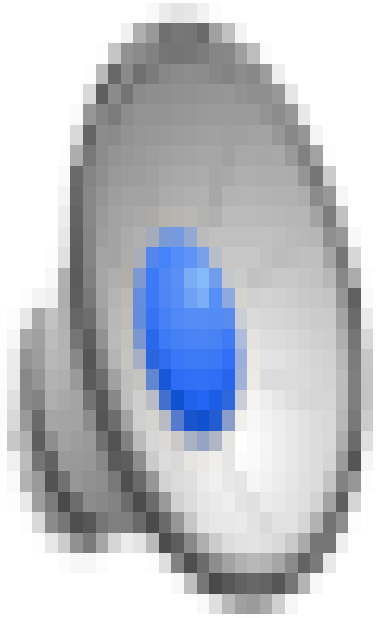
代价不一致的情况下
注意视频里水的深浅情况

# 这是什么搜索?



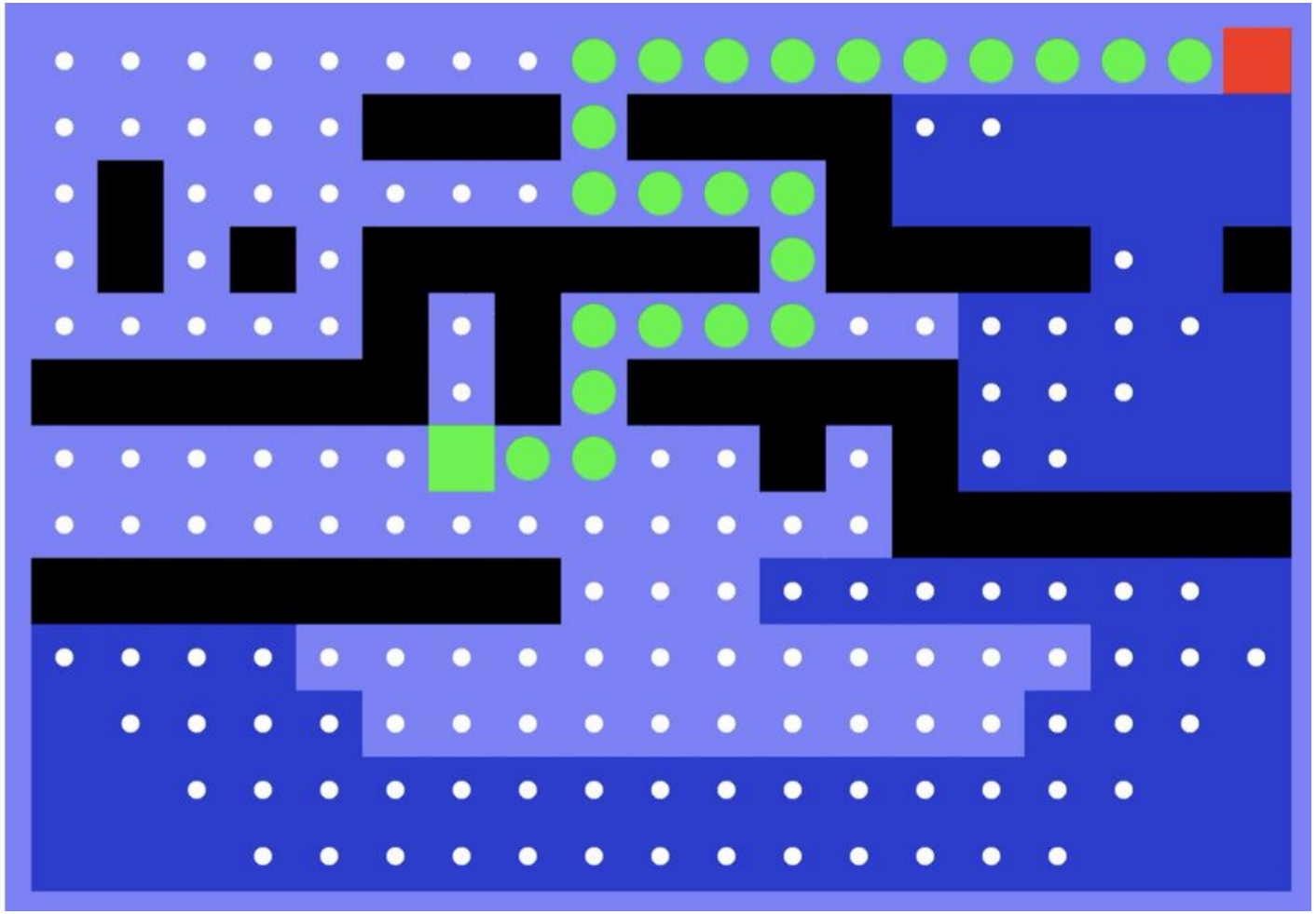
代价不一致的情况下
注意视频里水的深浅情况

# 这是什么搜索?



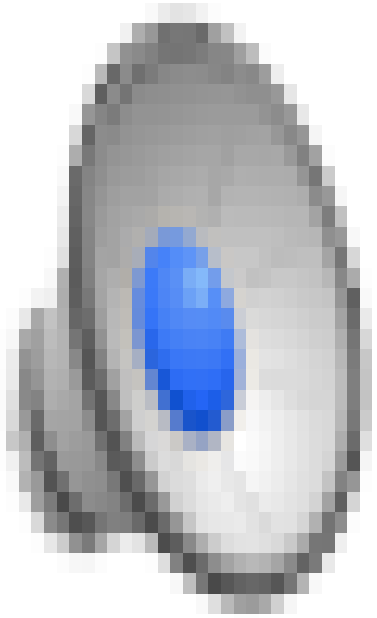
代价不一致的情况下
注意视频里水的深浅情况

# 这是什么搜索?



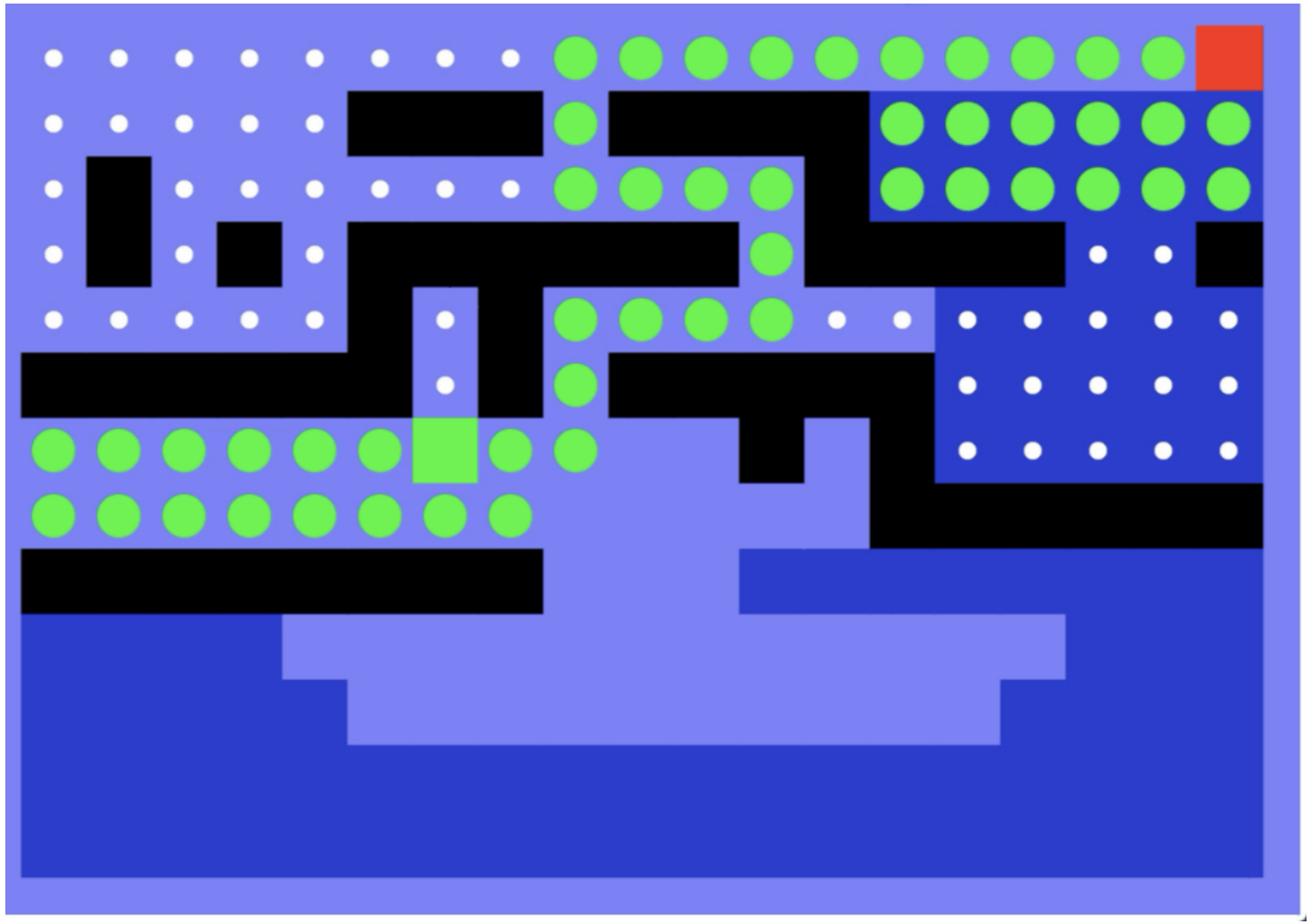
代价不一致的情况下
注意视频里水的深浅情况

# 这是什么搜索?



代价不一致的情况下
注意视频里水的深浅情况

# 这是什么搜索?

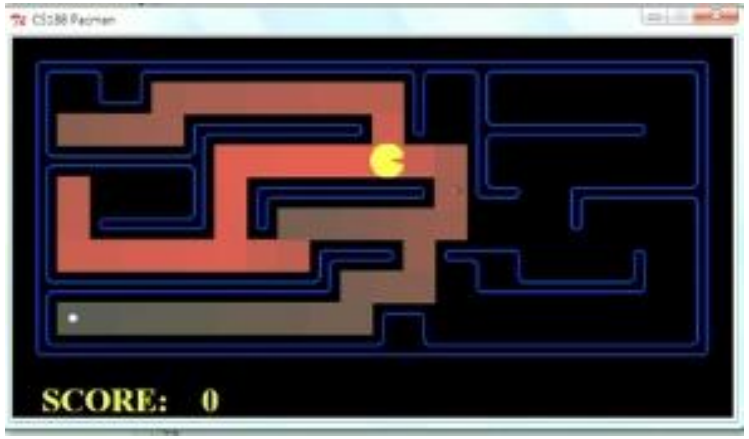


代价不一致的情况下
注意视频里水的深浅情况

# 几种知情算法的比较


贪心


UCS




A*