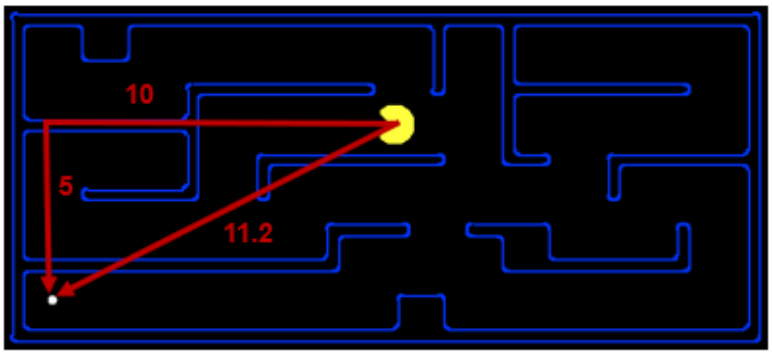
# 启发式搜索(Informed Search)-贪婪算法GBS

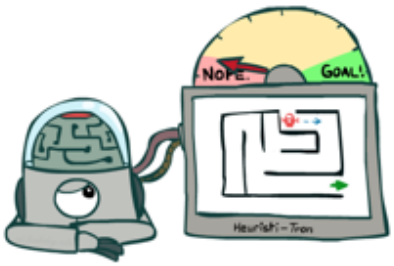
## 一、启发式搜索和启发式函数

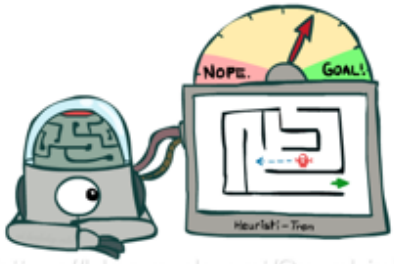
> 百科：启发式搜索(Heuristically Search)又称为有信息搜索(Informed Search)，它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的，这种利用启发信息的搜索过程称为启发式搜索。
> ***启发式搜索***的节点一般是基于f(n)来进行扩展的，我们如果了解无信息搜索中代价一致算法（UCS）的话，UCS中也有一个g(n)。启发式搜索中是通过f(n)被选择性扩展（一般使用优先队列），一致代价搜索中使用g(n)来选择性扩展（也是优先队列）。但是启发式搜索中f(n)含义要比g(n)丰富，***g(n)仅仅表示从当前节点到下一个节点的开销，但是f(n)可以表示从当前节点到目标节点的估计。***

启发式函数（对于大多数最佳优先搜索）：h(n)=节点n到目标节点的最小代价路径的代价估计值
启发式搜索：采用了f(n)（即启发式函数）的搜索策略。

- 启发策略:
  - ***估计***一个状态到目标距离的函数
  - 问题给予算法的额外信息，为特定搜索问题而设计
  - 例: Manhattan distance, Euclidean distance for pathing



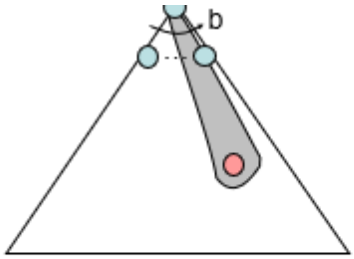


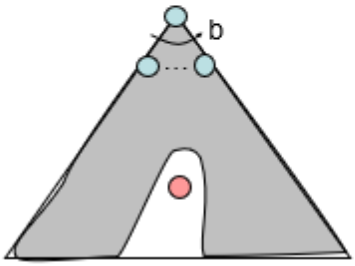

## 二、贪婪算法（贪婪最佳优先搜索）greedy best-first search （GBS）

贪婪算法的设计目的就是想要从当前节点扩展离目标节点最近的一个节点。这个算法只用到了最简单的启发式函数***f(n)=h(n)***，在罗马尼亚问题中，使用的是当前地点到目的地的直线距离（这个信息不能由问题本身的描述计算得到，而且这个信息是有用的——因为和实际路程相关，所以是一个有用的启发式），在此问题中，GBS搜索代价最小，但是不是最优。

GBS可能陷入死循环，但是它在有限状态空间的图搜索下是完备的，其他情况则不是。其时间和空间复杂度都是$O(b^m)$，其中m是搜索空间的最大深度。

- 策略: 扩展你认为最接近目标状态的节点
  - 启发式: 对每个状态估计到最近目标的距离
  - 只使用启发函数 f(n)=h(n) 来评价节点
- 通常情况:
  - 最佳优先使你直接（或很快）到达目标
- 最坏情况:
  - 类似DFS



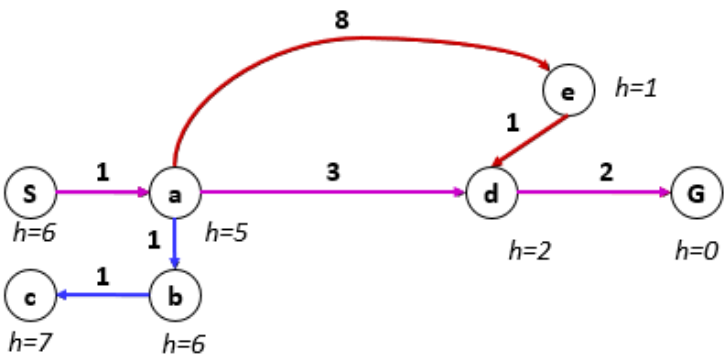



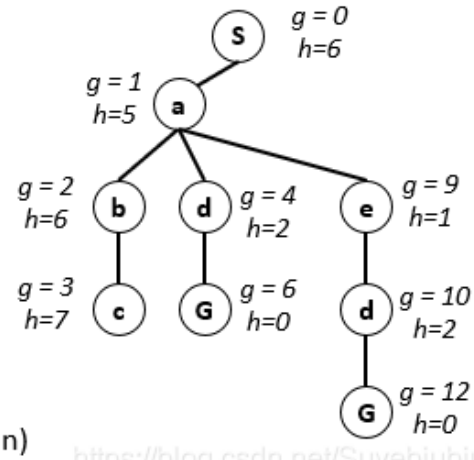



# 三、A*搜索（结合UCS和GBS）

*A搜索不同于贪婪算法，贪婪算法仅仅利用了启发式函数h(x)，A算法结合了代价一致算法UCS和贪婪算法的优点。A算法可以看到到达此节点已经花费的代价（ g(x) ）， A搜索的启发式函数变为了f(n)=g(n)+h(n)。整体来说，A算法既是完备的也是最优的。**

- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)

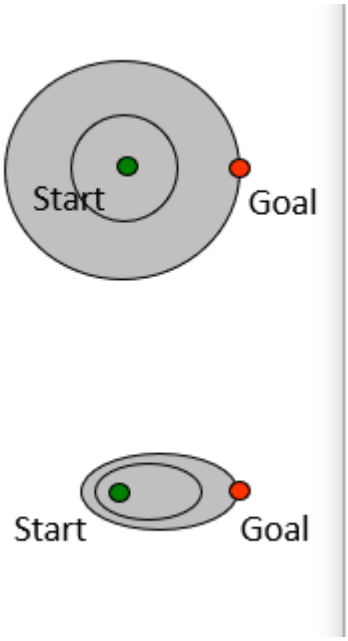



- A* Search orders by the sum: f(n) = g(n) + h(n)

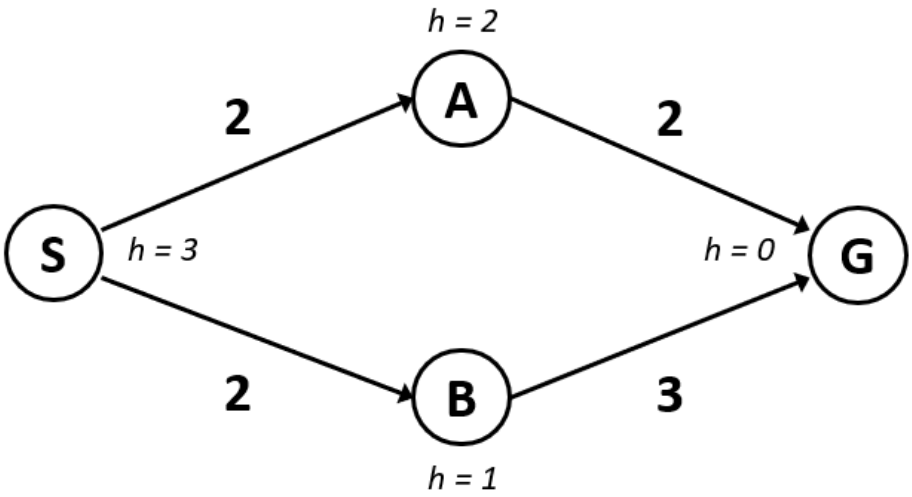


那么我们可以探讨下，***代价一致搜索和A搜索有啥区别呢***? ?我们通过观察下图可以看出，***代价一致搜索在所有方向上是等可能的扩展，而A搜索方式主要是朝着目标扩展，而且可以保证最优性***? .启发函数越接近真实的耗散，将扩展越少的节点，但通常会在每个节点计算启发函数本身时有更多的计算。




A*搜索算法结束的条件是什么?

- 当目标入列时，应该停止吗?



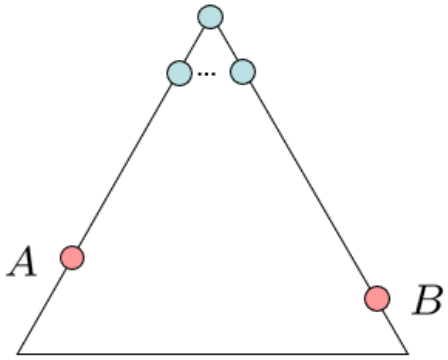

- 不: 只有目标出列时才停止!



怎么判证明A* 树搜索的最优性? （这个容易出证明题）

假定:

- A 最优目标节点
- B 次优目标节点
- h 可采纳的



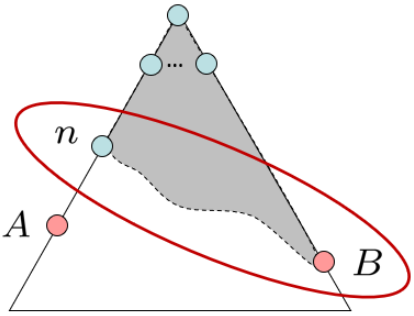

结论:

- A在B之前离开边缘集合



证明:

- 假设 B 在边缘集合上
- A的某个祖先节点n也在边缘集合上 (maybe A!)
- 那么 n 将在B之前被扩展
  1. f(n) is less or equal to f(A)
  2. f(A) is less than f(B)
  3. *n* expands before B
- A的所有祖先在B之前扩展
- A在B之前扩展
- A*是最优的

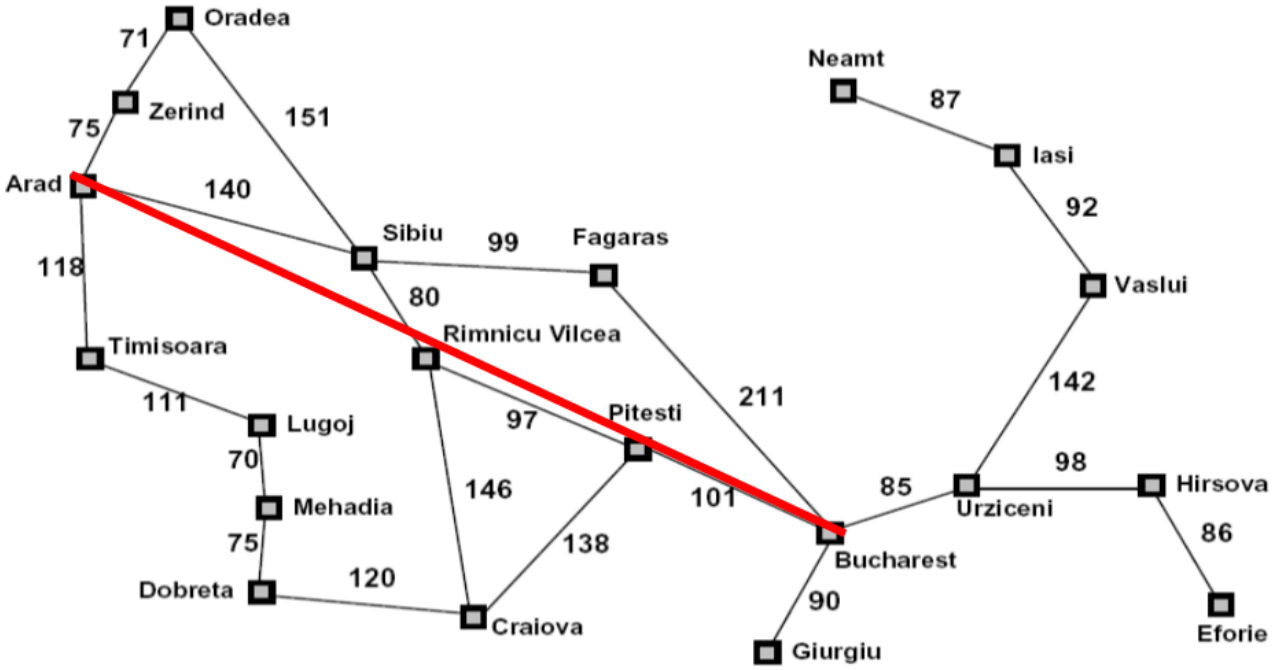



$$f(n) \leq f(A) < f(B)$$



# 四、怎么选择一个好的启发式函数

其实我们最大问题并不是理解启发式函数，而是怎么去想出最好的可采纳的启发函数。通常情况下可采纳启发函数都是松弛问题的解的耗散问题。其实不能被采纳的启发函数也不是一无是处，常常也是非常有用的。





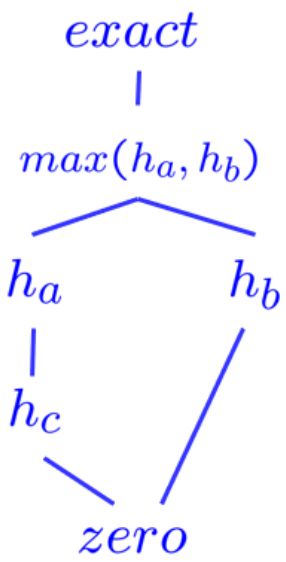

- 占优势: $h_a \geq h_c$ if

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

- 可采纳的启发函数集合:

$$h(n) = max(h_a(n), h_b(n))$$

- Trivial heuristics
  - 底层是零启发式(what does this give us?)
  - 顶层是精确的启发式






优势可以直接转化为效率：使用ha的A*算法绝不会比使用hc扩展更多的节点。

论证：每个f(n)<C*的节点都会被扩展，那么h(n)*<C*-g(n)的节点必将被扩展。ha>=hc，那么每个被使用ha的A*算法扩展的节点必定也会被使用hc的A*所扩展，而hc还可能引起其他节点的扩展。

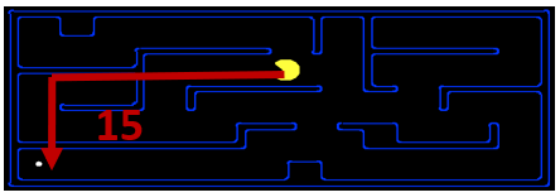
因此，***使用值更高的启发函数总是更好的。前提可采纳的启发函数***

- 启发函数 $h$ 是 *可采纳的*，那么：

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

其中 $h^*(n)$ 是到最近目标的真实耗散

- 例：




- 想出可采纳的启发函数是A*算法实际使用中的重点