A\*算法，A\*（A-Star)算法是一种静态路网中求解[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84/6334920" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)最有效的**直接**搜索方法，也是解决许多搜索问题的[有效](https://baike.baidu.com/item/%E6%9C%89%E6%95%88" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)算法。算法中的[距离](https://baike.baidu.com/item/%E8%B7%9D%E7%A6%BB/16370594" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)估算值与实际值越接近，最终[搜索](https://baike.baidu.com/item/%E6%90%9C%E7%B4%A2/1806" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)速度越快。

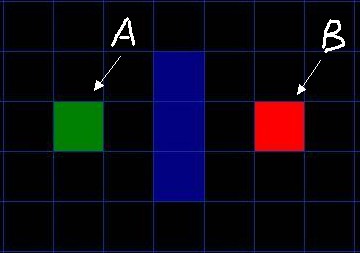
公式表示为： f(n)=g(n)+h(n)

其中， f(n) 是从初始[状态](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81/5809342" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)经由状态n到目标状态的代价估计，

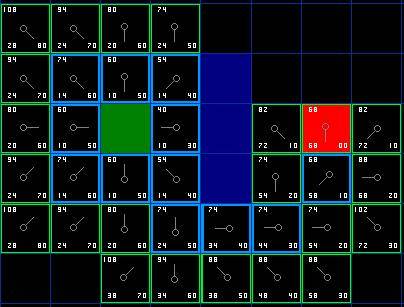
g(n) 是在[状态空间](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81%E7%A9%BA%E9%97%B4" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)中从初始状态到[状态](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81/5809342" \t "https://baike.baidu.com/item/A%2A%E7%AE%97%E6%B3%95/_blank)n的实际代价，

h(n) 是从状态n到目标状态的最佳路径的估计代价。

（对于路径搜索问题，状态就是图中的节点，代价就是距离）



1. 从起点A开始, 把它作为待处理的方格存入一个"开启列表", 开启列表就是一个等待检查方格的列表.
2. 寻找起点A周围可以到达的方格, 将它们放入"开启列表", 并设置它们的"父方格"为A
3. . 从"开启列表"中删除起点 A, 并将起点 A 加入"关闭列表", "关闭列表"中存放的都是不需要再次检查的方格



1. 把它从 "开启列表" 中删除, 并放到 "关闭列表" 中.

5. 检查它所有相邻并且可以到达 (障碍物和 "关闭列表" 的方格都不考虑) 的方格. 如果这些方格还不在 "开启列表" 里的话, 将它们加入 "开启列表", 计算这些方格的 G, H 和 F 值各是多少, 并设置它们的 "父方格" 为 C.

6. 如果某个相邻方格 D 已经在 "开启列表" 里了, 检查如果用新的路径 (就是经过C 的路径) 到达它的话, G值是否会更低一些, 如果新的G值更低, 那就把它的 "父方格" 改为目前选中的方格 C, 然后重新计算它的 F 值和 G 值 (H 值不需要重新计算, 因为对于每个方块, H 值是不变的). 如果新的 G 值比较高, 就说明经过 C 再到达 D 不是一个明智的选择, 因为它需要更远的路, 这时我们什么也不做.

源代码：



