算法思想：

1. A*算法：
   a. 首先，对输入的初始状态和目标状态进行一些检测，如判断障碍物位置是否相同、始末状态是否相同等，排除一些异常情况。
   b. 从初始状态 A 开始，将其作为一个待处理点加入一个开启列表中(OPEN)。并计算当前状态 A 的 f(n)=g(n)+h(n)的值。(g 为路径耗散值，h 为启发函数值)
   c. 将开启列表中的初始状态取出，作为起点，并放入关闭列表(CLOSE)。寻找起点周围所有的可达状态结点，也将他们加入开启列表。这些可达状态点的父状态指向起点状态，并将到达该状态的行为记录下来，利用 hash 函数进行去重。
   d. 接着，从开启列表中寻找 f 值最小的状态结点，重复 c 步骤，直到找出的起点结点的 h 值为 0。此状态即为目标状态。
   e. 根据目标状态，沿父状态指针一路搜索到初始状态，并同时将到达当前状态的行为压栈。最后，依次出栈，既可以得到从初始状态到达目标状态的动作序列。

2. IDA*算法：
   a. 首先，对输入的初始状态和目标状态进行一些检测，如判断障碍物位置是否相同、始末状态是否相同等，排除一些异常情况。
   b. 从初始状态 A 开始，将其作为一个待处理点加入一个开启列表中(OPEN)。并计算当前状态 A 的 f(n)=g(n)+h(n)的值。(g 为路径耗散值，h 为启发函数值) 将该状态结点的 f 值设为阈值 f(max)。
   c. 将开启列表中的顶部状态取出，作为起点。寻找起点周围的可达状态结点中 f 值小于截断值的状态，对其中具有最小耗散值 f 的状态进行搜索。
   d. 判断当前状态是否为目标状态，如果是，就返回；否则，将到达当前状态的动作压入栈中，并继续 c 步骤，同时，利用 hash 函数进行状态去重。
   e. 依次弹出动作栈中的执行动作信息即可。

算法的空间复杂度：

哈希表大小固定为 10000。每个状态约 160B，优先队列空间具有灵活性，可扩展，空间复杂度为 $O(4*5^{n-1}+10000)$。

算法的时间复杂度：

A*：最坏情况下，算法的时间复杂度为 $O(5^n ln(n))$，n 为步数。平均大概有 1/5 左右的重复状态。

IDA*：最坏情况下，算法的时间复杂度为 $O(m5^n ln(n))$，n 为步数，m 为重来次数。n<=30 时，m<=3，影响不大。平均大概有 1/5 左右重复状态。

实验结果说明：

上图是在给定的 5 步执行测试用例下跑出的结果。