人工智能基础第一次编程
实验报告

自96 曲世远 2019011455

1.作业要求

两种搜索方式均已在code.py中实现

求得的解如下：

[[2, 2], [2, 3], [1, 3], [0, 3], [0, 2], [0, 1], [0, 0], [1, 0], [2, 0], [2, 1], [2, 2], [2, 3], [1, 3], [1, 2], [0, 2], [0, 1], [1, 1], [2, 1], [3, 1], [3, 2], [3, 3], [2, 3], [2, 2], [1, 2], [1, 1], [2, 1], [2, 2], [1, 2], [0, 2], [0, 1], [1, 1], [1, 0], [2, 0], [2, 1], [2, 2], [1, 2], [1, 1], [2, 1], [3, 1], [3, 2], [2, 2]]

易知是通过有信息搜索的方法会更好。从常理上来说，无信息搜索就是一种蛮力的遍历算法（尤其是本题中我使用的bfs算法）其在层数较低时可以较为高速的运行并得到答案，但在需要求解层数较高的问题时，就会带来栈大小不足以及时间消耗过高的问题。而有信息搜索则结合了问题本身的条件，是一种有目的的搜索方式，显然会比蛮力的方法更加高效。

本着尝试的心态，我使用Google Colab尝试了bfs方法，在经过一个小时的搜索之后，其仅达到了15层的深度，显然是不可取也并不实用的。

而从计算量以及算法设计上来讲，有信息搜索虽然仍然具有指数的计算的复杂度，但是已经相比无信息搜索极大地提高了搜索效率，并且同样的保证了完备性与最优性。尤其是A*算法，还同时减少了分支因子，因此在有信息搜索中也是效率较高的算法。

在仔细思考了算法运行时间d的问题后，我觉得本程序的复杂度问题在于使用了List的搜索而非哈希查找，如果可以将List中的元素进行哈希后存储，再通过哈希表实现o(1)的查找，应当可以实现一个o(n)->o(1)的复杂度简化，应该是可以实现有/无信息搜索的快速实现。