A Star Search

A\*搜索

问题：

对于的矩阵

点可以与上下左右的相邻点交换位置，除此之外不能随意改变位置，将该矩阵变成

求最少变换次数以及变化经过，若将矩阵的初始状态看作起点，最终状态看作终点，即搜索到的最短路径。

本问题的原型是“八数码问题”。

解法：

与之前问题不同，本问题将每种矩阵状态看作一个节点，是一种时间上的状态搜索。本文用A\*搜索来解决该问题，A\*算法是一种启发式搜索，与DFS和BFS这种无差别搜索不同，A\*算法设置一个评价函数来计算节点的搜索代价（到目标的距离），优先搜索那些离目标最近的点，从而提高搜索效率。

A\*算法的评价函数，其中是节点，表示点到的评价距离，表示从节点到节点的实际最短距离，表示从点到节点的估算距离。在A\*算法的等待队列中，总是优先选取最小的点进行搜索。

在本问题中矩阵的估算距离为矩阵与在对应位置（其中）上不同数字的数量之和，：

对于下面的矩阵，，：

与之前的问题不同，本问题中的节点是一种矩阵状态。之前的解法中我们用染色的方式来标记节点是否被访问过，编码实现时可以用数组下标来标记节点的颜色。而矩阵状态是无法作为数组下标的，不过我们可以用哈希表来记录矩阵状态是否被访问过，以及从节点到达的节点距离。矩阵可以通过下面这两种方式分别映射为字符串或数字（string和int都可以作为哈希表键值）：

设置表、表和g分数表。g分数表是一个哈希表，用来存储每个节点的实际距离。表是一个优先队列，与之前搜索算法中的队列功能相同，用于管理等待访问的节点，但是我们需要从表中总是优先取出可能离最近的节点，因此其优先级为评价距离。表是一个节点的集合，用于查询所有已经访问过的节点。

初始时我们将节点插入表和表中，令。

每一次搜索中，从表中取出评价距离最小的节点，若则算法结束；否则将插入表中（也可称为染色，染色的、属于表中的节点都是不能再被访问的），该矩阵中的与上下左右四个数字交换位置，得到新的四个节点、、、，若他们不在表中，将其插入表和表中。

在维护中节点的优先级时需要使用，因为。



当搜索到表中没有节点可以访问时，则说明节点永远无法到达节点，两个矩阵状态无法转换。更复杂一些的情况，在可以到达的基础上，需要求出从到的路径，这时我们可以把表改为哈希表，用来存储节节点及其父节点，最后从节点反向，通过查找表就可以找到一条反向的路径。

本问题中A\*搜索的时间复杂度为。

八数码问题：

<http://www.d.umn.edu/~jrichar4/8puz.html>

<https://www.cs.princeton.edu/courses/archive/fall12/cos226/assignments/8puzzle.html>