

**《人工智能导论》大作业**



**题 目：** 基于迭代加深的A\*算法求解八数码问题

# 基于迭代加深的A\*算法求解八数码问题

## 研究背景与目的

##### 背景

八数码问题是人工智能领域中一个经典的搜索问题，也是很好的算法教学案例。这个问题涉及到一个3x3的棋盘，上面摆放着编号为1到8的数字方块以及一个空白格子，通过移动方块，将它们按照特定的顺序排列，达到目标状态。这个问题在计算机科学和人工智能课程中被广泛应用，可以用来教授搜索算法、启发式算法等。

A\*算法是一种经典的启发式搜索算法，结合了广度优先搜索和启发式搜索的优点，能够高效地找到最优解。而迭代加深搜索则是一种深度优先搜索的变种，通过逐步增加搜索的深度来逼近最优解，能够在有限的内存空间内求解大规模的问题。

这个课题的背景可以从学术研究和实际应用两个方面来说明，即八数码问题作为经典问题在人工智能领域的研究意义，以及基于A\*算法和迭代加深搜索求解八数码问题的算法设计与实现。

##### 目的

掌握A\*算法和迭代加深搜索算法的原理和实现方法，加深对这两种经典算法的理解。

将所学的算法知识应用于解决一个经典的人工智能问题，即八数码问题，提高问题求解能力。

完成这个大作业，进一步培养问题分析、算法设计和编程实现的能力，为以后在人工智能领域的学习和研究打下基础。

最终目的是设计并实现一个高效的算法，能够解决八数码问题，并对比不同算法在解决这一问题上的效率和性能，为人工智能领域的算法研究和应用提供参考。

## 系统总体设计

系统设计主要包括状态表示、评估函数、剪枝策略、迭代加深A算法和交互设计。首先，状态表示使用一个结构体来存储八数码问题的状态，其中包含了一个一维数组表示当前状态的数字排列，以及深度和父状态指针等信息。评估函数采用曼哈顿距离作为启发式方法，估算当前状态到达目标状态所需的最小步数。剪枝策略包括最优性剪枝和记忆化剪枝，前者通过评估函数提前判断搜索深度是否超出限制，后者利用哈希映射记录已搜索过的状态以避免重复搜索。迭代加深A算法结合A\*搜索和迭代加深搜索的特点，通过逐步增加搜索深度并利用启发式搜索策略来引导搜索。最后，交互设计使用户可以输入初始状态并查看搜索结果，提高了程序的友好性和可用性。综合这些设计，构建了一个高效的八数码问题求解程序。

## 系统详细设计

##### 搜索空间定义

状态空间：状态空间表示所有可能的八数码局面。在八数码问题中，每个状态可以表示为一个3x3的方格，其中包含数字1到8以及一个空格，空格用0表示。

初始状态：初始状态是问题的起始局面，即输入的初始八数码排列。

目标状态：目标状态是问题的目标局面，即目标八数码排列，在这里默认为：123456780。

移动规则：将空格与相邻数字进行交换，可以上下左右四个方向移动。

##### 状态表示

八数码问题的状态表示通常使用一个3x3的矩阵来表示。每个格子中包含一个数字，数字1到8代表相应的方块，而数字0代表空格。

##### 启发式函数选择

在此，启发式函数是当前深度加上到达目标状态所需的最小次数。这个启发式函数的设计思想是综合考虑了以下两个因素：

当前深度：表示从初始状态到当前状态的路径长度。深度优先搜索（DFS）算法是一个盲目搜索算法，它会优先探索深度较深的分支。因此，通过将当前深度纳入启发式函数，可以让搜索算法更倾向于探索较浅的分支，以加速搜索过程。

到达目标状态所需的最小次数：这是一个估计值，表示从当前状态到达目标状态所需的最小移动次数。在这里选用曼哈顿距离来求取最小到达次数。曼哈顿距离是八数码问题中常用的一种启发式函数，它衡量了当前状态与目标状态之间的“距离”。在八数码问题中，曼哈顿距离的计算方法是将每个数字移动到其在目标状态中应该所在的位置所需的步数的总和。这个距离是通过计算每个数字在当前状态与目标状态中的行列偏移量之和得到的。利用曼哈顿距离作为到达目标状态所需的最小次数的估计值，我们可以更加合理地评估当前状态与目标状态之间的距离。这个估计值可以帮助搜索算法更加智能地选择下一个状态，使其更有可能朝着解决方案的方向前进。

将这两个因素结合起来，就构成了这里的启发式函数。通过考虑当前深度和到达目标状态的最小次数，我们可以更加智能地指导搜索算法，使其更有效率地探索状态空间，并找到解决方案。

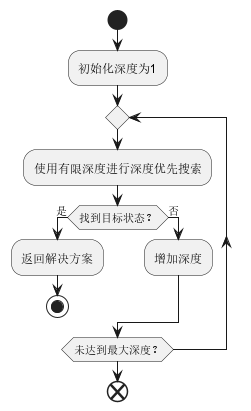
##### 迭代加深搜索策略

迭代加深搜索（IDDFS）是一种深度优先搜索算法，其核心思想在于逐步增加搜索深度，直到找到目标状态或达到最大深度。

迭代加深搜索的基本思路为：首先，将搜索深度设置为1，并执行深度优先搜索。这意味着首先搜索树的第一层，然后逐渐向下探索。在每一次搜索中，如果找到了目标状态，则算法结束，返回解决方案。这个目标状态可以是问题的目标状态，或者是达到了指定的搜索深度。如果在当前深度内没有找到目标状态，那么增加搜索深度，并重新执行深度优先搜索。这样，算法会逐步向下探索更深层次的状态。

IDDFS 的优点在于它不需要事先知道目标状态的深度，因此可以应用于各种搜索问题，并且保证找到的解是最优的。但是，IDDFS 的缺点在于可能会重复探索相同的状态，这会导致一些情况下的效率较低。在八数码问题中，IDDFS 可以有效地搜索状态空间，并找到最优解决方案，使其成为一种常用的搜索算法。

##### 算法流程图



##### 输入输出接口设计

输入接口为读入9个数字，之后根据输入内容判断是否合法。

输出接口为按照搜索算法依次给出每个状态。

输入和输出接口通过简易的交互界面进行连接。

## 编码实现过程

采用cmake构建项目，进行分文件编程。在编程实现过程中，根据项目要求共划分了四个文件，分别是：A\*算法核心、康托展开、输入输出简易交互界面和测试。



##### 康托展开

康托展开用于将一个排列转换成一个唯一的整数。在八数码问题中，康托展开可以将一个状态（即一个排列）映射成一个整数，从而方便进行状态的存储、比较和查找。

对于不同的中间状态而言，如果当前状态在前面的状态中存在，那么就不用再用当前状态进行搜索了，因为当前状态到达目标状态所用的迭代深度一定不如之前已经存在的相同状态到目标状态所用的迭代深度。

因此，我们可以用哈希映射记录中间状态。但是数组的哈希映射所用空间较大，发现对于八数码而言只有0到8这九个数字，可以用字符串进行映射。本次作业用康托展开进行哈希映射。

可以用康托展开对哈希映射进行进一步优化。

康托展开的公式为：

核心代码实现为：

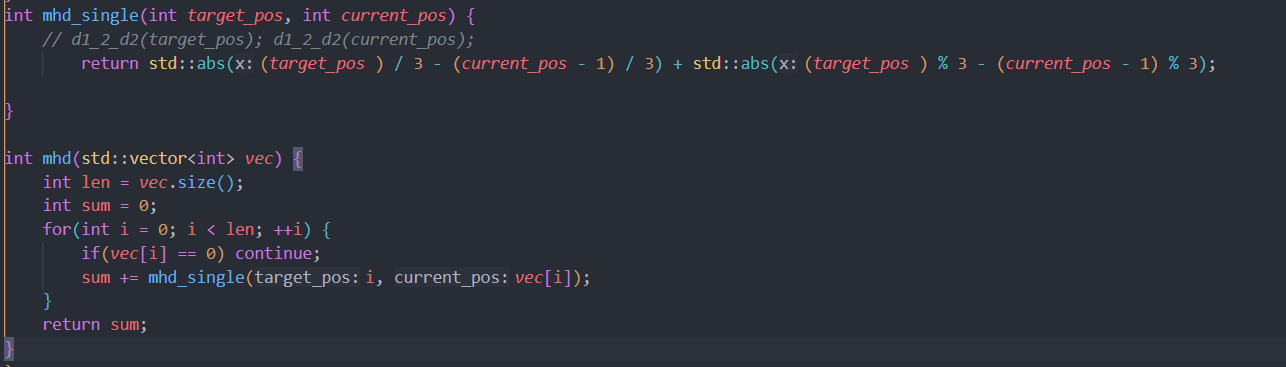


由于每一个位置的状态只有0到8共九个状态，因此直接将0到8的阶乘预处理处理，用数组fact进行保存。

##### A\*搜索算法核心

将矩阵用一维数组进行表示，在一维数组和矩阵之间用变换规则进行等价。

评估函数中的曼哈顿距离的代码实现：



在搜索算法中，进行剪枝优化。

最优性剪枝：评估函数剪枝。对于当前迭代加深的最大迭代加深来说，如果当前深度加上曼哈顿距离已经大于最大迭代深度，那么到达目标状态的迭代深度不可能小于最大迭代深度，因此直接返回即可。

窗体顶端

if(cur\_depth + mhd > max\_depth) return false;

窗体底端

记忆化剪枝：康托展开优化的哈希映射。

if(vis.find(cantor) != vis.end()) {

// 状态已存在映射关系

std::swap(arr[zero\_pos], arr[new\_pos]);

continue;

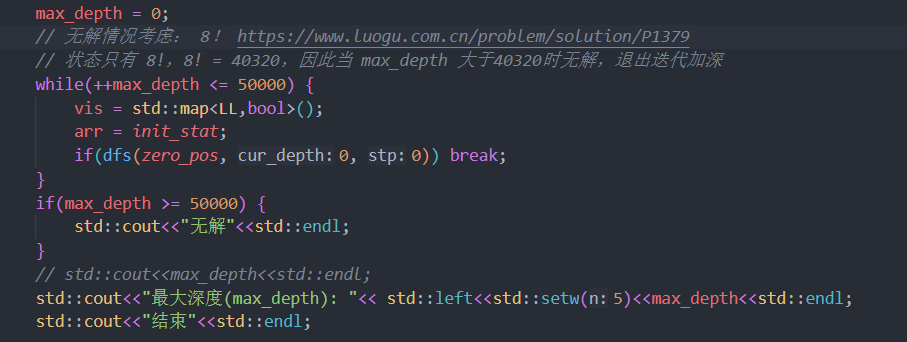
}

// 记录状态的映射

vis[cantor] = true;

八数码中，一共用8!个状态表示，即40320个状态空间。因此，当最大深度大于该值时表示无解，可以结束程序运行。

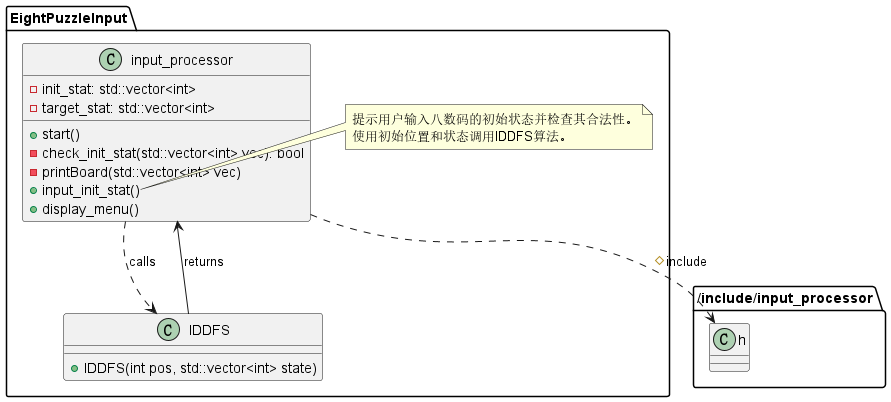
迭代加深核心部分的代码如下：

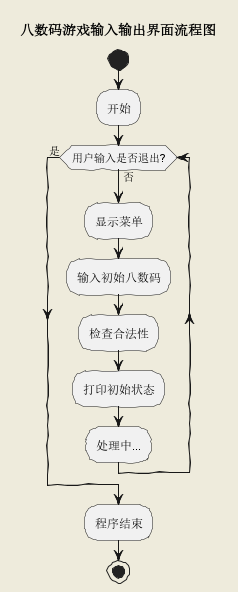


##### 输入输出简易交互界面

这个模块的主要目的是获取用户输入的初始状态，并开始调用迭代加深A\*算法处理八数码。

首先，用户可以在控制台中选择开始游戏或退出，程序会不断循环直到用户选择退出。其次，程序提供了功能来输入八数码的初始状态，并检查其合法性，确保每个数字都出现了一次。如果初始状态合法，程序会打印出初始状态，并调用 IDDFS 算法处理。同时，程序还提供了功能来打印当前八数码的状态，以便能够清晰地了解八数码的状态。

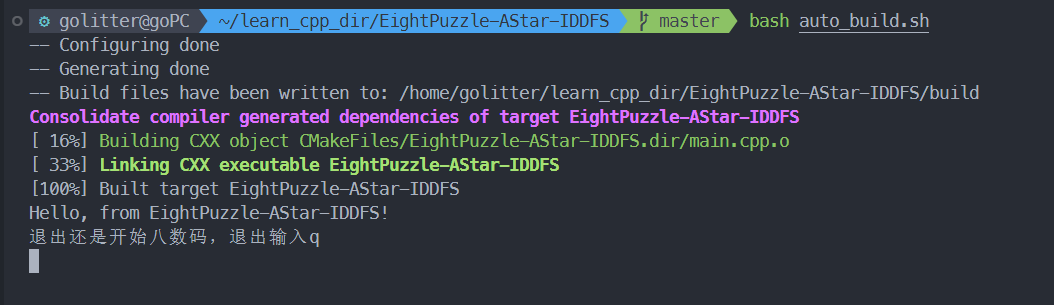




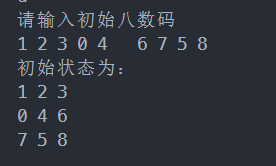
## 结果展示

在终端输入“bash auto\_build.sh”命令构建该项目。

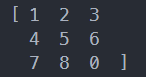
可以看到进入八数码的交互页面：



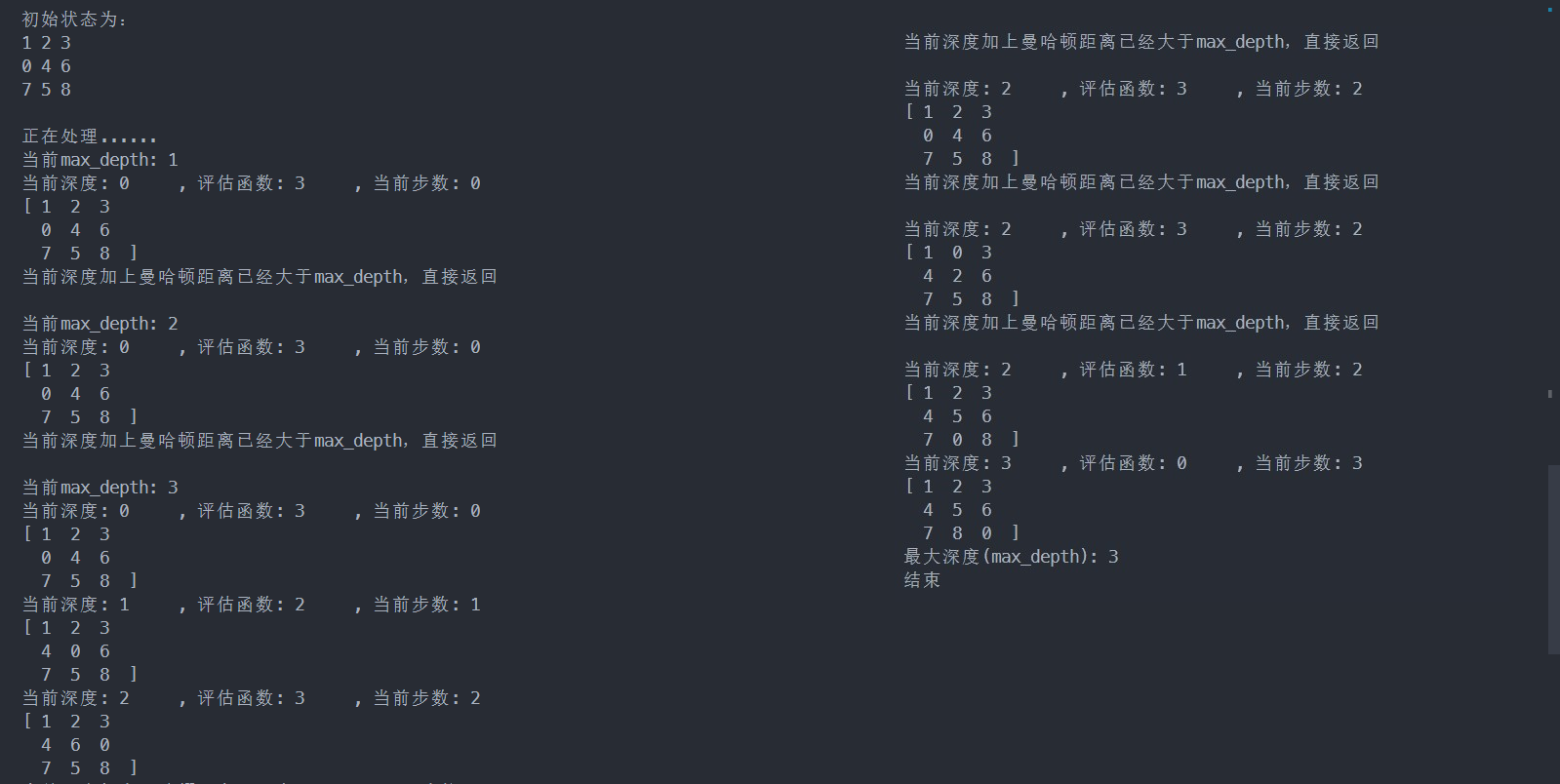
输入初始状态：



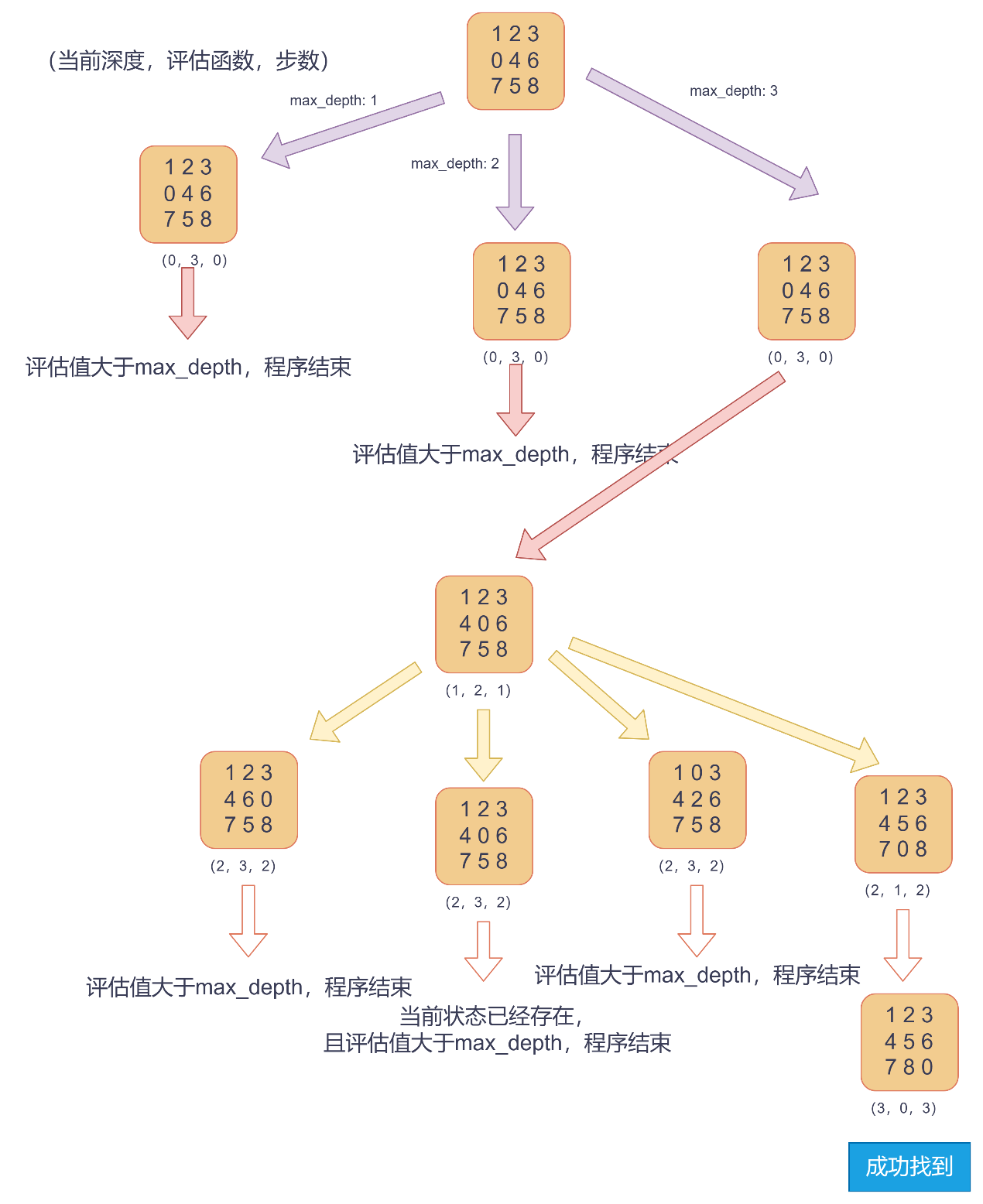
目标状态（默认）：



程序运行截图：



搜索树：



## 待改进

对于最大迭代深度的处理，我是从1开始，依次进行判断的。对于每次搜索的八数码状态来说，最多五万次，加上最优性和记忆化剪枝，迭代深度要远远小于五万次。但是由于迭代加深的最大深度每次只加一，A\*搜索算法会进行很多次，时间复杂度近似于调和级数，并且由于递归次数的增加可能会导致性能下降或内存消耗增加。特别是对于无解情况，递归次数极多导致性能极差、内存消耗剧增。因此基于迭代加深的A\*算法求解八数码问题在理论上可以解决所有初始状态的求解，但是实际上效率较差。因此在每次的最大迭代深度和无解判定上还可以进行优化。

对于本次算法实现来说，评估函数中的曼哈顿距离求解依赖于默认目标状态（默认：123456780）导致如果是非默认目标状态需要重新设计评估函数中曼哈顿距离的求解。

## 总结体会与课程建议

我通过实践加深了对 CMake 构建管理项目的理解和应用，同时巩固了 C++ 多文件模块化编程的技能。这些技能对于软件开发是至关重要的，能够帮助我更有效地组织和管理项目，提高代码的可维护性和扩展性。通过实际操作，我发现了 CMake 在项目管理中的强大之处，它可以灵活地管理各种依赖关系，并且能够适应不同平台和环境的需求。

在课程中，我了解了人工智能的基本概念、算法原理和应用场景。在未来，我打算继续深入学习机器学习、深度学习等领域的知识，并尝试将其应用到实际项目中。我认为，通过不断地学习和实践，我能够在人工智能领域取得更多的成就和突破。

实现迭代加深的A\*算法求解八数码问题是一项有挑战性的任务。我通过实现算法，进一步加深了对A\*算法的理解，学会了如何定义和使用启发式函数。 总之，通过实现迭代加深的 A\* 算法求解八数码问题，我不仅学会了一种重要的搜索算法和问题求解技术，还培养了解决复杂问题的能力和思维方式，这些都是在计算机科学和人工智能领域非常有价值的技能和经验。

对于未来的课程建议，我希望能够有更多的实践机会，例如项目实践或者案例分析，这样能够帮助我更好地将理论知识转化为实际能力。另外，希望课程内容能够更贴近实际应用，包括最新的研究进展和技术趋势，这样能够更好地满足我对知识的追求和实践的需求。

总的来说，我对这段学习经历感到非常满意和充实，我相信这些所学所得将会对我的未来发展产生积极的影响。