A*算法求解 8 数码问题

2050289 朱昀玮，2050283 陆明奇，1951112 林日中

摘 要

八数码问题也称为九宫问题。在 3×3 的棋盘，摆有八个棋子，每个棋子上标有 1 至 8 的某一数字，不同棋子上标的数字不相同。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。要求解决的问题是：给出一个初始状态和一个目标状态，找出一种从初始转变成目标状态的移动棋子步数最少的移动步骤。本实验基于 Python 语言，运用 A*算法对八数码问题进行目标结点搜索，通过 Python 中的 Tkinter 库实现界面的可视化及搜索树的绘制，并选择多种不同的启发函数，对其搜索效率进行了多方位的比较。

关键词：A*算法，启发函数，八数码，Tkinter

装 订 线

A* algorithm to solve 8 digital problems

ABSTRACT

The eight digital problem is also known as the nine house problem. On a 3 by 3 board, the pendulum has eight pieces. Each piece is marked with a number from 1 to 8. There is also a space on the board, and adjacent pieces can be moved to the space. The problem is to give an initial state and a target state, and to find a moving step with the least number of moves from the initial state to the target state. Based on Python language, this experiment uses the A* algorithm to search the target node of eight-digit problem, uses Tkinter to realize the visualization of the interface and the drawing of the search tree. Also, we select a variety of different heuristic functions to make a multi-directional comparison of its search efficiency

Key words： A* algorithm, heuristic function, 8-puzzle, Tkinter

装 订 线

目录

1 实验概述.....1
1.1 实验目的.....1
1.2 实验内容.....1
2 实验方案设计.....2
2.1 总体设计思路与总体架构.....2
2.2 核心算法及基本原理.....2
3 实验过程.....5
3.1 环境说明.....5
3.2 源代码文件清单.....5
3.3 实验结果展示.....5
4 总结.....8
4.1 实验中存在的问题及解决方案.....8
4.2 心得体会.....8
4.3 后续改进方向.....8
4.4 总结.....8
5 附录.....9
5.1 成员分工与自评.....9

装

订

线

1 实验概述

1.1 实验目的

熟悉和掌握启发式搜索策略的定义、评价函数 f(n)和算法过程，并利用 A*算法求解 8 数码问题，理解求解流程和搜索顺序。使用任意语言制作相关图形化界面，可视化 A* 算法的搜索过程、搜索结果与搜索树。

1.2 实验内容

1. 采用 Python 语言实现 A*算法的求解八数码问题的程序，设计了三种不同的启发函数：第一种：计算所有棋子到其目标的 Manhattan 距离和；第二种：计算不在位的棋子数；第三种：计算所有棋子到其目标的 Euclidean 距离和。

2. 设置相同初始状态和目标状态，针对三种启发函数，求得问题的解，并比较它们对搜索算法性能的影响，包括扩展节点数、生成节点数和运行时间。得出不同启发函数 h(n)求解 8 数码问题的结果比较表，进行性能分析。

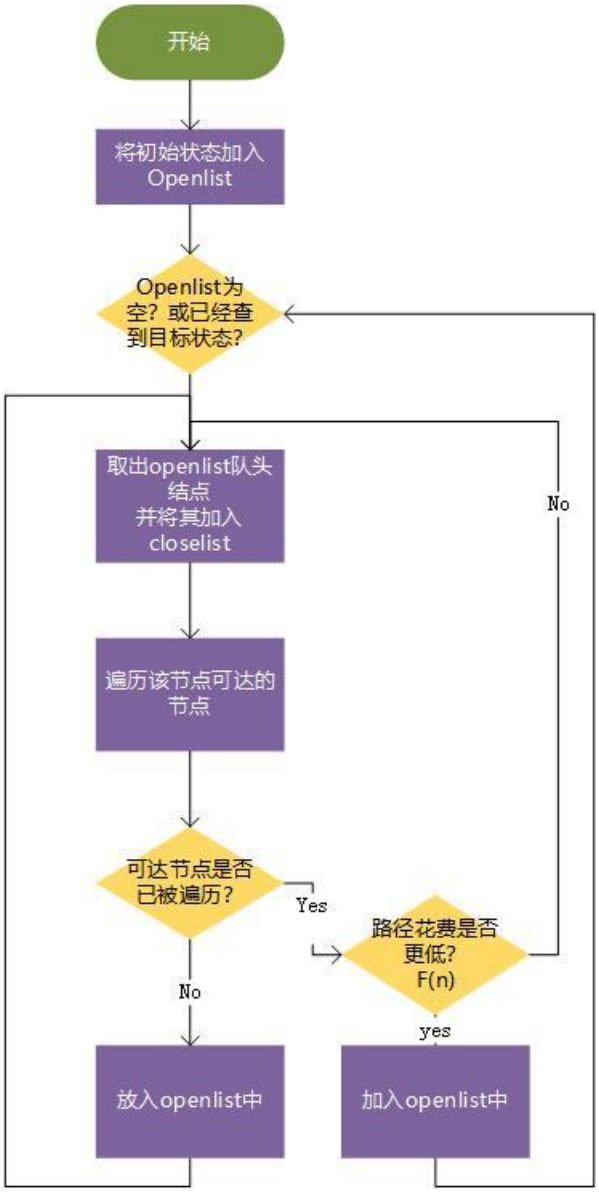
3.使用 Python 语言的 tkinter 包编写图形化界面，绘制 A* 算法的搜索树，高亮显示起点到搜索重点的路径。并在图形界面上显示搜索使用的时间和搜索结果。提供初始矩阵和结束矩阵的输入窗口，提供三种启发函数的选择窗口，基本实现用户与程序的交互。

2 实验方案设计

2.1 总体设计思路与总体架构

程序整体采用求解与图形化界面分离的策略，分别在 A_star.py 与 gui.py 中实现 A*算法求解 8 数码问题与 gui 图形化界面。

2.2 核心算法及基本原理



核心算法采用的是 A*算法。A*算法是一种有信息（启发式）的搜索策略。它对于结点的评价函数 f(n)中结合了 g(n)，即到达此结点已经花费的代价，和 h(n)，即从该结点到目标结点所花代价。有公式：f(n)=g(n)+h(n)。

A*算法既是完备的也是最优的。它每次优先扩展 f(n)值最小的结点，直到找到目标结点，或是扩展完所有可扩展的结点（无解的情况）。h(n)是一个可采纳启发式，它从不会过高地估计到达目标的代价，因为 g(n)是当前路径到达结点 n 的实际代价，所以有结论：f(n)永远不会超过经过结点 n 的解的实际代价。

A*算法需要的时间复杂度是指数级的，存储结点信息也需要比较大的空间。因此需要耗费一定的时间和空间。在本实验中，g(n)为搜索树中节点 n 的深度。对于选取不同的 h(n)，将对算法的效率有很大影响。h(n)的选取大致有如下三种情况：

（1）如果 h(n)< 到目标状态的实际距离，这种情况下，搜索的点数多，搜索范围大，效率低，但能得到最优解。

（2）如果 h(n)= 到目标状态的实际距离，即距离估计 h(n)等于最短距离，那么搜索将严格沿着最短路径进行，此时的搜索效率是最高的。

（3）如果 h(n)> 到目标状态的实际距离，搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。

具体到八数码问题的实例中，首先需要判断给定的初始八数码和目标八数码是否可解。当且仅当初始八数码与目标八数码的逆序数的奇偶性相同时，该八数码问题有解，否则无解。

对于一个八数码结点，其可扩展的方向最多只有上、下、左、右四个方向。空格的每一次移动产生的路径耗散为 1。开始时把初始的结点加入 Open 表中，后续将扩展出的结点加入 Open 表中，将扩展过的结点加入 Close 表中。每次从 Open 表中取出 f(n)值最小的结点进行扩展，直到

找到目标结点，或者 Open 表为空为止。

本实验中设计了三种不同的估价函数 h(n)：第一种：计算所有棋子到其目标的 Manhattan 距离之和；第二种：计算不在位的棋子数；第三种：计算所有棋子到其目标的 Euclidean 距离之和。

第一种：曼哈顿距离是标准的启发式函数，其值是两个点在标准坐标系上的绝对轴距总和，在多数网格地图中的启发式算法中，曼哈顿距离能表现出较好的性能，在我们的实验所用到的三种启发式函数中，曼哈顿函数表现出的性能也最优。

第二种：用不在位置的棋子个数总和作为启发式函数，h(n)<= n 到目标节点的距离实际值，搜索的点数多，搜索范围大，效率低，但能得到最优解。

第三种：如果棋子可以沿着任意角度移动（而不是网格方向），可使用直线距离作为启发式函数，即欧几里德距离。我们的实验中，由于不能沿任意距离移动，因此 h(n)小于到目标节点的距离实际值。这种方法的缺点是，计算欧几里得距离涉及到平方根运算，将耗费一些时间。

综上所述，在 A*算法的框架下，第二种评估函数可能产生多个 h(n)值相同的结点，在问题比较复杂的时候会使结点规模变得巨大；而第一种评估函数产生的 h(n)值差异性较高，在相同规模下，产生的结点数会比第一种少。因此从效率角度，采用第一种评估函数的 A*算法的效率更高，速度更快。

装

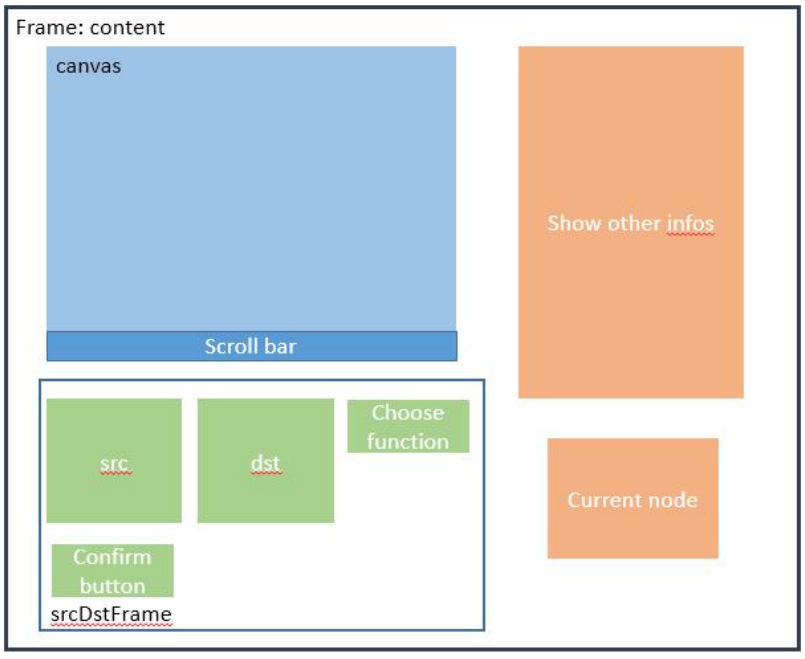
订

线

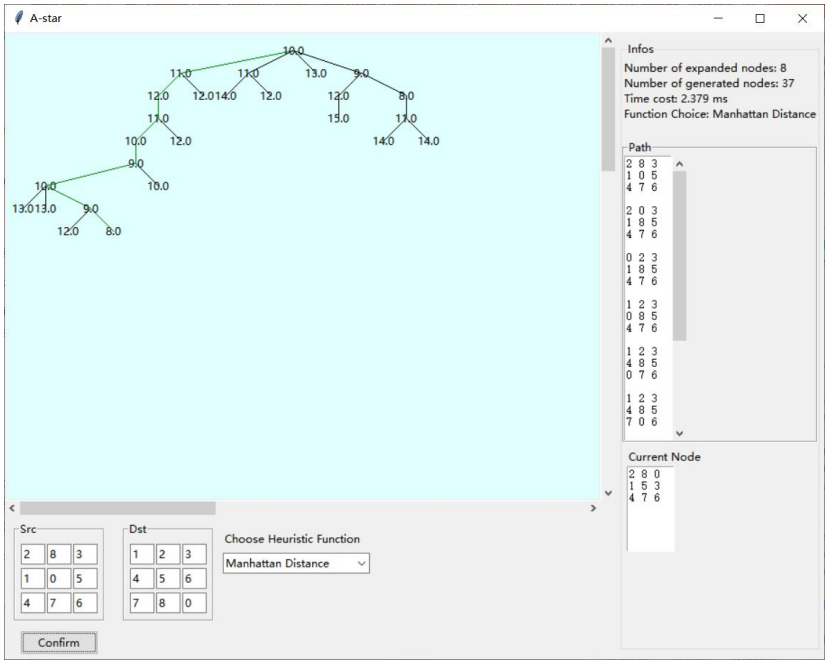
2.3 图形化界面设计

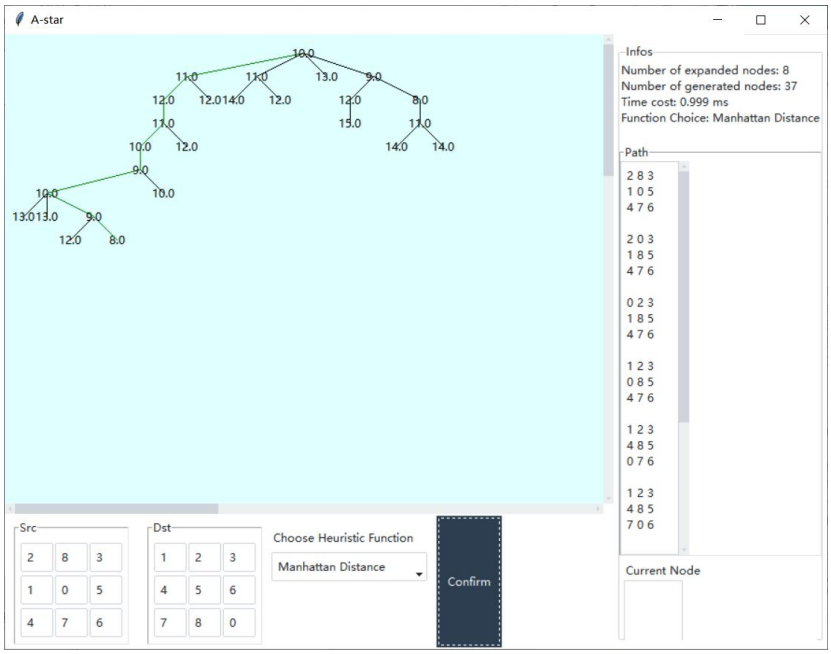
图形化界面使用 tkinter 包进行编写，使用 ttkbootstrap 进行页面美化。

图形化页面主要由 3 个部分组成，在 canvas 画布模块显示 A* 搜索算法生成的搜索树。SrcDstFrame 模块用于完成初始状态矩阵、目标状态矩阵的输入，启发函数的选择，以及启动 a*算法的按钮 button。Info 模块用于显示搜索算法得到的结果，包括程序执行时间，拓展的节点数（open，close）以及在 canvas 中选中的节点对应的矩阵信息。



最终 gui 呈现结果如下所示





在图形化界面的设计过程中，我们制作了一些更加人性化的设计，让整体的界面使用体验更加，包括窗口内各元素随着窗口变化会重新布局，让整个设计更加有响应性；为矩阵输入设计 9 个小输入框，更加符合输入逻辑；使用 combobox 预设 3 中不同的启发函数，让选择更加方便；以及能够点击绘制出的搜索树上的节点，将显示结果放在右下角 current node 方框内。

3 实验过程

3.1 环境说明

操作系统 windows10
开发语言：Python
开发环境：Python >= 3.0.0
Tk = 8.6.10
ttkbootstrap = 1.7.4

3.2 源代码文件清单

A_star.py: 用于实现 A star 算法解决 8 数码问题的算法

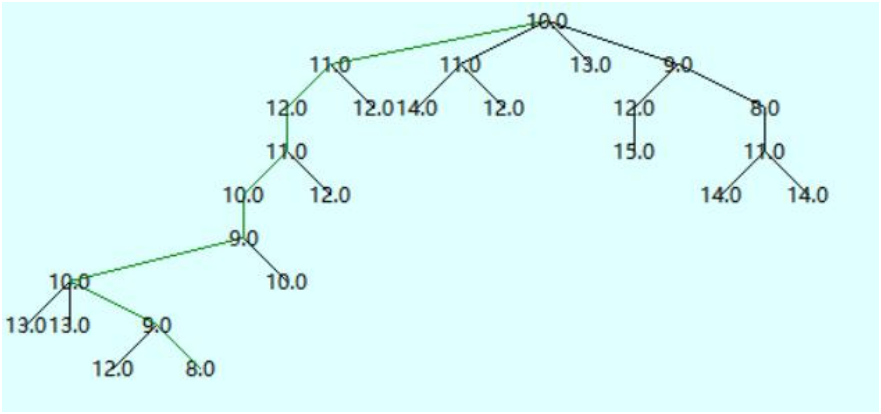
Gui.py: 用于实现图形化界面

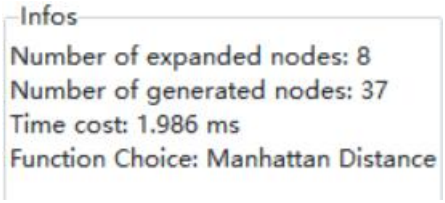
3.3 实验结果展示

设定初始状态矩阵为 2 8 3 1 0 5 4 7 6，结束状态为 1 2 3 4 5 6 7 8 0，其中 0 的位置标示不放置任何数。

3.3.1 曼哈顿距离作为启发函数

扩展节点数：8 个
生成节点数：37 个
运行时间：1.986 ms



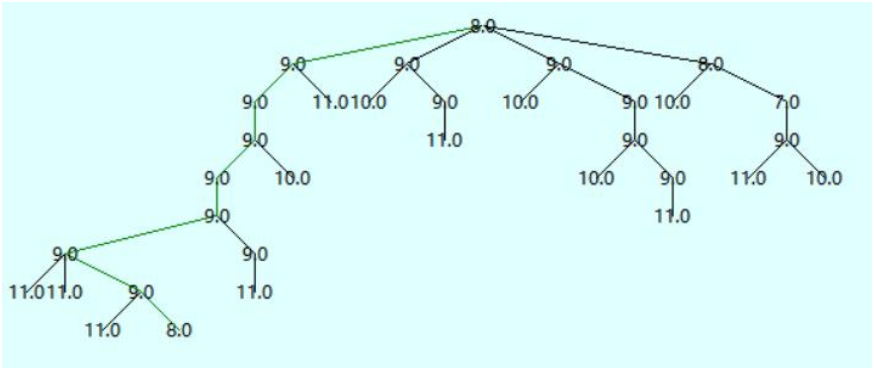


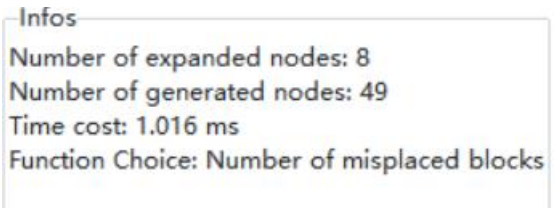
3.3.2 错误放置的位置个数作为启发函数

扩展节点数 8 个

生成节点数 49 个

运行时间 1.016ms



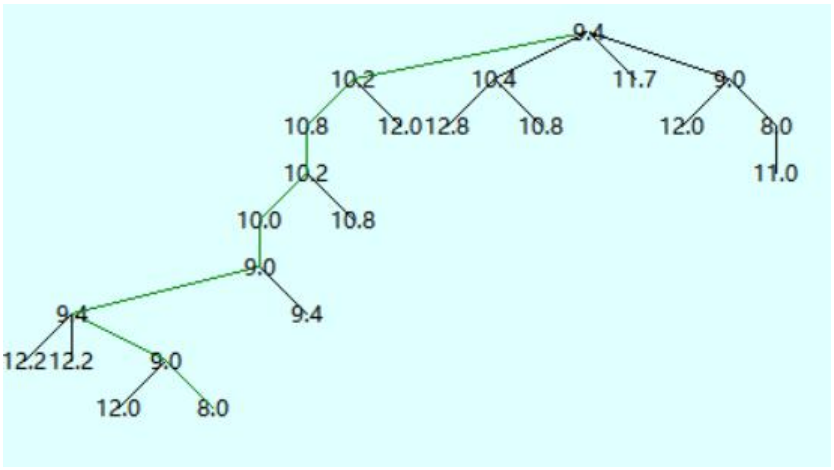


3.3.3 欧几里得距离作为启发函数

扩展节点数 8 个

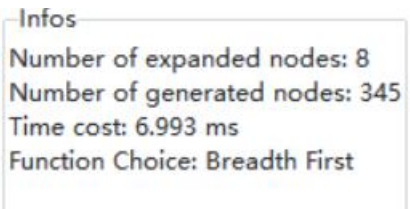
生成节点数 32 个

运行时间 1.0ms

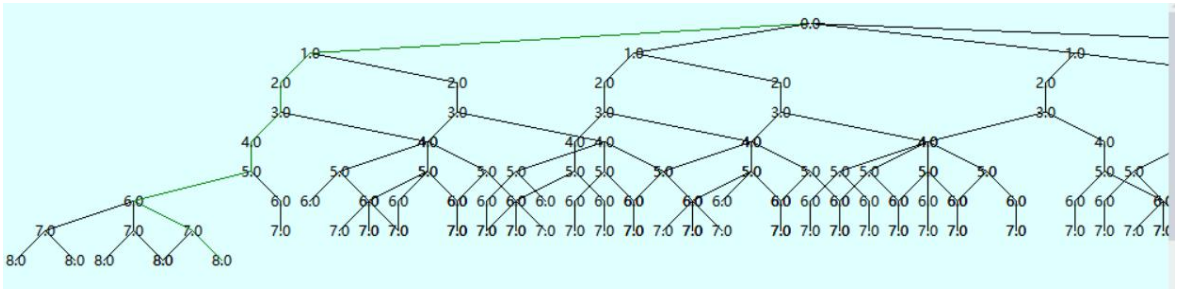


3.3.3 对照组：广度优先搜索

将启发函数设置为 0，则转变为广度优先搜索



拓展的节点数 8 个



生成的节点数 345 个

运行时间 6.993 秒

3.3.4 实验总结

对于 4 种不同的启发函数，总结实验结果如下所示

	Manhattan	Misplaced	Euclid	Breadth First
Expanded nodes	8	8	8	8
Generated nodes	37	49	32	345
Time cost	1.986	1.1016	1.0	6.993

实验结果表明，使用欧几里得距离作为启发函数时，需要生成的总结点数量最小，因此程序运行需要的时间最短。同时，三种启发函数与广度优先搜索相比，都有非常大的运行速度优势，能够较好地完成本题中指定的搜索任务。

4 总结

4.1 实验中存在的问题及解决方案

（1）最短路径（最优解）能否找到以及搜索效率关键在于启发函数的选取：启发函数值 h(n)< 到目标节点的距离实际值，这种情况下，搜索的点数多，搜索范围大，效率低。但能得到最优解。如果 h(n)=到目标节点的距离实际值，即 h(n)等于最短路径，那么搜索将严格沿着最短路径进行，此时的搜索效率是最高的。如果 h(n)> 到目标节点的距离实际值,搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。在本实验中，我们通过查阅文献资料，并自己动手将不同启发函数进行比较，得出了一个较优的启发函数：计算所有棋子到其目标的 Manhattan 距离之和。

（2）画布上绘制结点难以直接与数据结构中的搜索节点直接相连接：为了完成点击画布上对应节点，显示其对应矩阵的功能，在每次点击画布时读取鼠标指针所在位置，根据鼠标位置在搜索树中查找该位置对应的节点。考虑到本题中节点数量不多，因此直接使用深度优先搜索进行查找。

4.2 心得体会

在本次实验中，我们小组成员基于分工协作，完成了 A*算法求解 8 数码问题的程序和其相应的图形化界面显示。通过本次实验，我们动手实践 A*算法的编写，加深了对搜索算法的知识。通过自己设计和实现不同的启发函数并进行相应的结果比较，我们了解到了不同启发函数对搜索程序的运行时间，搜索树的大小，以及是否能够搜索到最优解均有影响，因此，合理选择启发函数对搜索程序十分重要。

在本次实验中，我们使用 python 编写图形化界面，第一次了解如何从用户的角度完善页面，从使用者而非是程序编写者的角度看待功能。为此，我们设计了画布、多种选择栏和输入栏，以及信息显示栏，为了达成更好的交互体验。我们还使用 ttkBootstrap 进一步提升页面美观程度，让用户体验更加。

在本次试验中，我们更是了解到沟通协作在团队工作中的重要性，通过组员的积极配合和及时沟通，我们顺利完成本次作业。

4.3 后续改进方向

（1）添加不同的启发函数，比较效果

（2）使用 python 中的双端队列，而不是列表实现 A*算法，提升程序运行速度。

4.4 总结

在本次实验中，我们小组成员基于分工协作，完成了 A*算法求解 8 数码问题的程序和其相应的图形化界面显示。通过本次实验，我们动手实践 A*算法的编写，加深了对搜索算法的知识。通过自己设计和实现不同的启发函数并进行相应的结果比较，我们了解到了不同启发函数对搜索程序的运行时间，搜索树的大小，以及是否能够搜索到最优解均有影响，因此，合理选择启发函数对搜索程序十分重要。

在本次实验中，我们使用 python 编写图形化界面，第一次了解如何从用户的角度完善页面，从使用者而非是程序编写者的角度看待功能。为此，我们设计了画布、多种选择栏和输入栏，以及信息显示栏，为了达成更好的交互体验。我们还使用 ttkBootstrap 进一步提升页面美观程度，让用户体验更加。

在本次试验中，我们更是了解到沟通协作在团队工作中的重要性，通过组员的积极配合和及

时沟通，我们顺利完成本次作业。

5 附录

5.1 成员分工与自评

朱昀玮：图形界面编写，10/10

林日中：图形界面编写，10/10

陆明奇：A*算法编写，10/10