

华东师范大学数据科学与工程学院实验报告

课程名称: AI 基础

年级: 2022 级

上机实践日期:

2024 年 4 月 14 日

指导教师: 杨彬

姓名: 朱维清

上机实践名称: 实验1

学号: 10215300402

一、实验任务

以 BFS、DFS、Dijkstra 算法、A*算法分别解决图最短路问题、八数码问题、迷宫问题与迷宫寻路可视化。

二、使用环境

图最短路问题、八数码问题——C++

迷宫问题与迷宫寻路可视化——Python

三、实验过程

1.图最短路:

- 朴素 Dijkstra: 定义二维数组 `cost` 来表示边的邻接矩阵; 定义数组 `dist` 以在循环遍历过程中实时更新源点到各点的最短距离; 定义布尔数组 `visited` 来表示各点是否收敛 (即求得最短距离)。

问题解决: 测评时有 `wrong answer`, 经过修改, 问题出在对输入有重边先后覆盖的问题 (`cost` 的输入) 和 `dist` 的更新判断条件, **已修改**。

- 堆优化 Dijkstra: 定义结构体 `edge` 表示边的终端顶点和长度, 通过 `edges` 向量记录所有边; 定义布尔数组 `visited` 来表示各点是否收敛 (即求得最短距离); 定义结构体 `node` 表示点当前离源点的最短距离和点编号, 并建立 `priority_queue` 来储存顶点, 通过 `pop` 当前最短距离的 `unvisited` 点来实现堆优化 Dijkstra。

问题解决: 测评时有 `wrong answer`, 经过修改, 问题出在 `dist` 的初始化 (修改为 `fill(dist, dist + MAX_node, INT_MAX)`) 和 `dist` 的更新判断条件, **已修改**。

2.八数码问题:

- DFS: 用字符串表示某一状态的八数码图, 定义栈来存取这些字符串, `stack` 元素为 `pair<string, int>`, 其中 `string` 为状态, `int` 为当前深度; 从栈中 `pop` 出状态移动更新下一状态 `push` 至栈中来实现 DFS。
- BFS: 区别于 DFS, 不用栈来保存当前状态的下一状态, 改用队列。由于八数码问题的性质, 用 DFS 搜寻耗时大概率比 BFS 高, 通过按深度递增进行广搜, 找到最终状态时直接访问当前深度即可。
- Dijkstra: 同图最短路堆优化 Dijkstra, 定义 `unordered_map` `distance` 和 `visited` 来表示某一状态的当前最低深度和是否收敛; 定义 `priority_queue` 来以较低复杂度实现 Dijkstra 算法。
- A-Star: 使用欧氏距离作为启发函数; 在输入起始状态后通过逆序数是否为偶数判断

是否有解；为了在找到最终状态后返回所有移动步骤，定义 `unordered_map prev` 来存取某一状态的来源步骤，其余同其特例 Dijkstra 解法。

问题解决：测评有一个样例超时（同 Dijkstra 解法），经过修改，问题出在每次 `pop` 出新状态时未通过 `visited` 来判断是否跳过，且在 `pq.push` 的判断条件中有逻辑问题，已修改。

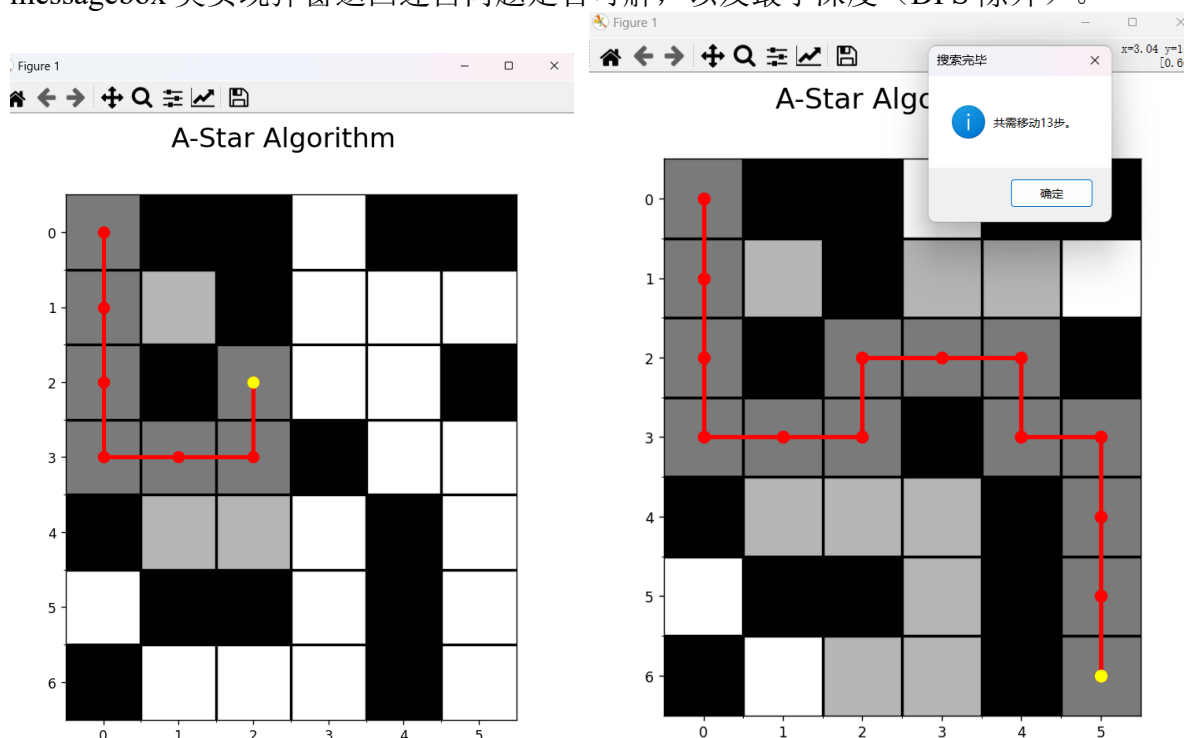
3. 迷宫问题：

- **DFS：**同八数码 DFS 解法，在 python 中用列表实现栈功能，每次四个方向移动点时判断点是否在图内，且没有撞上障碍物。
- **BFS：**同八数码 BFS 解法，在 python 中用列表实现队列功能。
- **Dijkstra：**同八数码 Dijkstra 解法，没有调用库实现堆优化，直接用列表的 `sort` 函数来实现优先队列。
- **A-Star：**同八数码 A-Star 解法，改用曼哈顿距离作为启发函数，调用 `heapq` 库函数对列表操作实现优先队列功能。

4. 迷宫寻路可视化：

根据已有代码，定义 `MazeVisualizer` 类，类函数 `visualize(self, path=[], visited=[], move=None, last_call=False, issolvable=False, steps=0)` 作为可视化接口，实现了 `visited` 的点上色、当前搜索到的点标明、以当前搜索到的点为终点的当前最短路径上色。

搜索算法循环时，在每次 `visual_move`、`visual_path`、`visual_visited` 更新时调用 `visualize()` 函数，通过 `plt.pause(0.3)` 实现实时可视化搜索过程；在搜索结束后通过 `tkinter` 库的 `messagebox` 类实现弹窗返回迷宫问题是否可解，以及最小深度（DFS 除外）。



四、总结

进一步熟悉了各搜索算法的原理与具体实现。