

如何运行

1. **环境准备:** 确保你的 Python 环境安装了必要的库:

```
pip install matplotlib numpy
```

2. **准备输入文件:** 创建一个文本文件 (例如 `maze_input.txt`), 按照以下格式描述包含特殊元素的迷宫:

- 第一行包含两个整数 N 和 M , 分别代表迷宫的行数和列数。
- 接下来 N 行, 每行包含 M 个整数, 用空格分隔。这些整数代表不同的单元格类型:
 - 0: 空地 (Path)
 - 1: 墙壁 (Wall)
 - 2: 沼泽 (Swamp - 代价 * 2)
 - 3: 加速器 (Accelerator - 代价 * 0.5)
 - 4, 5, ...: 传送门 (Portal - 零代价传送, 需成对出现)

示例 `maze_input.txt` :

```
5 6
0 0 1 0 3 0
4 0 1 2 0 0
0 0 0 0 1 5
1 0 2 0 0 4
0 0 1 0 0 5
```

3. **执行脚本:** 在终端中, 使用重定向将输入文件传递给相应的 Python 脚本:

```
python maze_DFS.py < maze_input.txt      # 运行 DFS
python maze_BFS.py < maze_input.txt      # 运行 BFS
python maze_dijkstra.py < maze_input.txt # 运行 Dijkstra
python maze_A_star.py < maze_input.txt    # 运行 A*
```

或者, 你可以直接运行 `python <name.py>`, 然后在终端中粘贴迷宫数据, 最后按 `Ctrl+D` (Linux/macOS) 或 `Ctrl+Z` 后跟 `Enter` (Windows) 来结束输入。

输出说明

脚本执行后会:

1. 在终端打印信息, 通常包括:

- **路径长度 (步数):** X (表示路径包含的节点数减 1, 即移动的次数。注意: 传送门移动不计入步数)
- **实际距离 (路径总代价):** Y (考虑了斜线移动代价 ($\sqrt{2}$)、沼泽 (代价*2)、加速器 (代价*0.5) 以及零代价传送门的累积路径成本。对于 Dijkstra 和 A*, 这通常是它们找到的最优代价。)

2. 弹出一个 matplotlib 窗口, 动态展示搜索过程和最终路径:

- **背景单元格颜色:**
 - 白色: 空地 (Path)
 - 黑色: 墙壁 (Wall)
 - 棕色: 沼泽 (Swamp)
 - 浅蓝色: 加速器 (Accelerator)
 - 黄色/紫色/橙色等: 传送门 (Portals - 不同 ID 可能对应不同颜色)
- **蓝色小点 (Visited):** 代表算法在搜索过程中访问或处理过的节点, 按相应算法的顺序出现。
- **红色粗线和圆点 (Final Path):** 代表最终找到的从起点 (0, 0) 到终点 (N-1, M-1) 的路径 (如果找到)。