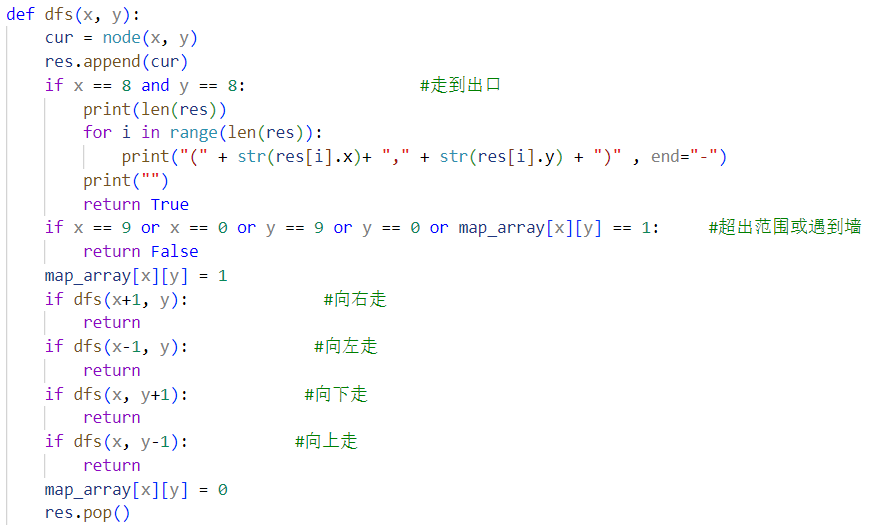
一、盲目搜索解决迷宫问题

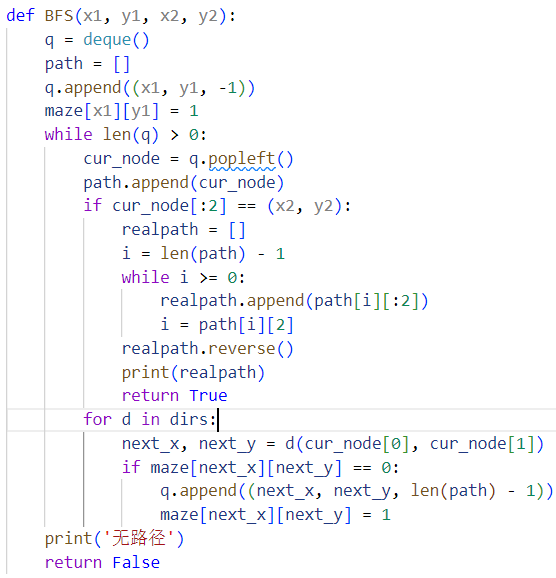
1.算法思路：DFS：先用二维列表储存地图，一维列表储存路径，每次到达一个格子后先进入路径，判断是否为墙，若不是则置为墙并依次向右左下上进行递归探索，最后恢复为通路并弹出路径列表；若为墙则返回上一级递归；当寻找到出口时则输出当前储存的路径。

BFS：使用python中的collections中的deque模拟队列操作，先对起点进行第一次操作后开始循环，当队列不为空时，将一个格子进队列后判断是否为墙，若不是则置为墙并将其周围格子不是墙的进队，当找到终点时，将已寻找的格子回溯存入路径后反转输出。当队列空时还未找到终点则无路径。

1. 代码分析



函数接收要寻找的格子位置，然后存入路径，之后判断并操作，结束后还原状态。



先对起点操作后开始循环，依次进入队列后判断操作即可。

1. 算法效率比较

两种算法在时间复杂度上相同，空间复杂度上BFS略高，在寻找最短路上BFS第一次总是能找到最短路径，DFS则需要多次遍历。

DFS多次实际运行时间：



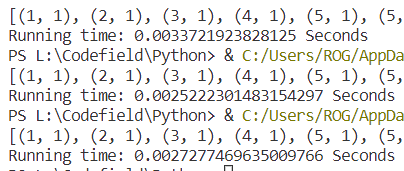


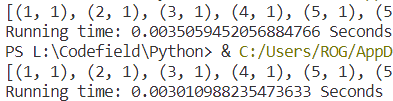






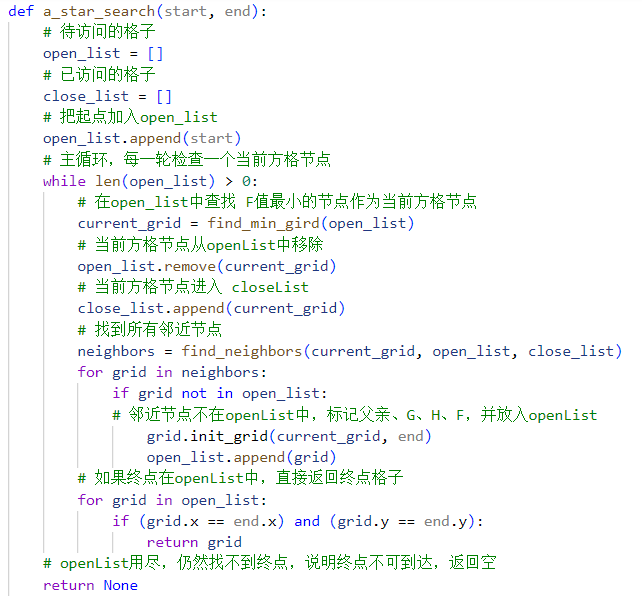
BFS多次实际运行时间：

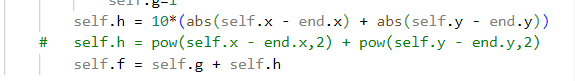




多次运行下来后BFS的平均时间略微高于DFS。

1. 启发式搜索
2. 算法思路：A\*算法，两个列表作为open表和close表，先把起点加入open表后开始循环，每次循环查找当前open表中的格子估计值最小的结点将其加入close表，并将其周围非墙且不在close中的结点加入open表；若在open表中找到终点，则直接返回终点，否则open已用尽还未找到，则无路径。估计值使用两种启发式函数。
3. 代码分析：





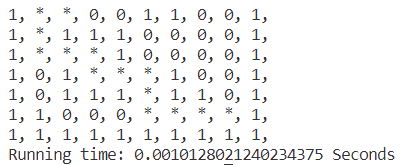
启发式函数设置。

1. 效率比较：

使用第一种启发式函数：







使用第二种启发式函数：









基本上差距不大，但若是问题规模足够大的话第二种估价方式的效率应该低于第一种。