智能计算技术

实验报告

**实验1：基于A\*算法的无人车路径规划**

**一、实验目的**

实验前要求掌握状态空间图的知识表示方法，基于状态空间图的知识搜索一般过程。掌握基于状态空间图的知识表示的要素，掌握状态空间图的知识搜索流程图，通过最短路径求解问题培养应用图搜索解决实际问题的能力。

通过实验掌握A\*智能搜素算法及其在无人车路径规划中的应用。提升使用智能搜索算法解决实际问题的能力。

**二、实验要求**

C/C++或Python或其他。

**三、实验内容**

生成一个NxN的二维网格，随机指定一些格子为障碍， 并设定左上角有辆无人车（占一个格子）要去右下角，使用A\*算法为该无人车计算起点到终点的不撞到障碍的最优路径。

**四、实验设计**

A\*算法核心函数的设计：

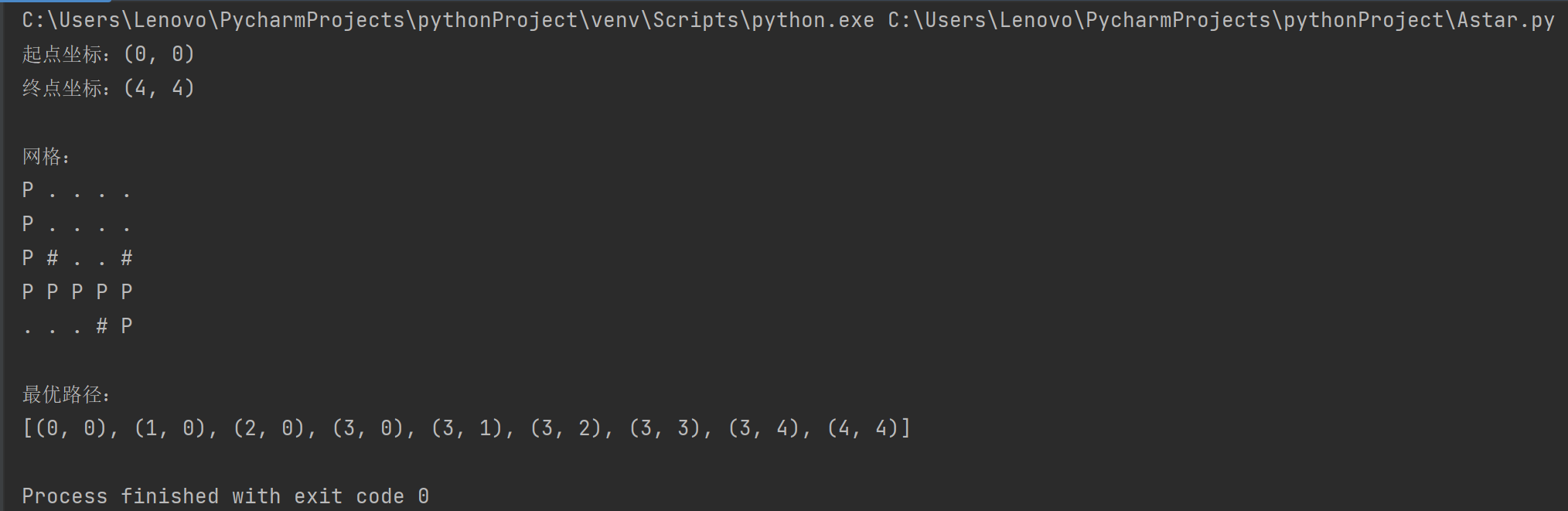
open\_list是一个存储节点对象的列表，用于存储待探索的节点。节点对象按照其 f 值（综合代价）进行排序，因此在堆（heap）中保持最小的 f 值节点位于堆顶。这样可以通过从堆顶弹出节点来选择具有最小综合代价的节点进行探索。

代码使用 heapq 模块来实现堆的操作，并通过 heappush 将节点添加到堆中，以及使用 heappop 弹出堆顶节点。此外，为了支持更新节点的代价和父节点，代码还使用了列表的索引操作来查找和更新节点。

def a\_star(grid, start, goal):  
 open\_list = []  
 heapq.heappush(open\_list, start)  
 start.g = 0  
  
 while open\_list:  
 current\_node = heapq.heappop(open\_list)  
 if current\_node == goal:  
 return get\_path(current\_node)  
  
 neighbors = get\_neighbors(grid, current\_node)  
 for neighbor in neighbors:  
 new\_g = current\_node.g + 1  
 if new\_g < neighbor.g:  
 neighbor.g = new\_g  
 neighbor.h = heuristic(neighbor, goal)  
 neighbor.f = neighbor.g + neighbor.h  
 neighbor.parent = current\_node  
 if neighbor not in open\_list:  
 heapq.heappush(open\_list, neighbor)  
 else:  
 # 如果邻居节点已经在开放列表中，更新其代价和父节点 \*\*\*  
 index = open\_list.index(neighbor)  
 open\_list[index] = neighbor  
 heapq.heapify(open\_list)  
  
 return None

**五、实验结果**

N=5



N=10

