



人工智能

本科生实验报告

(2022学年春季学期)

课程名称: Artificial Intelligence

教学班级	信计系统结构方向	专业（方向）	ICS
学号	20337268	姓名	张文沁

一、实验题目

盲目搜索

二、实验内容

[illegible]

三、实验代码

1. 类型一：

```
#第一种实现，DFS实现
direction=[(0,1),(1,0),(0,-1),(-1,0)]#表示上下左右四个方向
path=[]#找到的路径
matrix = []#输入的矩阵

def read(filePath):#读入矩阵
    with open(filePath, encoding='utf-8') as file_obj:
        for line in open(filePath,encoding='utf-8'):
            line = line.strip()
            line = list(line)
            matrix.append(line)

def mark(matrix,pos):#做标记，表示已经遍历过
    matrix[pos[0]][pos[1]]=2

def able(matrix,pos):#检查迷宫matrix的位置pos是否可通行
    return matrix[pos[0]][pos[1]]==0

def find_path(matrix,pos,end):#主函数
    mark(matrix,pos)
    if pos==end:
        print(pos,end=" ") #已到达出口，输出这个位置。成功结束
        path.append(pos)
        return True
    for i in range(4): #否则按四个方向顺序检查
        nextp=pos[0]+direction[i][0],pos[1]+direction[i][1]
        #考虑下一个可能方向
        if able(matrix,nextp):#旁边无法通过
            if find_path(matrix,nextp,end):#如果从nextp可达出口，输出这个位置，成功结束
                print(pos,end=" ")
                path.append(pos)
                return True
    return False

"""同上，展开
def func(path,row,clo):
    if matrix[row-1][clo] == 0 : #上面
        path.append((row-1,clo))
    elif matrix[row + 1][clo] == 0: #下面
        path.append((row + 1,clo))
    elif matrix[row][clo - 1] == 0: #左边
        path.append((row,clo - 1))
    elif matrix[row][clo + 1] == 0: #右边
        path.append((row,clo + 1))
    else:
        path.pop()
"""

def see_path(matrix,path):#可视化
    for i,p in enumerate(path):
        if i==0:
            matrix[p[0]][p[1]] ="E"
        elif i==len(path)-1:
            matrix[p[0]][p[1]]="S"
        else:
            matrix[p[0]][p[1]] =3
```

```

print("\n")
for r in matrix:
    for c in r:
        if c==3:#路径, 绿色
            print('\033[0;32m'+""+" "+"'\033[0m',end="")
        elif c=="S" or c=="E":#开始和结尾, 蓝色
            print('\033[0;34m'+c+" " + '\033[0m', end="")
        elif c==2:#探索过的路径, 红色
            print('\033[0;35m'+""+" "+"'\033[0m',end="")
        elif c==1:#墙
            print('\033[0;40m'+""+"*2+'\033[0m',end="")
        else:
            print(" "*2,end="")
    print()

if __name__ == '__main__':
    filePath = r'MazeData.txt'
    read(filePath)
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 'S':
                matrix[i][j] = 0
                start = (i,j)
            elif matrix[i][j] == 'E':
                matrix[i][j] = 0
                end = (i,j)
            else:
                matrix[i][j] = int(matrix[i][j])
    #print(matrix)
    #start=(1,34)
    #end=(16,1)
    find_path(matrix,start,end)
    see_path(matrix,path)

```

2. 类型二:

```

import os
import sys
start = []# 记录起点
end = []# 记录终点
matrix = [] # 储存迷宫数据
state = []# 访问状态标记 0为未访问, 1为正向队列访问, 2为逆向队列访问
action = []# 父节点生成此节点时所采取的动作
path1 = []#正向, 用1表示状态
path2 = []#反向, 用2表示
dirx = [0,0,1,-1]#参考代码, 同第一个拆开
diry = [1,-1,0,0]
def read(file_path):
    global matrix
    global state
    global action
    with open(file_path,encoding = 'utf-8') as file_obj:
        for line in file_obj:
            line = line.strip()
            matrix.append(list(line))
    state = [[0 for i in range(len(matrix[0]))] for j in range(len(matrix))]

```

```

    action = [[0,0] for i in range(len(matrix[0])) for j in
range(len(matrix))]

def __dbfs__(start,end):
    global matrix # 储存迷宫数据
    global action #父节点生成此节点时所采取的动作
    dirx = [0,0,1,-1]#参考代码，同第一个拆开
    diry = [1,-1,0,0]
    path1 = [start]
    path2 = [end]
    flag = 0

    while(len(path1) and len(path2)):
        state[start[0]][start[1]] = 1
        state[end[0]][end[1]] = 2

        if(len(path1) <= len(path2)):
            flag = 1
            temp = path1[0]
            del path1[0]
        else:
            flag = 0
            temp = path2[0]
            del path2[0]

        for i in range(4):
            dx = temp[0] + dirx[i]
            dy = temp[1] + diry[i]
            if(dx >= 0 and dx < len(matrix) and dy >= 0 and dy <
len(matrix[0]) and matrix[dx][dy]!='1'):
                if(state[dx][dy] == 0):
                    state[dx][dy] = state[temp[0]][temp[1]]
                    if(flag):
                        action[dx][dy] = [dirx[i],diry[i]] # 正向序列，
action记为正向

                        path1.append([dx,dy])
                    else:
                        action[dx][dy] = [-dirx[i],-diry[i]] # 逆向序列，
action记为逆向

                        path2.append([dx,dy])
                    else:
                        if state[temp[0]][temp[1]] + state[dx][dy] == 3: # 相
遇

                            if(flag):
                                path = []
                                dx = temp[0]
                                dy = temp[1]
                                while(action[dx][dy] != [0,0]): # 如果没有遍历
到起点（在path路径上，只有起点的action是[0,0]）
                                    path.append(action[dx][dy]) # 将父节点产生
此节点的动作记录到path中

                                dx -= action[dx][dy][0]
                                dy -= action[dx][dy][1]
                                path.reverse()

                                path2 = []
                                dx = temp[0]+dirx[i]
                                dy = temp[1]+diry[i]

```

```

        while(action[dx][dy] != [0,0]): # 如果没有遍历
到终点（在path路径上，只有终点的action是[0,0]）
            path2.append(action[dx][dy]) # 将父节点产
生此节点的动作记录到path中

            dx += action[dx][dy][0]
            dy += action[dx][dy][1]

            path.append([dirx[i],diry[i]])
            path += path2
            return path
    else:
        path = []
        dx = temp[0]
        dy = temp[1]
        while(action[dx][dy] != [0,0]): # 如果没有遍历
到起点（在path路径上，只有起点的action是[0,0]）
            path.append(action[dx][dy]) # 将父节点产生
此节点的动作记录到path中

            dx += action[dx][dy][0]
            dy += action[dx][dy][1]
            path2 = []
            dx = temp[0]+dirx[i]
            dy = temp[1]+diry[i]
            while(action[dx][dy] != [0,0]): # 如果没有遍历
到终点（在path路径上，只有终点的action是[0,0]）
                path2.append(action[dx][dy]) # 将父节点产
生此节点的动作记录到path中

                dx -= action[dx][dy][0]
                dy -= action[dx][dy][1]
            path2.reverse()
            path2.append([dirx[i],diry[i]])
            path2 += path
            return path2

def search_path():
    global start
    global end
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 'S':
                start = [i,j]
                state[i][j] = 1
            if matrix[i][j] == 'E':
                end = [i,j]
    path = __dbfs__(start,end)
    return path

#打印出路径,可视化，同第一个
def see_path(matrix,path):#可视化
    global start
    dx = start[0]
    dy = start[1]

    while(len(path)):
        dx += path[0][0]
        dy += path[0][1]
        matrix[dx][dy] = '2'
        del path[0]

```

```

for i,p in enumerate(path):
    if i==0:
        matrix[p[0]][p[1]] ="E"
    elif i==len(path)-1:
        matrix[p[0]][p[1]]="S"
    else:
        matrix[p[0]][p[1]] =3
print("\n")
for r in matrix:
    for c in r:
        if c=='2':#路径, 绿色
            print('\033[0;32m'+"*"+ " "+'\033[0m',end="")
        elif c=="S" or c=="E":#开始和结尾, 蓝色
            print('\033[0;34m'+c+" " + '\033[0m', end="")
        elif c=='1':#墙
            print('\033[0;40m'+ " "*2+'\033[0m',end="")
        else:
            print(" "*2,end="")
    print()

if __name__ == '__main__':
    filePath =r'MazeData.txt'
    read(filePath)
    path = search_path()
    print(path)
    see_path(matrix,path)

```

四、实验结果及分析

1. 实验结果展示示例

The screenshot shows a Python IDE with a terminal window at the top and a maze visualization at the bottom. The terminal output lists the coordinates of the path found by the algorithm, starting from (16, 1) and ending at (1, 34). The maze visualization below shows a grid with black walls, a green path of asterisks, and 'E' for entrance and 'S' for exit.

2. 实现2结果展示示例:

```
PS F:\CodeFile for Python> & 'D:\python\Python\python.exe' 'c:\Users\张文沁\.vscode\extensions\ms-python.python-2022.2.1924887327\pythonFiles\lib\python\debugpy\launcher' '59369' '--' 'f:\CodeFile for Python\code_f11.py'
```



```
PS F:\CodeFile for Python>
```

五、 参考资料

参见可视化教程实现简单可视化

https://blog.csdn.net/weixin_43501684/article/details/90147421