**大数据算法分析与设计实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学号 | 1910716442 | 姓名 | 周俊磊 | 班级 | 19大数据2班 | |
| 上课地点 | 精工1-314 | 教师 | 肖晶晶 | 上课时间 | 2021-11-25 | |
| **实验七 分支限界法**  1. 实验目的  1.1．掌握分支限界法的基本思路。  1.2．熟悉用分支限界法策略解决图问题（如TSP问题）和组合问题。  2. 实验环境  2.1 pycharm  2.2 Window XP  3. 实验内容  布线问题   * **问题描述：**   印刷电路板将布线区域划分成n\*m个方格阵列，精确的电路布线问题要求确定连接方格a的中点到方格b的中点的最短布线问题。在布线时，电路只能沿着直线或直角布线。**为了避免线路相交，已布了线的方格做了封锁标记，其他线路不允许穿过被封锁的方格**。  IMG_256   * **编程任务：**   给定一个mxn的矩阵，给出器矩阵的方格阵列，求解最短路径和每一步路径的方向。  **数据输入：**  输入共有n+3行  第一行包括两个数，n，m表示电路板是一个n\*m的方格阵列  第二行包括两个数，row1，col1，表示起始点的行号和列号  第三行包括两个数，row2，col2，表示目标点的行号和列号  然后输入n行，每行m个数，0或1，表示电路板的封锁情况  数据规模：  0<=m<=100  0<=n<=100   * **结果输出:**   输出两行：  第一行表示从起始点到目的点的最短距离，第二行表示每一步路径的方向，其中向右为（1，0） ；向下为（0，1），向左为（-1，0），向上为（0，-1） | | | | | | |
| 4. 教师批改意见  签字：  日期： | | | | | | 成绩 |

**实验报告细表**

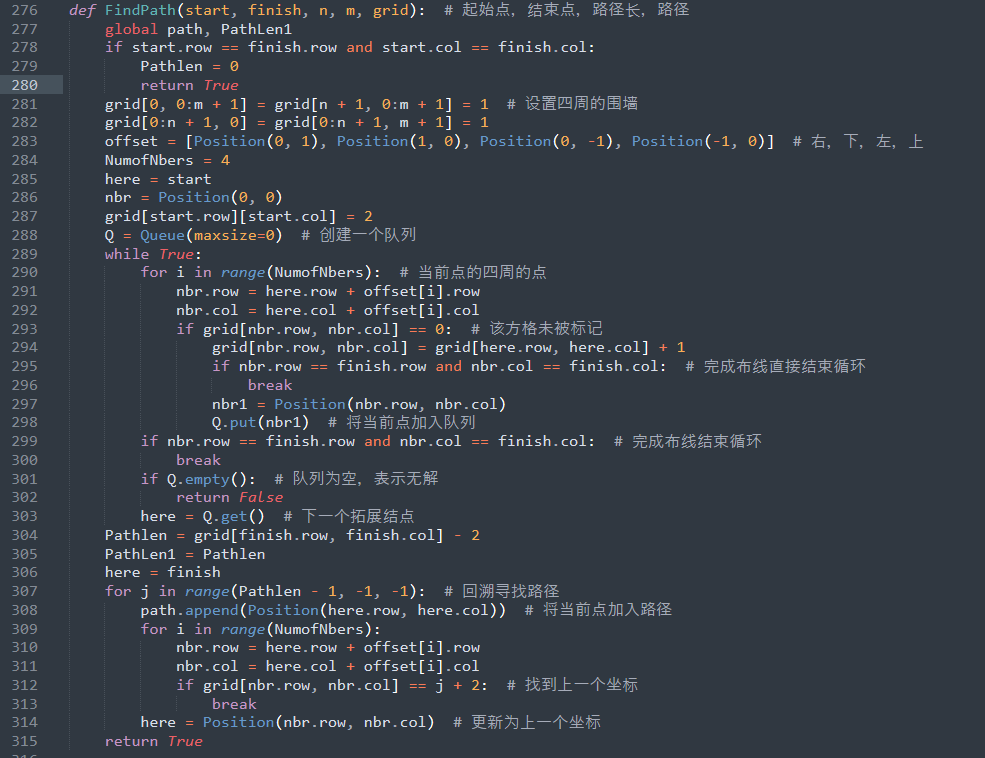
**1布线问题**

**1.1 算法设计思想**

利用分支限界法，首先将起始点作为当前活结点，寻找活结点四周可以布线的点加入队列作为下一个拓展结点。后弹出队列第一个元素作为当前活结点继续寻找拓展结点，循环直至找到终点或者队列为空（表示无解）

**1.2 程序源码**

主函数源代码：



程序源码：

import numpy as np

from queue import Queue

global path, PathLen1

PathLen1 = 0 # 用于保存路线长度

path = [] # 保存路线

### 声明position类存放当前点的横纵坐标

class Position(object):

def \_\_init\_\_(self, row, col):

self.row = row

self.col = col

def FindPath(start, finish, n, m, grid): # 起始点，结束点，路径长，路径

global path, PathLen1

if start.row == finish.row and start.col == finish.col:

Pathlen = 0

return True

grid[0, 0:m + 1] = grid[n + 1, 0:m + 1] = 1 # 设置四周的围墙

grid[0:n + 1, 0] = grid[0:n + 1, m + 1] = 1

offset = [Position(0, 1), Position(1, 0), Position(0, -1), Position(-1, 0)] # 右，下，左，上

NumofNbers = 4

here = start

nbr = Position(0, 0)

grid[start.row][start.col] = 2

Q = Queue(maxsize=0) # 创建一个队列

while True:

for i in range(NumofNbers): # 当前点的四周的点

nbr.row = here.row + offset[i].row

nbr.col = here.col + offset[i].col

if grid[nbr.row, nbr.col] == 0: # 该方格未被标记

grid[nbr.row, nbr.col] = grid[here.row, here.col] + 1

if nbr.row == finish.row and nbr.col == finish.col: # 完成布线直接结束循环

break

nbr1 = Position(nbr.row, nbr.col)

Q.put(nbr1) # 将当前点加入队列

if nbr.row == finish.row and nbr.col == finish.col: # 完成布线结束循环

break

if Q.empty(): # 队列为空，表示无解

return False

here = Q.get() # 下一个拓展结点

Pathlen = grid[finish.row, finish.col] - 2

PathLen1 = Pathlen

here = finish

for j in range(Pathlen - 1, -1, -1): # 回溯寻找路径

path.append(Position(here.row, here.col)) # 将当前点加入路径

for i in range(NumofNbers):

nbr.row = here.row + offset[i].row

nbr.col = here.col + offset[i].col

if grid[nbr.row, nbr.col] == j + 2: # 找到上一个坐标

break

here = Position(nbr.row, nbr.col) # 更新为上一个坐标

return True

def main():

global path, PathLen1

n, m = 7, 7 # m,n表示方阵的行数和列数

grid = np.zeros((n + 2, m + 2), dtype=int) # 初始化方阵

# 输入方阵的值

for i in range(1, n + 1):

grid[i, 1:n + 1] = np.array(input().split(), dtype=int)

start\_x, start\_y = 3, 2 # 起点的坐标

finish\_x, finish\_y = 4, 6 # 终点的坐标

start = Position(start\_x, start\_y)

finish = Position(finish\_x, finish\_y)

if FindPath(start, finish, n, m, grid):

print("Path length: ", PathLen1)

path.append(start)

path = np.flipud(path)

print("The way to move is as follows:")

for i in range(len(path) - 1):

print("(", path[i+1].col - path[i].col, ",", path[i+1].row - path[i].row, ")", end=" ")

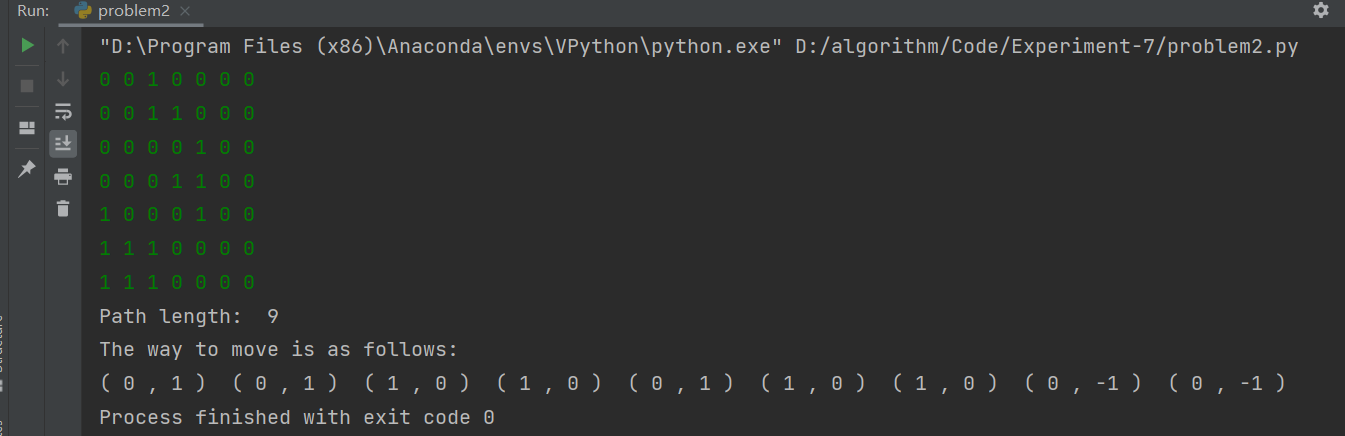
else:

print("No path!")

if \_\_name\_\_ == '\_\_main\_\_':

main()

**1.3 实验结论（结果验证）**



**1.4 心得体会**

通过此次实验，我进一步加深了对分支限界法的理解，并尝试使用此方法解决问题。