

**实验报告**

**课程名称： 算法设计与分析**

**学 院： 应用技术学院**

**专业班级： 18计科一班**

**学 号： 201833050027**

**姓 名： 叶成宇**

1．实验目的



2．实验内容

import numpy as np

r = [

[13, 15, 24, 44, 29, 50],

[16, 18, 8, 21, 53],

[7, 26, 4, 38],

[12, 1, 29],

[9, 4],

[11]

]

def least\_cost(r):

# 一共points个点

points = len(r) + 1

# 初始化记录代价和停靠点矩阵

m = np.zeros((points, points), dtype=np.int16)

s = np.zeros((points, points), dtype=np.int16)

# 为m矩阵赋初值等于r,row = 行 col = 列

for row in range(len(r)):

for col in range(len(r[row])):

m[row][row+col+1] = r[row][col]

print('初始矩阵为:\n{}'.format(m))

# 遍历不同间隔

for interval in range(2, points):

# 遍历不同起点

for start in range(points - interval):

end = start + interval

# 遍历两点间不同中间节点

for k in range(start, end):

if m[start][k] + m[k][end] < m[start][end]:

m[start][end] = m[start][k] + m[k][end]

s[start][end] = k

# 显示途径停靠节点的结果

show\_stop\_point(s, 0, len(r))

def show\_stop\_point(s, start, destination):

stop\_point = s[start][destination]

if stop\_point != 0:

# 递归查看停靠的节点

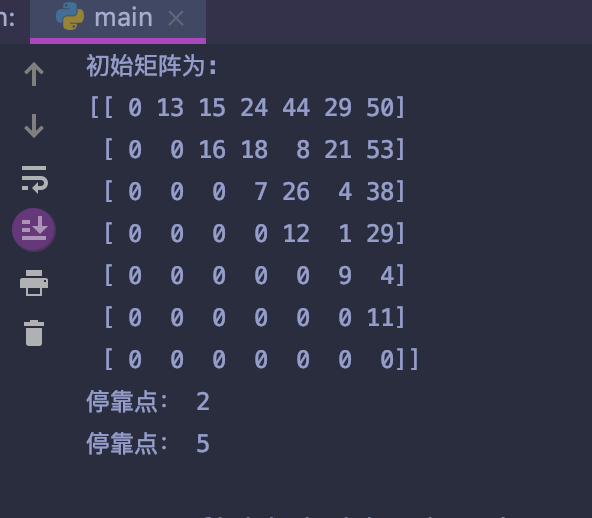
show\_stop\_point(s, start, stop\_point)

print('停靠点：', stop\_point+1)

show\_stop\_point(s, stop\_point, destination)

least\_cost(r)

3．实验步骤和实验结果



4．分析与讨论

**算法思路：**

初始化m [ i ] [ j ] = r [ i ] [ j ] m[i][j]=r[i][j]m[i][j]=r[i][j]储存各子问题最低代价，初始化s [ i ] [ j ] = 0 s[i][j]=0s[i][j]=0存储各个子问题最低代价的停靠点。

interval表示跨度为interval的两个站点。逐渐增大interval，依次算它们的最小花费，直到最后的终点。

因为相邻两点的代价已经知道，**interval=1**不需要考虑。

**interval=2**

中间停靠点只有一个，依次遍历求得1到3，2到4，3到5，4到6的最低代价。

**interval=3**

依次遍历求得1到4，2到5，3到6的最低代价。中间停靠点有2个，需要依次考虑。例如：1到4，最优的停靠点可能是2或3。

**interval=4**

依次遍历求得1到5，2到6。中间停靠点有3个，需要依次考虑。因为间隔为2两点的最优解已经算过，直接查表即可。

**interval=5**

求得1到6的代价，中间停靠点可能有4个，依次遍历，最优值即为最优解。

根据s [ i ] [ j ] s[i][j]s[i][j]可得到需要停靠哪些点。