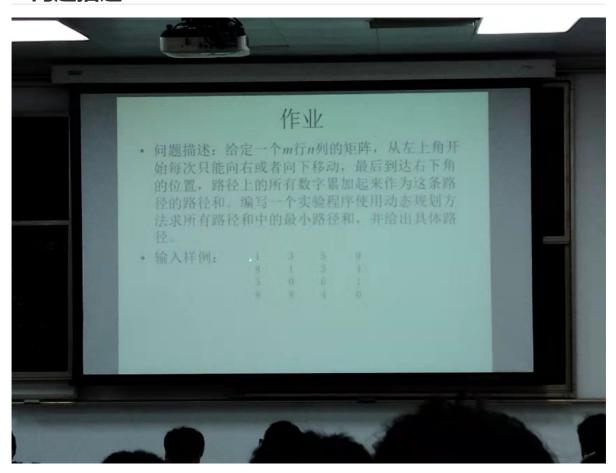
高级算法设计与分析作业_03

1.问题描述:



2.思路分析:

根据给出矩阵M, 计算形状,

```
col , row = Matrix.shape
```

初始化全零矩阵dp,

```
dp = np.zeros((col,row))
```

同时为了输出最短路径,我们可以新设同等形状大小的列表 min_id ,

```
min_id = [[0 for _ in range(col)] for _ in range(row)]
```

在规定了只能向下和向右移动寻找最短路径的情况下,dp数组转移方程可以定义为比较dp[i-1,j]和dp[i,j-1]的大小. 先更新dp的第0行和第0列边界值, dp数组更新规则:

```
egin{cases} dp[i,0] = dp[i-1,0] + Matrix[i,0] & 0 \leq i \leq col \ dp[0,j] = dp[0,j-1] + Matrix[0,j] & 0 \leq i \leq row \ dp[i,j] = min(dp[i-1,j] + Matrix[i,j], dp[i,j-1] + Matrix[i,j]) & 0 \leq i \leq col, \ 0 \leq i \leq row \end{cases}
```

并将每次更新dp时选取的上一个点(i-1,j)或(i,j-1)放入 $min_id[i,j]$ 中,在最后输出路径的时候使用while循环输出完整路径:

```
      id = [col-1,row-1]
      #根据最后的点坐标,从min_id

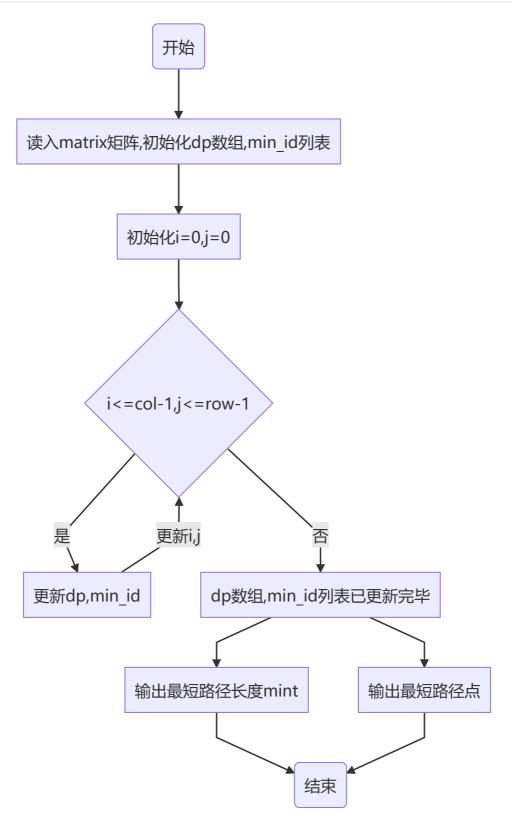
      列表中一直查找上一个点坐标直到[0,0]
      #逐一输出最短路径上的点

      traj.append([col-1,row-1])
      #逐一输出最短路径上的点

      id = min_id[id[0]][id[1]]
      traj.append(id)

      traj.reverse()
      #转置列表即为正序输出
```

3.流程图



4.完整代码如下

```
import numpy as np
def mintraj(matrix):
   col,row = matrix.shape
   dp = np.zeros((col,row))
                                                        #dp数组
   min_id = [[0 for _ in range(col)] for _ in range(row)] #列表,记录达到本点的最短
路径的上一个点
   traj = []
                                                        #存储最短路径
   # dp updatad
   dp[0,0] = matrix[0,0]
                                                        #首先更新dp[0,0]
   #更新dp[:,0]和dp[0,:]
   for i in range(1,col):
       dp[i,0] = dp[i-1,0] + matrix[i, 0]
       min_id[i][0] = [i-1, 0]
                                                        #记录到达该点的上一个点
   for j in range(1, row):
       dp[0,j] = dp[0,j-1] + matrix[0,j]
       min_id[0][j] = [0, j-1]
   #更新dp[i,j]
   for i in range(1,col):
                                                        #由于只能向右,向下移动,
对于dp[i,j],只需比较左边和上边的点的大小即可
       for j in range(1,row):
           dp[i,j] = min(dp[i-1,j] + matrix[i,j],
                        dp[i,j-1] + matrix[i,j])
                                                            #更短的路径长度计入
dp[i,j]中
          if dp[i-1,j]<=dp[i,j-1]:
                                                       #根据左边和上边的点大小比
较,记录下上一个点的坐标
               min_id[i][j] = [i-1,j]
           else:
               min_id[i][j] = [i,j-1]
   mint = dp[col-1, row-1]
                                                        #最短路径记录在dp[col-
1, row-1]中
   id = \lceil col - 1, row - 1 \rceil
                                                        #根据最后的点坐标,从
min_id中一直查找上一个点坐标直到结束
   traj.append([col-1,row-1])
   while(id!=[0,0]):
                                                        #逐一输出路径上的点
        id = min_id[id[0]][id[1]]
        traj.append(id)
   traj.reverse()
   return mint, traj
                                                        #返回最短路径长度,轨迹
def main():
                                                        #建立随机矩阵matrix
   matrix = np.random.randint(0,15,(5,5))
   mint,traj = mintraj(matrix)
   print("最短路径长度: ",mint,'\n',
         "最短路径: ",traj,'\n',
         "原矩阵", matrix)
if __name__ == '__main__':
   main()
```

5.运行结果

```
■ 項目 ▼
 misaka (GMM) [misaka] D:\misaka (GMM)
                                                    min_id[0][j] = [0, j-1]
                                                                                                                                     A4 A60 x4 ^
                                                 #由于只能向右,向下移动,对于dp[i,j],只需比较左边和上边的点的大小即可
 |||| 外部库
 草稿文件和控制台
                                                                                                #根据左边和上边的点大小比较,记录下上一个点的坐标
                                                          min_id[i][j] = [i-1,j]
                                        21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
                                                           min_id[i][j] = [i,j-1]
                                                 mint = dp[col-1,row-1]
                                                                                                #最短路径记录在dp[col-1,row-1]中
#根据最后的点坐标,从min_id中一直查找上一个点坐标直到结束
                                                 id = [col-1,row-1]
                                                 traj.append([col-1,row-1])
                                                 while(id!=[0,0]):

id = min_id[id[0]][id[1]]
                                                     traj.append(id)
                                                 trai.reverse()
                                                 return mint,traj
                                                 matrix = np.random.randint(0,15,(5,5))
                                                 mint,traj = mintraj(matrix)
                                              mintraj ×
取起姆涅瓦庚: 41.0
      最短路径: [[0, 0], [1, 0], [2, 0], [2, 1], [3, 1], [3, 2], [3, 3], [4, 3], [4, 4]]
      原矩阵:
[[0 6 11 2 0]
<u>=</u>
      [ 9 13 0 13 8]
[ 3 1 13 13 2]
  =
ė ii
      [ 9 1 6 1 14]
[12 1 12 11 9]]
      进程已结束,退出代码为 0
> misaka (GMM) [misaka] D:\misaka (GMM)
                                                      min_id[0][j] = [0, j-1]
 > IIII 外部库
                                                   for i in range(1,col):
                                                                                                 #由于只能向右,向下移动,对于dp[i,j],只需比较左边和上边的点的大小即可
   草稿文件和控制台
                                                     #根据左边和上边的点大小比较,记录下上一个点的坐标
                                                           min_id[i][j] = [i-1,j]
                                                           min_id[i][j] = [i,j-1]
                                                  mint = dp[col-1,row-1]
id = [col-1,row-1]
                                                                                                #最短路径记录在dp[col-1,row-1]中
#根据最后的点坐标,从min_id中一直查找上一个点坐标直到结束
                                                  traj.append([col-1,row-1])
                                                  while(id!=[0,0]):
    id = min_id[id[0]][id[1]]
    traj.append(id)
                                                  trai.reverse()
                                                  return mint, traj
                                               def main():
                                                 matrix = np.random.randint(0,15,(5,5))
                                                  $ -
运行: 🧶 mintraj 🗡
       D:\anaconda\envs\misaka\python.exe "C:/Users/92062/Desktop/SYSU course/高级算法作业/mintraj.py"
▶ ↑
       最短路径长度: 34.0
 ş
        最短路径: [[0, 0], [1, 0], [1, 1], [2, 1], [2, 2], [3, 2], [3, 3], [4, 3], [4, 4]]
 原矩阵: [[3147129]
<u>=</u>
    ÷
        [10 2 4 13 2]
[4 2 2 12 3]
    î
        [10 9 0 0 8]
[1 7 7 5 10]]
```

```
② 正

> ■ misaka (GMM) [misaka] D\misaka (GMM)

> III 分配條

『 查稿文件和控制台
                                                                 ▲ 4 ▲ 60 × 4 ~
#由于只能向右,向下移动,对于dp[i,j],只需比较左边和上边的点的大小唧可
                                                     18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
                                                                          if dp[i-1,j]<=dp[i,j-1]:
    min_id[i][j] = [i-1,j]</pre>
                                                                                                                             #根据左边和上边的点大小比较,记录下上一个点的坐标
                                                                          else:
                                                                              min_id[i][j] = [i,j-1]
                                                                 mint = dp[col-1,row-1]
id = [col-1,row-1]
traj.append([col-1,row-1])
while(id1=[0,0]):
id = min_id[id[0]][id[1]]
traj.append(id)
                                                                                                                             #最短路径记录在dp[col-1,row-1]中
#根据最后的点坐标,从min_id中一直查找上一个点坐标直到结束
                                                                 traj.reverse()
return mint,traj
                                                             def main():
                                                               运行: 🥏 mintraj 🗴
                                                                                                                                                                                           $ −
► ↑

• ↓
         D:\anaconda\envs\misaka\python.exe "C:/Users/92062/Desktop/SYSU course/高級算法作业/mintraj.py"
         最短路径长度: 31.0
          最短路径: [[0, 0], [0, 1], [0, 2], [0, 3], [1, 3], [1, 4], [2, 4], [3, 4], [4, 4]]
原矩阵: [[7 6 1 1 5]
[2 4 5 0 3]
[1 12 11 4 5]
[2 13 12 4 2]
= =
* II
          [10 12 2 5 6]]
进程口绘画,澳中伊瓜市 a
▶ 运行 III TODO ● 问题 图 终端 参 Python Packages ● Python 控制台
```

三个用例输出结果均满足问题描述。