矩阵连乘——动态规划

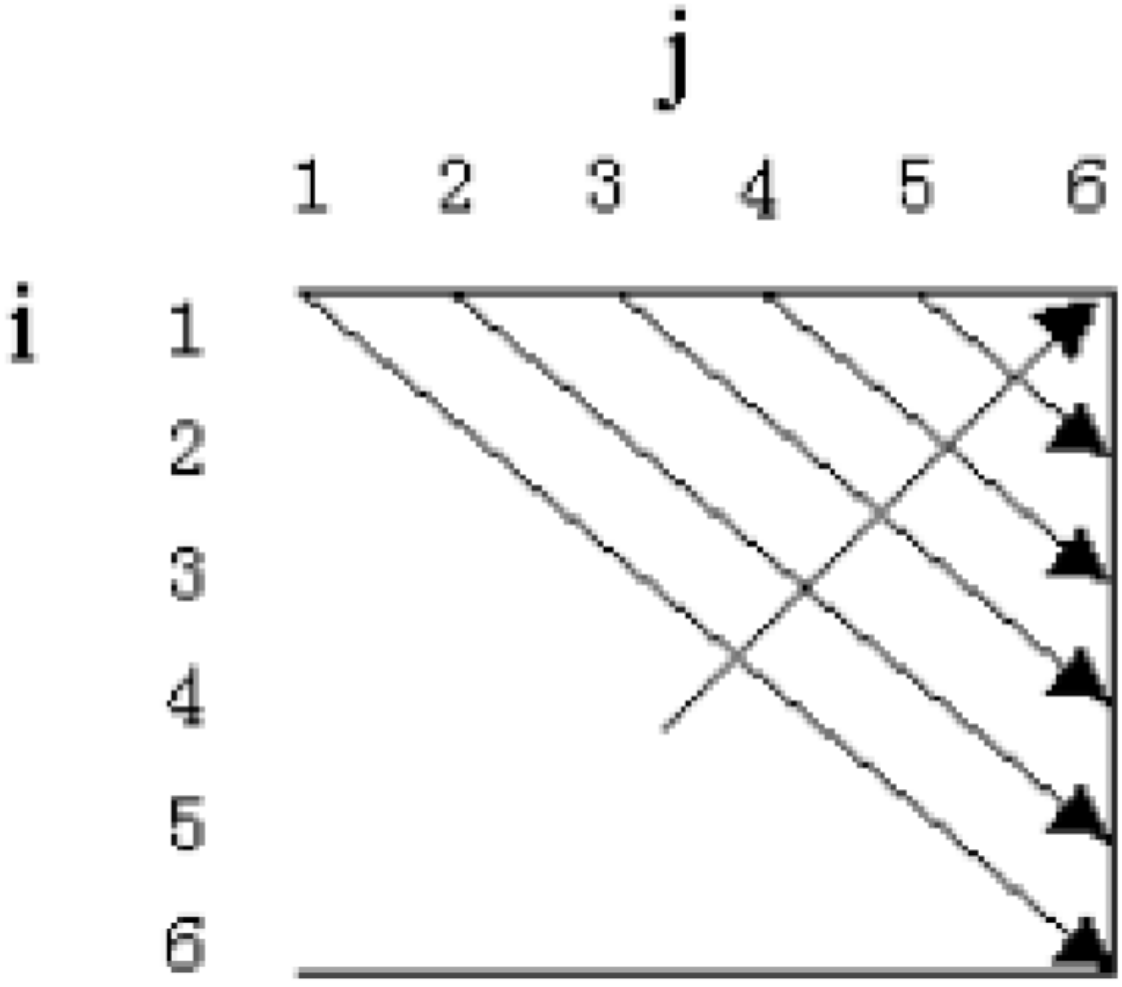
2054305 陈敬麒

实现

用于存储的矩阵：dp[矩阵数][矩阵数]

记录路径：path[矩阵数][矩阵数]

计算原理



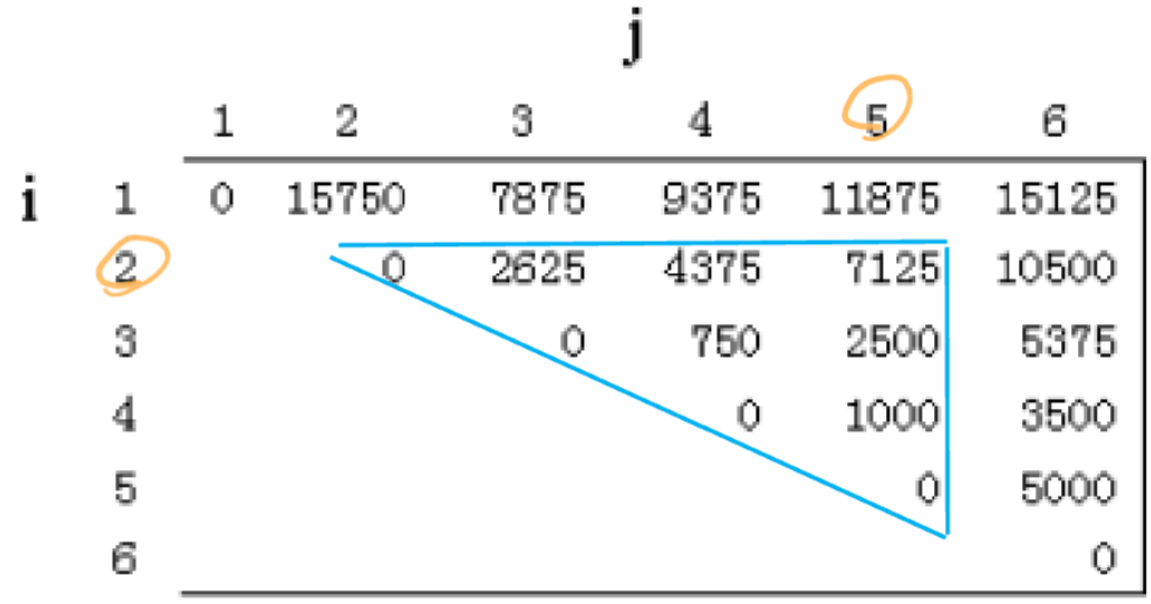
首先dp[i][j]记录Ai...Aj的乘法次数
同一个斜线上的共同点：连续的相同个数矩阵相乘 的所有情况
从内到外的原理：从一个矩阵自身，到两个矩阵，到三个，到cnt个逐步递推
while循环顺序入下——（以4个矩阵相乘为例）

```
for num in range(cnt):  # （矩阵数量1->cnt）整体上从左下到右上
    for row in range(cnt-num):  # 只考虑num数量矩阵相乘下，遍历所有可能性
        col = row+num  # dp[row][col]的意义是：从第row个矩阵连乘到第col个矩阵 的最小乘法次数
        ''' 执行顺序如下:
            dp[row][row+num] = j
            j = j+1
            [ 1.  5.  8. 10.]
            [ 0.  2.  6.  9.]
            [ 0.  0.  3.  7.]
            [ 0.  0.  0.  4.]
        '''
```

具体举例(注意下图中不是从零开始计算):

dp[2][5]的计算只需要考虑蓝色三角形之内的数据

计算的时候把2——5的矩阵分成两部分，各部分的数据是蓝色三角形内已经计算过的



i \ j	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

$$m[2][5]=\min\begin{cases} m[2][2]+m[3][5]+p_1p_2p_5=0+2500+35\times15\times20=13000 \\ m[2][3]+m[4][5]+p_1p_3p_5=2625+1000+35\times5\times20=7125 \\ m[2][4]+m[5][5]+p_1p_4p_5=4375+0+35\times10\times20=11375 \end{cases}$$

$$=7125$$

$$\therefore\ s[2][5]=3 \quad (A_2A_3)(A_4A_5)$$

代码如下

```
if row == col:
    dp[row][col] = 0
else:
    tmp = [dp[row][mid]+dp[mid+1][col]+P[row]*P[mid+1]*P[col+1] for mid in
range(row, col)]
    dp[row][col] = min(tmp)
    path[row][col] = tmp.index(dp[row][col]) + row
```

其中tmp就是需要计算最小值的数组，每个元素的mid加括号为：(left…mid)(mid+1…right)

path[left][right]就记录这个加括号的mid值

加括号的位置求解（路径）

利用递归，从每一次分隔的位置划分为两半部分，在两端加括号（分割位置记录在path\[left\]\[right\]中

括号记录方式：数组bracket[矩阵数量][2]

bracket[i][0]：矩阵i左边的左括号数

bracket[i][1]：矩阵i右边的右括号数

```
def add_bracket(left, right):
    if left == right:
        return
    bracket[left][0] = bracket[left][0] + 1
    bracket[right][1] = bracket[right][1] + 1

    if right-left >= 2:
        mid = path[left][right]
        add_bracket(left, mid)
        add_bracket(mid+1, right)

bracket = np.zeros((cnt, 2), int)
add_bracket(0, path[0][cnt-1])
add_bracket(path[0][cnt-1]+1, cnt-1)
```

输出如下：

```
#  输入
Input（输入数字，并用英文,分隔）: P = 30,35,15,5,10,20,25
#  dp数组
[    0 15750  7875  9375 11875 15125]
[    0     0  2625  4375  7125 10500]
[    0     0     0   750  2500  5375]
[    0     0     0     0  1000  3500]
[    0     0     0     0     0  5000]
[    0     0     0     0     0     0]
#  路径数组
[-1  0  0  2  2  2]
[-1 -1  1  2  2  2]
```

```
[-1 -1 -1  2  2  2]
[-1 -1 -1 -1  3  4]
[-1 -1 -1 -1 -1  4]
[-1 -1 -1 -1 -1 -1]
#  最后输出
Output:15125
(A(BC))((DE)F)
```

```
Input（输入数字，并用英文,分隔）： P = 10,30
Output:0
(A)
```

```
Input（输入数字，并用英文,分隔）： P = 10,30,5,60
Output:4500
(AB)C
```

复杂度分析

空间复杂度：n*n

时间复杂度：三重循环O(n^3)