Python 作业 05_2212794_曹瑜

2-1：

In [10]:
```
mat1 = []
print('请逐行输入第一个矩阵的元素(同一行元素之间以空格隔开，最后一行输入0表示矩阵结束):')
while True:
    linedata = input()
    if linedata == '0':
        break
    val_list = linedata.split()
    mat1.append([eval(x) for x in val_list])

mat2 = []
print('请逐行输入第二个矩阵的元素(同一行元素之间以空格隔开，最后一行输入0表示矩阵结束):')
while True:
    linedata = input()
    if linedata == '0':
        break
    val_list = linedata.split()
    mat2.append([eval(x) for x in val_list])
print('第一个矩阵:\n', mat1)
print('第二个矩阵:\n', mat2)
```

```
请逐行输入第一个矩阵的元素(同一行元素之间以空格隔开，最后一行输入0表示矩阵结束):
73 32 108
105 107 101
32 80 121
116 104 111
110 33 0
0
请逐行输入第二个矩阵的元素(同一行元素之间以空格隔开，最后一行输入0表示矩阵结束):
1 -1 2
-2 3 -3
2 -2 1
0
第一个矩阵:
 [[73, 32, 108], [105, 107, 101], [32, 80, 121], [116, 104, 111], [110, 33, 0]]
第二个矩阵:
 [[1, -1, 2], [-2, 3, -3], [2, -2, 1]]
```

2-6：

```
import numpy as np
import random
from time import per_counter
class MyMatrix:
    def __init__(self):
        self.mat = [] #
    def inputElements(self):
        print('请逐行输入矩阵元素(同一行元素之间以空格隔开，最后一行输入0表示矩阵结束):' )
        while True:
            linedata = input()
            if linedata=='0':
                 break
            val_list = linedata.split()
            self.mat.append([eval(x) for x in val_list])
    def outputElements(self):
        for rowindex in range(len(self.mat)):
            for colindex in range(len(self.mat[0])):
                print(self.mat[rowindex][colindex], end =' ' )
    def __mul__(self,mat2):
        mat_rlt = MyMatrix()
        for row1 in range(len(self.mat)):
             row_rlt =[]
             for col2 in range(len(mat2.mat[0])):
                 rlt=0
                 for col1 in range(len(self.mat[0])):
                     rlt += self.mat[row1][col1]*mat2.mat[col1][col2]
                 row_rlt.append(rlt)
             mat_rlt.mat.append(row_rlt)
        return mat_rlt
def randomElements(self,rows,cols):
        self.mat = []
        for r in range(rows):
            linedata = [random.random()*10000 for _ in range(cols)]
            self.mat.append(linedata)
```

```
if __name__=='__main__':
    mat1,mat2=MyMatrix(),MyMatrix()
    n_vals=[10,50,100,150,200]
    repeats=10
    for n in n_vals:
        array_total_time=0
        list_total_time=0
        for i in range(repeats):
            mat1.randomElements(n,n)
            mat2.randomElements(n,n)
            array1=np.array(mat1.mat)
            array2=np.array(mat2.mat)
            start=perf_counter()
            array_rlt=array1@array2
            end=perf_counter()
            array_total_time+=end-start
            start=perf_counter()
            mat_rlt=mat1*mat2
            end=perf_counter()
            list_total_time+=end-start
        print('两个%d*%d矩阵（ndarray对象）乘积运算消耗时间平均值为：%.8f秒'%(n,n,array_total_time/repeats))
        print('两个%d*%d矩阵（列表对象）乘积运算消耗时间平均值为：%.8f秒'%(n,n,list_total_time/repeats))
```

```
两个10*10矩阵（ndarray对象）乘积运算消耗时间平均值为：0.00000612秒
两个10*10矩阵（列表对象）乘积运算消耗时间平均值为：0.00013645秒
两个50*50矩阵（ndarray对象）乘积运算消耗时间平均值为：0.00050963秒
两个50*50矩阵（列表对象）乘积运算消耗时间平均值为：0.01221263秒
两个100*100矩阵（ndarray对象）乘积运算消耗时间平均值为：0.00045228秒
两个100*100矩阵（列表对象）乘积运算消耗时间平均值为：0.09534268秒
两个150*150矩阵（ndarray对象）乘积运算消耗时间平均值为：0.00034629秒
两个150*150矩阵（列表对象）乘积运算消耗时间平均值为：0.32136067秒
两个200*200矩阵（ndarray对象）乘积运算消耗时间平均值为：0.00072982秒
两个200*200矩阵（列表对象）乘积运算消耗时间平均值为：0.74146493秒
```

2-10：

In [7]:

```
import numpy as np
X = np.array([
    [1,-1,2],
    [-2,3,-3],
    [2,-2,1]
])
X_inv=np.linalg.inv(X)
A=np.array([
    [73,32,108],
    [105,107,101],
    [32,80,121],
    [116,104,111],
    [110,33,0]
])
B=A@X
print('密文矩阵：\n',B)
C=B@X_inv
print('还原的明文矩阵：\n',np.around(C,0))
```

```
密文矩阵：
 [[ 225 -193  158]
 [  93   14  -10]
 [ 114  -34  -55]
 [ 130  -26   31]
 [  44  -11  121]]
还原的明文矩阵：
 [[ 73.  32. 108.]
 [105. 107. 101.]
 [ 32.  80. 121.]
 [116. 104. 111.]
 [110.  33.   0.]]
```

2-11：

In [8]:
```
import numpy as np
F=np.array([
    [1,2,-1],
    [-2,-3,-4],
    [3,4,5]
])
C=np.array([
    [0.3,0.1],
    [0.2,0.4]
])
R=np.zeros((2,2))
for row in range(2):
    for col in range(2):
        R[row,col]=np.sum(F[row:row+2,col:col+2]*C)
print('卷积结果：\n',R)
```

```
卷积结果：
 [[-1.1 -1.7]
 [ 1.3  1.5]]
```

In [ ]: