ESE5023 Assignment 01

12132191 Penghan Chen

September 2021

1 Flowchart

code availability:

```
def Print_values(a,b,c):
    if (a>b):
        if(b>c):
            print(a,b,c)
        else:
            if (a>c):
                print(a,c,b)
            else:
                print(c,a,b)
    else:
        if(b>c):
            if(a>c):
                print(c,a,b)
        else:
            print(c,b,a)

import random
a=random.randint(0,999)
b=random.randint(0,999)
c=random.randint(0,999)
print('init a,b,c:')
print(a,b,c)
print('\nfunction output:')
Print_values(a,b,c)
```

output:

```
init a,b,c:
851 292 530

function output:
851 530 292
```

2 Matrix multiplication

2.1 Sub question

code availability:

```
import numpy as np

M1=np.random.randint(0, 51, size=(5, 10))
M2=np.random.randint(0, 51, size=(10, 5))

print('M1\n',M1)
print('M2\n',M2)
```

output:

```
M1
 [[29 10 38 21 48 46 29 11 24 41]
 [47  6 26 10 39 2 45 13 19 22]
 [ 4  1  3 14 39 22 10 42 48 39]
 [ 6 33 25 45 45 32 47 30 26 28]
 [40 20 22 46 22 45 20 26 14 46]]
M2
 [[41 45 37 44 19]
 [ 6  6 47  7 25]
 [17 15  6 34 39]
 [45 23 19 1 46]
 [17 22 21 29 31]
 [27 42 39 33 19]
 [ 2  5 35 44 27]
 [38 44 28 42 25]
 [ 0 16 28 4 42]
 [28 18 32 12 34]]
```

2.2 Sub question

code availability:

```
import numpy as np

def Matrix_multip(M1,M2):
    Result=np.zeros((M1.shape[0],M2.shape[1]))
    for i in range(Result.shape[0]):
        for j in range(Result.shape[1]):
            for k in range(M1.shape[1]):
                Result[i,j]+=M1[i,k]*M2[k,j]
    return Result

# M1 M2 have been defined in the subquestion 2.1
```

```
print('M1M2\n',Matrix_multip(M1,M2))
```

output:

```
M1M2
 [[ 6522. 7157. 8279. 7895. 9071.]
 [ 4772. 5210. 6439. 7067. 6850.]
 [ 4816. 5703. 6274. 5020. 7151.]
 [ 6541. 6687. 9080. 7519. 10050.]
 [ 8109. 7978. 8935. 7397. 9113.]]
```

3 Pascal triangle

code availability:

```
from scipy.special import comb

def Pascal_triangle(k):
    Result=np.zeros((k))
    for i in range(k):
        Result[i]=comb(k-1,i)
    print(Result)

print('100th line of the Pascal triangel:')
Pascal_triangle(100)
print('200th line of the Pascal triangel:')
Pascal_triangle(200)
```

output:

```
100th line of the Pascal triangel:
[1.00000000e+00 9.90000000e+01 4.85100000e+03 1.56849000e+05
 3.76437600e+06 7.15231440e+07 1.12052926e+09 1.48870315e+10
 1.71200863e+11 1.73103095e+12 1.55792785e+13 1.26050526e+14
 9.24370525e+14 6.18617197e+15 3.80007707e+16 2.15337701e+17
 1.13052293e+18 5.51961194e+18 2.51448989e+19 1.07196674e+20
 4.28786696e+20 1.61305471e+21 5.71901217e+21 1.91462581e+22
 6.06298174e+22 1.81889452e+23 5.17685364e+23 1.39966784e+24
 3.59914587e+24 8.81170195e+24 2.05606379e+25 4.57640004e+25
 9.72485009e+25 1.97443926e+26 3.83273504e+26 7.11793650e+26
 1.26541093e+27 2.15461861e+27 3.51543037e+27 5.49849366e+27
 8.24774049e+27 1.18686997e+28 1.63901091e+28 2.17264238e+28
 2.76518120e+28 3.37966592e+28 3.96743390e+28 4.47391483e+28
 4.84674106e+28 5.04456723e+28 5.04456723e+28 4.84674106e+28
 4.47391483e+28 3.96743390e+28 3.37966592e+28 2.76518120e+28
 2.17264238e+28 1.63901091e+28 1.18686997e+28 8.24774049e+27
 5.49849366e+27 3.51543037e+27 2.15461861e+27 1.26541093e+27
 7.11793650e+26 3.83273504e+26 1.97443926e+26 9.72485009e+25
```

```
 4.57640004e+25 2.05606379e+25 8.81170195e+24 3.59914587e+24
 1.39966784e+24 5.17685364e+23 1.81889452e+23 6.06298174e+22
 1.91462581e+22 5.71901217e+21 1.61305471e+21 4.28786696e+20
 1.07196674e+20 2.51448989e+19 5.51961194e+18 1.13052293e+18
 2.15337701e+17 3.80007707e+16 6.18617197e+15 9.24370525e+14
 1.26050526e+14 1.55792785e+13 1.73103095e+12 1.71200863e+11
 1.48870315e+10 1.12052926e+09 7.15231440e+07 3.76437600e+06
 1.56849000e+05 4.85100000e+03 9.90000000e+01 1.00000000e+00]
200th line of the Pascal triangel:
[1.00000000e+00 1.99000000e+02 1.97010000e+04 1.29369900e+06
 6.33912510e+07 2.47225879e+09 7.99363675e+10 2.20395985e+12
 5.28950363e+13 1.12255022e+15 2.13284541e+16 3.66461620e+17
 5.74123205e+18 8.25854149e+19 1.09720623e+21 1.35322101e+22
 1.55620416e+23 1.67520801e+24 1.69382143e+25 1.61358779e+26
 1.45222901e+27 1.23785235e+28 1.00153508e+29 7.70746561e+29
 5.65214145e+30 3.95649902e+31 2.64781088e+32 1.69656030e+33
 1.04217276e+34 6.14522558e+34 3.48229449e+35 1.89841216e+36
 9.96666383e+36 5.04373594e+37 2.46252990e+38 1.16090695e+39
 5.28857612e+39 2.32983218e+40 9.93244246e+40 4.10031599e+41
 1.64012640e+42 6.36049017e+42 2.39275583e+43 8.73634104e+43
 3.09743000e+44 1.06689256e+45 3.57177074e+45 1.16272537e+46
 3.68196366e+46 1.13464594e+47 3.40393783e+47 9.94483799e+47
 2.83045389e+48 7.85050418e+48 2.12254372e+49 5.59579709e+49
 1.43891925e+50 3.60992023e+50 8.83808056e+50 2.11215145e+51
 4.92835339e+51 1.12301823e+52 2.49962123e+52 5.43568426e+52
 1.15508290e+53 2.39901834e+53 4.87073421e+53 9.66877089e+53
 1.87687905e+54 3.56335009e+54 6.61765016e+54 1.20236179e+55
 2.13753207e+55 3.71872018e+55 6.33187490e+55 1.05531248e+56
 1.72182563e+56 2.75044873e+56 4.30198392e+56 6.58911461e+56
 9.88367191e+56 1.45204563e+57 2.08952907e+57 2.94548074e+57
 4.06756864e+57 5.50318111e+57 7.29491449e+57 9.47500388e+57
 1.20590958e+58 1.50399959e+58 1.83822173e+58 2.20182602e+58
 2.58475229e+58 2.97385478e+58 3.35349582e+58 3.70649538e+58
 4.01536999e+58 4.26374340e+58 4.43777374e+58 4.52742573e+58
 4.52742573e+58 4.43777374e+58 4.26374340e+58 4.01536999e+58
 3.70649538e+58 3.35349582e+58 2.97385478e+58 2.58475229e+58
 2.20182602e+58 1.83822173e+58 1.50399959e+58 1.20590958e+58
 9.47500388e+57 7.29491449e+57 5.50318111e+57 4.06756864e+57
 2.94548074e+57 2.08952907e+57 1.45204563e+57 9.88367191e+56
 6.58911461e+56 4.30198392e+56 2.75044873e+56 1.72182563e+56
 1.05531248e+56 6.33187490e+55 3.71872018e+55 2.13753207e+55
 1.20236179e+55 6.61765016e+54 3.56335009e+54 1.87687905e+54
 9.66877089e+53 4.87073421e+53 2.39901834e+53 1.15508290e+53
 5.43568426e+52 2.49962123e+52 1.12301823e+52 4.92835339e+51
 2.11215145e+51 8.83808056e+50 3.60992023e+50 1.43891925e+50
 5.59579709e+49 2.12254372e+49 7.85050418e+48 2.83045389e+48
 9.94483799e+47 3.40393783e+47 1.13464594e+47 3.68196366e+46
 1.16272537e+46 3.57177074e+45 1.06689256e+45 3.09743000e+44
 8.73634104e+43 2.39275583e+43 6.36049017e+42 1.64012640e+42
 4.10031599e+41 9.93244246e+40 2.32983218e+40 5.28857612e+39
```

```
 1.16090695e+39 2.46252990e+38 5.04373594e+37 9.96666383e+36
 1.89841216e+36 3.48229449e+35 6.14522558e+34 1.04217276e+34
 1.69656030e+33 2.64781088e+32 3.95649902e+31 5.65214145e+30
 7.70746561e+29 1.00153508e+29 1.23785235e+28 1.45222901e+27
 1.61358779e+26 1.69382143e+25 1.67520801e+24 1.55620416e+23
 1.35322101e+22 1.09720623e+21 8.25854149e+19 5.74123205e+18
 3.66461620e+17 2.13284541e+16 1.12255022e+15 5.28950363e+13
 2.20395985e+12 7.99363675e+10 2.47225879e+09 6.33912510e+07
 1.29369900e+06 1.97010000e+04 1.99000000e+02 1.00000000e+00]
```

4 Add or double

4.1 Sub question

code availability:[1]

```
def Least_moves(x):
    if(x==1):
        return 0
    elif(x%2!=0):
        return 1+Least_moves(x-1)
    else:
        return 1+min(Least_moves(x-1),Least_moves(int(x/2)))

x=random.randint(0,101)
print('x=',x)
print('Least moves=',Least_moves(x))
```

output:

```
x= 72
Least moves= 7
```

5 Dynamic programming

5.1 Sub question

code availability:

```
import numpy as np

def Find_expression(x,print_str=True):
    string_0='a1b2c3d4e5f6g7h8i9'
    all_str=[]
    w=['+','-','']
    for a in ['-','']:
```

```
        for b in w:
            for c in w:
                for d in w:
                    for e in w:
                        for f in w:
                            for g in w:
                                for h in w:
                                    for i in w:
                                        str_t=string_0.replace('a',a)
                                        str_t=str_t.replace('b',b)
                                        str_t=str_t.replace('c',c)
                                        str_t=str_t.replace('d',d)
                                        str_t=str_t.replace('e',e)
                                        str_t=str_t.replace('f',f)
                                        str_t=str_t.replace('g',g)
                                        str_t=str_t.replace('h',h)
                                        str_t=str_t.replace('i',i)
                                        all_str.append(str_t)
    cnt=0
    for i in range(len(all_str)):
        if(count_str(all_str[i])==x):
            if(print_str==True):
                print(all_str[i],'=',x)
            else:
                pass
            cnt+=1
    return cnt

# count a expression
def count_str(string):
    string+='e'
    i=0
    result=0
    num='0'
    t='+'
    while(string[i]!='e'):
        if(string[i] not in ['+','-']):
            num+=string[i]
        elif(string[i]=='+'):
            if(t=='+'):
                result+=int(num)
            else:
                result-=int(num)
            t='+'
            num='0'
        elif(string[i]=='-'):
            if(t=='+'):
                result+=int(num)
            else:
                result-=int(num)
```

```
            t='-'
            num='0'
        i+=1
    if(t=='+'):
        result+=int(num)
    else:
        result-=int(num)
    return result
```

output:

```
x= 85
 we can find the expression:
-1+2+3-4-5-6+7+89 = 85
-1+2+34+56-7-8+9 = 85
-1+2-3+4-5+6-7+89 = 85
-1+23+45-6+7+8+9 = 85
-1-2+3+4+5-6-7+89 = 85
-1-2+3-4+5+67+8+9 = 85
-1-2-3-4+5-6+7+89 = 85
-12+3-4+5+6+78+9 = 85
-12+34-5+67-8+9 = 85
1+2+3-4-5+6-7+89 = 85
1+2+34+56-7+8-9 = 85
1+2-3+4+5-6-7+89 = 85
1+2-3-4+5+67+8+9 = 85
1+23+45+6-7+8+9 = 85
1-2+3+4-5+67+8+9 = 85
1-2-3+4-5-6+7+89 = 85
1-2-3-4+5+6-7+89 = 85
12+3+4+56-7+8+9 = 85
12-3-4+5+6+78-9 = 85
12-3-4+56+7+8+9 = 85
12-34+5+6+7+89 = 85
the number of solutions for 85 is 21
```

5.2 Sub question

code availability:

```
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(0,101)
y=[]

for i in range(len(x)):
    y.append(Find_expression(x[i],print_str=False))
```

```
9
10 fig,ax=plt.subplots()
11 ax.plot(x,y,label='a')
12 plt.xlabel('the given sum x')
13 plt.ylabel('the number of solutions')
14
15 max_solutions=max(y)
16 min_solutions=min(y)
17 max_index=y.index(max_solutions)
18 min_index=y.index(min_solutions)
19 print('the max number of solutions is',max_solutions)
20 print('the max number of solutions is yield by the number:',max_index)
21 print('the min number of solutions is',min_solutions)
22 print('the min number of solutions is yield by the number:',min_index)
```

output:

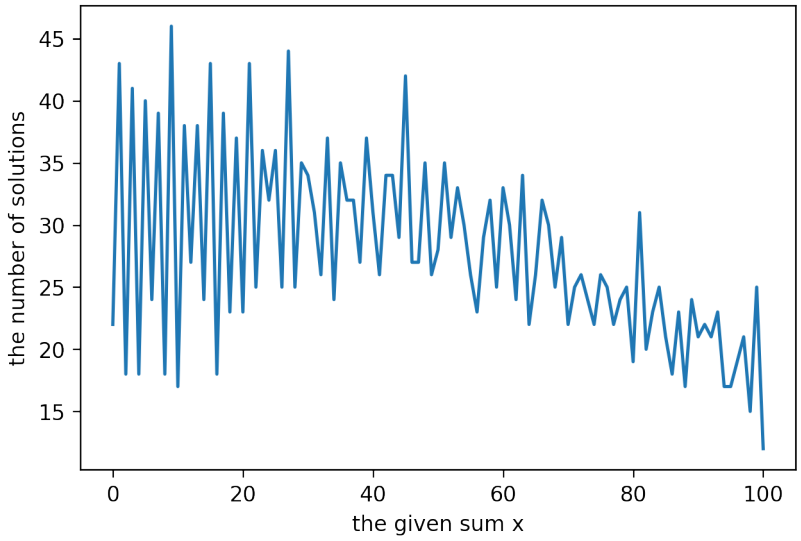

Figure 1: question 5.2

```
1 the max number of solutions is 46
2 the max number of solutions is yield by the number: 9
3 the min number of solutions is 12
4 the min number of solutions is yield by the number: 100
```

References

[1] OldDriver1995. Csdn article. https://blog.csdn.net/OldDriver1995/article/details/105529928/, 2020.