



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Assignment 1

## Report

张子严

UID: 12132873

量子科学与工程研究院

2021年9月29日

## Question 1. Flowchart

According to the Truth Table below, we have 5 possible cases.

	a>b	b>c	a>c	OUTPUT
1	F	F		c, b, a
2	F	T	F	c, a, b
3	F	T	T	a, c, b
4	T	F	F	c, a, b
5	T	T		a, b, c

Hence, we set 5 input to test our program.

Case 1.

**Input:** a = 1, b = 2, c = 3

**Output:** 3, 2, 1

Case 2.

**Input:** a = 1, b = 3, c = 2

**Output:** 2, 1, 3

Case 3.

**Input:** a = 2, b = 3, c = 1

**Output:** 2, 1, 3

Case 4.

**Input:** a = 2, b = 1, c = 3

**Output:** 3, 2, 1

Case 5.

**Input:** a = 3, b = 2, c = 1

**Output:** 3, 2, 1

## Resource:

```
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ python3 PS1_1.py
3, 2, 1
2, 1, 3
2, 1, 3
3, 2, 1
3, 2, 1
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ cat PS1_1.py
def flowchart(a, b, c):
    if a > b:
        if b > c:
            print('%s, %s, %s' % (a, b, c))
            return 0
        else:
            if a > c:
                print('%s, %s, %s' % (a, c, b))
                return 0
            print('%s, %s, %s' % (c, a, b))
    else:
        if b > c:
            if a > c:
                print('%s, %s, %s' % (a, c, b))
                return 0
            print('%s, %s, %s' % (c, a, b))
            return 0
        print('%s, %s, %s' % (c, b, a))
        return 0

if __name__ == '__main__':
    flowchart(1, 2, 3)
    flowchart(1, 3, 2)
    flowchart(2, 3, 1)
    flowchart(2, 1, 3)
    flowchart(3, 2, 1)
```

## Question 2. Matrix Multiplication

### 2.1 Make two matrices M1 and M2

Resource:

```
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ cat PS1_2.py
import numpy as np

def create_matrix():
    m1 = np.random.randint(0, 51, size=(5, 10))
    m2 = np.random.randint(0, 51, size=(10, 5))
    return m1, m2

def Matrix_multip(matrix_1, matrix_2):
    k = 0
    x = 0
    result = []
    row1, length1 = np.shape(matrix_1)
    row2, length2 = np.shape(matrix_2)
    for q in range(row1):
        for i in range(row1):
            for j in range(row2):
                x += int(matrix_1[k][j]) * int(matrix_2[j][i])
                result.append(x)
                x = 0
            k += 1
    return np.reshape(result, (5, 5))

if __name__ == '__main__':
    m1, m2 = create_matrix()
    print('Matrix 1:')
    print(m1, end='\n\n')
    print('Matrix 2:')
    print(m2)
    print(np.matmul(m1, m2)) # Output the correct answer calculated by built-in function
    print(Matrix_multip(m1, m2)) # Output reshaped matrix by our function
```

For the first question, we use `np.random.randint()` function to randomly generate two matrices named M1 and M2.

### 2.2 Matrix Multiplication

For question 2.2, we use `numpy` built-in function `np.matmul()` to verify our approach with simple operator. Results are showed below. We can see that our algorithm works well.

```
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ python3 PS1_2.py
Matrix 1:
[[33  0  4 23 27 18  0 29 25 16]
 [38 13 11 28 38 10 21 34 18 39]
 [12 16 47 20 32 44 16 29 43 34]
 [37 33 19 29 29 19 19 13  5  8]
 [ 5 10 37 35 24 10 39 48 21 47]]

Matrix 2:
[[27 26 36 11 49]
 [10 29 14 39 22]
 [38 14 30 44 33]
 [23 46 29 46 42]
 [23 38 24 23  3]
 [ 7  9 14 27 42]
 [43 32 31 50  4]
 [41 32 41 29  1]
 [46 21  5 36 49]
 [25 47 16 11 50]]
[[5058 5365 4445 4621 5606]
 [7262 8312 6503 6954 7171]
 [8479 7907 6474 9231 9175]
 [5298 6297 5442 6639 6003]
 [8854 8984 7195 9120 7231]]
[[5058 5365 4445 4621 5606]
 [7262 8312 6503 6954 7171]
 [8479 7907 6474 9231 9175]
 [5298 6297 5442 6639 6003]
 [8854 8984 7195 9120 7231]]
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$
```

## Question 3. Pascal triangle

Resource:

```
def pascal_triangle(k):
    """
    Find nth row of the pascal triangle
    :param n:
    :return: a list of row nth of pascal triangle
    """
    length = k - 1
    mid = length//2
    row = []
    for i in range(mid+1):
        num = factorial(length)//(factorial(i)*factorial(length - i))
        row.append(num)
    if k % 2 == 0:
        row.extend(row[::-1])
    else:
        row.extend(row[1::-1])
    return row

if __name__ == '__main__':
    print(pascal_triangle(int(input())))
```

According to the theory, for the  $n$ th line of Pascal triangle, it can be represented as two symmetric combination lists that consist of  $C_n^i$ , where  $i = 1, 2, 3, \dots, 1/2n$ .

We have our results of `Pascal_triangle(100)` and `Pascal_triangle(200)` below.

```
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ python3 PS1_3.py
100
[1, 99, 4851, 156849, 3764376, 71523144, 1120529256, 14887031544, 171200862756, 1731030945644, 15579278510796, 126050526132804, 924370524973896, 6186171974825304, 3800077070
2498296, 215337700647490344, 1130522928399324306, 5519611944537877494, 25144898858450330806, 107196674080761936594, 428786696323047746376, 1613054714739084379224, 5719012170
438571889976, 19146258135816088501224, 60629817430084280253876, 181889452290252840761628, 517685364210719623706172, 1399667836569723427057428, 3599145865465003098147672, 881
1701946483283447189128, 20560637875127661376774632, 45764000431735762419272568, 97248500917438495140954207, 197443926105102399225573693, 383273503615787010261407757, 7117936
49572175876199757263, 126541093257275113244012912, 2154618614921181030658724688, 3515430371713505892127392912, 5498493658321124600506947888, 8247740487481686900760421832, 1
1868699725888281149874753368, 16390109145274293016493707032, 21726423750712434928840495368, 27651812046361280818524266832, 33796659167774898778196326128, 3967433902304009856
5708730672, 44739148260023940935799206928, 48467410615025936013782474172, 50445672272782096667406248628, 50445672272782096667406248628, 48467410615025936013782474172, 447391
48260023940935799206928, 39674339023040098565708730672, 33796659167774898778196326128, 27651812046361280818524266832, 21726423750712434928840495368, 163901091452742930164937
07032, 11868699725888281149874753368, 8247740487481686900760421832, 5498493658321124600506947888, 3515430371713505892127392912, 2154618614921181030658724688, 126541093257275
7113244012912, 711793649572175876199757263, 383273503615787010261407757, 197443926105102399225573693, 97248500917438495140954207, 45764000431735762419272568, 205606378751276
61376774632, 8811701946483283447189128, 3599145865465003098147672, 1399667836569723427057428, 517685364210719623706172, 181889452290252840761628, 60629817430084280253876, 19
146258135816088501224, 5719012170438571889976, 1613054714739084379224, 428786696323047746376, 107196674080761936594, 25144898858450330806, 5519611944537877494, 1130522928399
324306, 215337700647490344, 38000770702498296, 6186171974825304, 924370524973896, 126050526132804, 15579278510796, 1731030945644, 171200862756, 14887031544, 1120529256, 7152
3144, 3764376, 156849, 4851, 99, 1]
```

(base) ZiYans-MacBook-Pro:ESE5023\_Assignments\_12132873 ziyanzhang\$ python3 PS1\_3.py  
200  
[1, 199, 19701, 1293699, 63391251, 2472258789, 7993637511, 2203959847089, 52895036330136, 1122550215450664, 21382454093562616, 366461620334848584, 5741232051912627816, 8258  
5414900589338584, 1097206226536401212616, 135322101272822816222644, 155620416463746238656036, 1675208012521503627885564, 16938214348828536681954036, 161358778967350073386147  
64, 1452229009170615066047532876, 12378523459120956991548018324, 100153507987433197477070330076, 770746561468507650149628192324, 5652141450769056101097273410376, 39564990155  
383392707680913872632, 264788197962950397351403038993768, 1696560304355200694140471323923032, 10421727583896232835434323846955768, 61452255753319166029629978545842632, 34822  
944926880607501236545093180248, 1898412158917053376377708907120493352, 996666383414530225982971762382590098, 50437359403955349931489584373269471102, 2462529900310761202537  
43264881256829498, 1160906953003644566910503963011639339062, 5288576119238825249258962498164134766838, 23298321822592662584573267221641999107962, 993244246121055615447597181  
55421154091838, 410031599039717830992469605718533482276562, 1640126396158871323969878422874133929106248, 6360490170469769280761235835048470603119352, 23927558260338655865720  
839569944246554591848, 87363410392399278393445856104215039745835352, 3097430004821428961222217126187671504553416248, 10668925572166997553208121242401849017322632, 3057177073  
5028382091998706667681023581492775768, 11627253669347711916506428088408438467412653032, 36819636619601087735603688946626721813473401268, 113464594480811515266860347570217040  
490499665132, 340637834424345480058140271065112207149895396, 9944837986847597514565991693896112134614123804, 2830453888564316215684167855903197037677487121596, 78505041  
618987072397277863173164144252564854404, 2122543723143513438889364448559194557174782880396, 55957970882874445207083244558110603832551700321044, 14389192512739143053249977  
1720855838426561515111256, 360992202688017097651709953615480436754356081770344, 88380805546524618388669196782727965846871786403256, 2112151454780677477844107741463807511600  
151218353544, 4928353394488247448302918063415550860400352842824936, 11230182325145350742854190341225599501568017133650264, 24996212272097716169578681727244076309941715555544  
136, 54356842559958525638607609470356165943841508430310264, 115508290439911866982041170124506852630663205414409311, 239901833990586185270393199489360386232915888168388569, 4  
7073420526341648882313465629913511442586803250970731, 966877088507514019423099864608634283908418579587747869, 187687905416164486123307620769701845233989004735039981, 3563  
50088335876475674391061127984662690616811217249819, 6617650164052342026252440542094828659282574077974892521, 12023617903700734104036124365214547845738761352940297679, 21375  
20717690193962730887760381418392424464627449418096, 37187201796529515524203051309156714189560369968302412304, 63318749004901607514183573850726297133575765081163566896, 10553  
1248341502679190305956417877161889292941801939278160, 172182563083504371310499192050220632556214799782111453840, 27504487349702646326225982106196594862524939911684530160, 4  
3019839187996446817937910021738441760548772652853213840, 65891146098070507151525133244348285193215378606992378160, 98836719147105760727287729986652242778982306791048856724  
0 1452045626975998213153980230668100850703567223226520240760, 2089529072965460843319142283156535370524645516350358395240, 29454807414091435984137308683049956427877533182288  
18460760, 4067568642898341159714199521944993982897373629935035017240, 5503181105097755686672152294396168329802329028735635611560, 7294914488152838933495643739083292902296110  
572975144880440, 947500387541690574120698554616565629838460338787257143560, 12059095841439698216081617967847198925216767948220145455440, 15039995937067477550393928027315045  
850551249912948720736560, 18382217256426805894925912033385056039562638782492880900240, 22018260230225514753262905622406275915520083816392571627760, 2584752287896038688426515  
0078476932596480098393156497128240, 29738547828481305339960979122548728901326564817932744007760, 33534958189564025170594295606278353867453360326605009200240, 370649537884655  
01504341063564833970064027398255721325958160, 40153699937504293296369485528570134236029681437668103121340, 42637433954257136180681000097347668312485125656710356922660, 44377  
737380961509086014918468667981304831457316167922511340, 4572425732805164085702088537420819373252294873706668402660, 45274257328051640582702088537420819373252294873706668420  
660, 44377737380961509086014918468667981304831457316167922511340, 42637433954257136180681000097347668312485125656710356922660, 4015369993750429329636948552857013423602968144  
3698103121340, 37064953788465501504341063564833970064027398255721325958160, 33534958189564025170594295606278353867453360326605009200240, 297385478284813053399609791225487289  
01326564817932744007760, 2584752287896038688426510078476932596480098393156497128240, 22018260230225514753262905622406275915520083816392571627760, 18382217256426805894925912  
033385056039562638782492880900240, 15039995937067477550393928027315045850551249912948720736560, 12059095841439698216081617967847198925216767948220145455440, 9475003875416905  
741206985546165656298384603387887257143560, 7294914488152838933495643739083292902296110572975144880440, 5503181105097755686672152294396168329802329028735635611560, 406756864  
2898341159714199521944993982897373629935035017240, 2945480741409143598413730688304995642787753318228818460760, 2089529072965460843319142283156535370524645516350358395240, 14  
52045626975998213153980230668100850703567223226520240760, 988367191471057607272877299866522427789823067910488567240, 65891146098070507151525153324434828519321537860699237816  
0, 430198391879964468179379100217384417605487726528532213840, 275044873497026463262225982106196594862524939911684530160, 1721825630835043713104991920502206325562147997821114  
53840, 105531248341502679190305956417877161889292941801939278160, 63318749004901607514183573850726297133575765081163566896, 3718720179652951552420305130915671418956036996830  
2412304, 21375320717690193962730887760381418392424464627449418096, 12023617903700734104036124365214547845738761352940297679, 661765016405234202625244054209482865928257407797  
4892521, 3563350088335876475674391061127984662690616811217249819, 187687905416164486123307620769701845233989007435039981, 96687708850751401942309986460863428390841857958774  
7869, 487073420526341648882313465629913511442586803250970731, 239901833990586185270393199489360386232915888168388569, 115508290439911866982041170124506852630663205414409311,  
54356842559958525638607609470356165943841508430310264, 24996212272097716169578681727244076309941715555544136, 11230182325145350742854190341225599501568017133650264, 4928353  
394488247448302918063415550860400352842824936, 2112151454780677477844107741463807511600151218353544, 88380805546524618388669196782727965846871786403256, 3609920226880170976  
51709953615480436754356081770344, 14389192512739143053249977172085583842656151511256, 55957970882874445207083244558110603832551700321044, 2122543723143513438889364448755919  
737380961509086014918468667981304831457316167922511340, 45274257328051640858702088538742081937252294873706668402660, 45274257328051640858702088538742081937252294873706668420  
660, 44377737380961509086014918468667981304831457316167922511340, 42637433954257136180681000097347668312485125656710356922660, 4015369993750429329636948552857013423602968144  
3698103121340, 37064953788465501504341063564833970064027398255721325958160, 33534958189564025170594295606278353867453360326605009200240, 297385478284813053399609791225487289  
01326564817932744007760, 2584752287896038688426510078476932596480098393156497128240, 22018260230225514753262905622406275915520083816392571627760, 18382217256426805894925912  
033385056039562638782492880900240, 15039995937067477550393928027315045850551249912948720736560, 12059095841439698216081617967847198925216767948220145455440, 9475003875416905  
741206985546165656298384603387887257143560, 7294914488152838933495643739083292902296110572975144880440, 5503181105097755686672152294396168329802329028735635611560, 406756864  
2898341159714199521944993982897373629935035017240, 2945480741409143598413730688304995642787753318228818460760, 2089529072965460843319142283156535370524645516350358395240, 14  
52045626975998213153980230668100850703567223226520240760, 988367191471057607272877299866522427789823067910488567240, 65891146098070507151525153324434828519321537860699237816  
0, 430198391879964468179379100217384417605487726528532213840, 275044873497026463262225982106196594862524939911684530160, 1721825630835043713104991920502206325562147997821114  
53840, 105531248341502679190305956417877161889292941801939278160, 63318749004901607514183573850726297133575765081163566896, 3718720179652951552420305130915671418956036996830  
2412304, 21375320717690193962730887760381418392424464627449418096, 12023617903700734104036124365214547845738761352940297679, 661765016405234202625244054209482865928257407797  
4892521, 3563350088335876475674391061127984662690616811217249819, 187687905416164486123307620769701845233989007435039981, 96687708850751401942309986460863428390841857958774  
7869, 487073420526341648882313465629913511442586803250970731, 239901833990586185270393199489360386232915888168388569, 115508290439911866982041170124506852630663205414409311,  
54356842559958525638607609470356165943841508430310264, 24996212272097716169578681727244076309941715555544136, 11230182325145350742854190341225599501568017133650264, 4928353  
394488247448302918063415550860400352842824936, 2112151454780677477844107741463807511600151218353544, 88380805546524618388669196782727965846871786403256, 3609920226880170976  
51709953615480436754356081770344, 14389192512739143053249977172085583842656151511256, 55957970882874445207083244558110603832551700321044, 2122543723143513438889364448755919  
4557174782880396, 7850504181489707239727786317316414425256426544804, 2830453888564316215684167855903197037677487121596, 994487398684759751456599516938961121346144123804, 340  
39378344243454800581042710651122071498995396, 113464594480811515266860347570217040690499665132, 36819636619601087735603688946626721813473401268, 11627253669347719165064280  
88408438467412653032, 3571770735028382091998706667681023581492775768, 1066892557216269975532081212424201849017322632, 309743000482142896122217126187671504553416248, 87363410  
392399278393445856104215039745835352, 239275582603386586572083969944246554591848, 6360490170469769280761235835048470603119352, 1640126396158871323969878422874133929106248,  
410031599039717830992469605718533482276562, 99324424612105561544759718155421154091838, 23298321822592662584573267221641999107962, 5288576119238825249258962498164134766838,  
1160906953003644566910503963011639339062, 246252990031076120253743264881256829498, 50437359403955349931489584373269471102, 9966663834314530225982971762382590098, 18984121589  
17053376377708907120493352, 348229449268808607501236545093180248, 61452255753319166029629978545842632, 10421727583896232835434323846955768, 169656030435520069414047132392303  
2, 264781087962950397351403038993768, 3956499015538339277680913872632, 5652141450769056101097273410376, 770746561468507650149628192324, 100153507987433197477070330076, 1237  
8523459120956991548018324, 1452229009170615066047532876, 161358778796735007338614764, 16938214348828536681954036, 1675208012521503627885564, 155620416463746238656036, 135322  
1012728281622264, 1097206226536401212616, 82584514900583938584, 5741232051912627816, 366461620334848584, 21328454093562616, 1122550215450664, 52895036330136, 2203959847089,  
79936367511, 2472258789, 63391251, 1293699, 19701, 199, 1]  
(base) ZiYans-MacBook-Pro:ESE5023\_Assignments\_12132873 ziyanzhang\$

We can zoom in for a clearer view.

## Question 5. Dynamic programming

Resource:

```
(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ cat PS1_5.py
import matplotlib.pyplot as plt

class Solution:
    def __init__(self, target):
        self.answer = []
        self.target = target
        self.string = ''

    def Find_expression(self, s, pointer, path=''):
        if len(s) == 0 and pointer == self.target:
            self.answer.append(path)
        for i in range(len(s)):
            self.Find_expression(s[i + 1:], pointer + int(s[:i + 1]), path + '+' + s[:i + 1])
            self.Find_expression(s[i + 1:], pointer - int(s[:i + 1]), path + '-' + s[:i + 1])

    def __len__(self) -> int:
        count = 0
        for string in self.answer:
            if string[0] == '+':
                count += 1
        return count

    def __str__(self):
        for string in self.answer:
            if string[0] == '+':
                string = string[1:]
                print(string, end=' ')
                print(eval(string))
        return ''

def get_key(my_dict, n):
    result = []
    for item in my_dict.items():
        if item[1] == n:
            result.append(item[0])
    return result
```



```

if __name__ == '__main__':
    target = int(input())
    total_solution = []
    a = Solution(target)
    a.Find_expression("123456789", 0)
    print('Possible Solutions:')
    print(a)
    print('Total number:%d' % len(a))

# Question 2

y_dict = {}
for i in range(1, 101):
    a = Solution(i)
    a.Find_expression("123456789", 0)
    y_dict[i] = len(a)
max_key = get_key(y_dict, max(y_dict.values()))
min_key = get_key(y_dict, min(y_dict.values()))
x = []
y = []
for key in y_dict.keys():
    x.append(key)
for value in y_dict.values():
    y.append(value)
plt.plot(x, y)
plt.text(1, 26, 'Max(1,26)', color='r')
plt.text(45, 26, 'Max(45,26)', color='r')
plt.text(88, 6, 'Min(88,6)', color='r')
plt.title('Distribution')
plt.show()

```

(base) Ziyans-MacBook-Pro:ESE5023\_Assignments\_12132873 ziyanzhang\$

## Question 1.

For the function `Find_expression()` we set 50 as the input.

**Input:** 50

**Output:**

```

(base) Ziyans-MacBook-Pro:ESE5023_Assignments_12132873 ziyanzhang$ python3 PS1_5.py
50
Possible Solutions:
1+2+3+4-56+7+89=50
1+2+3-4+56-7+8-9=50
1+2-3+4+56+7-8-9=50
1+2+34-5-6+7+8+9=50
1+2+34-56+78-9=50
1+2-34+5-6-7+89=50
1-2+3-45+6+78+9=50
1-2-3+4+56-7-8+9=50
1-2-3-4-5-6+78-9=50
1-2+34+5+6+7+8-9=50
1-2+34-5-67+89=50
1-2-34-5-6+7+89=50
1-23+4+5-6+78-9=50
1-23-4-5-6+78+9=50
12+3+4-56+78+9=50
12-3-4-5+67-8-9=50
12-3+45+6+7-8-9=50

Total number:17

```

## Question 2.

The Distribution Chart is showed below. Axis-x denotes the number, axis-y denotes the aggregation of possible solutions.

