SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

18CSC305J-ARTIFICIAL INTELLIGENCE

SEMESTER – 6

BATCH-2

REGISTRATION NUMBER	**RA1811028010049**
NAME	**SHUSHRUT KUMAR**

B.Tech-CSE-CC, Third Year (Section: J2)

Faculty Incharge: Vaishnavi Moorthy, Asst Prof/Dept of CSE

Year 2020-2021

INDEX

Exercise: 1

Date : 21-01-2021

TOY PROBLEM

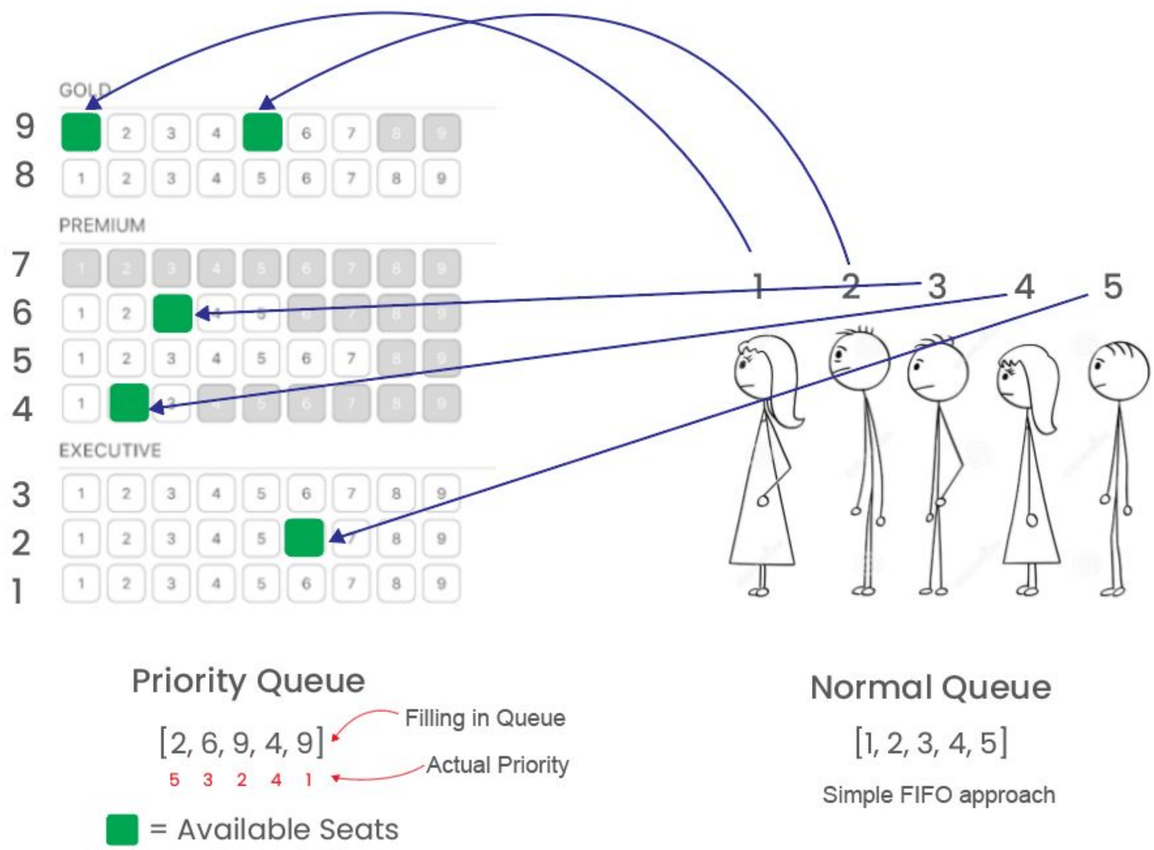
Problem Statement : Given an integer N and an array of seats[] where N is the number of people standing in a line to buy a movie ticket and seat[i] is the number of empty seats in the ith row of the movie theater. The task is to find the maximum amount a theater owner can make by selling movie tickets to N people. Price of a ticket is equal to the maximum number of empty seats among all the rows.

Algorithm :

1. Initialize queue q insert all seats array elements to the queue.
2. Tickets sold and the amount generated to be set to 0.
3. If tickets sold < N (People in the queue) and q top > 0
4. Then remove top element from queue and update total amount
5. Repeat step 3 and 4 until tickets sold = number of people in the queue.

Optimization technique : This problem can be solved by using a priority queue that will store the count of empty seats for every row and the maximum among them will be available at the top.

1. Create an empty priority_queue q and traverse the seats[] array and insert all elements into the priority_queue.
2. Initialize two integer variable ticketSold = 0 and ans = 0 that will store the number of tickets sold and the total collection of the amount so far.
3. Now check while ticketSold < N and q.top() > 0 then remove the top element from the priority_queue and update ans by adding top element of the priority queue. Also store this top value in a variable temp and insert temp – 1 back to the priority_queue.
4. Repeat these steps until all the people have been sold the tickets and print the final result.



Tool : VS Code and Python 3.9.0

Programming code :

```
def maxAmount(M, N, seats):

    q = []

    for i in range(M):
        q.append(seats[i])
```

```
    ticketSold = 0

    ans = 0

    q.sort(reverse = True)
    while (ticketSold < N and q[0] > 0):
        ans = ans + q[0]
        temp = q[0]
        q = q[1:]
        q.append(temp - 1)
        q.sort(reverse = True)
        ticketSold += 1

    return ans

if __name__ == '__main__':

    seats = []

    rows = int(input("Enter number of rows available : "))

    for i in range(0, rows):
        empty = int(input())
```

```
        seats.append(empty)


    print(seats)

    M = len(seats)

    N = int(input("Enter the number of People standing in the queue : "))

    print("Maximum Profit generated = ", maxAmount(N, M, seats))
```

Output screen shots :

```
PS E:\Studies\SRM University\SEM 6\AI\week 1> python -u "e:\Studies\SRM University\SEM 6\AI\week 1\solution.py"
Enter number of rows available : 4
2
3
5
3
[2, 3, 5, 3]
Enter the number of People standing in the queue : 4
Maximum Profit generated =  15
```

Result : Successfully found out the maximum amount the theater owner can make by selling movie tickets to N people for a movie.

Exercise: 2

Date : 29-01-2021

GRAPH COLORING PROBLEM

PROBLEM STATEMENT : Given a graph color its edges such that no two adjacent have the same color using minimum number of colors and return the Chromatic number.

ALGORITHM :

Initialize:

1. Color first vertex with first color.

Loop for remaining V-1 vertices.:

1. Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it.

2. If all previously used colors appear on vertices adjacent to v, assign a new color to it.

3. Repeat the following for all edges.

4. Index of color used is the chromatic number.

OPTIMIZATION TECHNIQUE:

Graph coloring problem is to assign colors to certain elements of a graph subject to certain constraints.

Vertex coloring is the most common graph coloring problem. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using the same color. The other graph coloring problems like Edge Coloring (No vertex is incident to two edges of same color) and Face Coloring (Geographical Map Coloring) can be transformed into vertex coloring.

Chromatic Number: The smallest number of colors needed to color a graph G is called its chromatic number. For example, the following can be colored at least 2 colors.

TOOLS : VS Code, Python 3.9.0

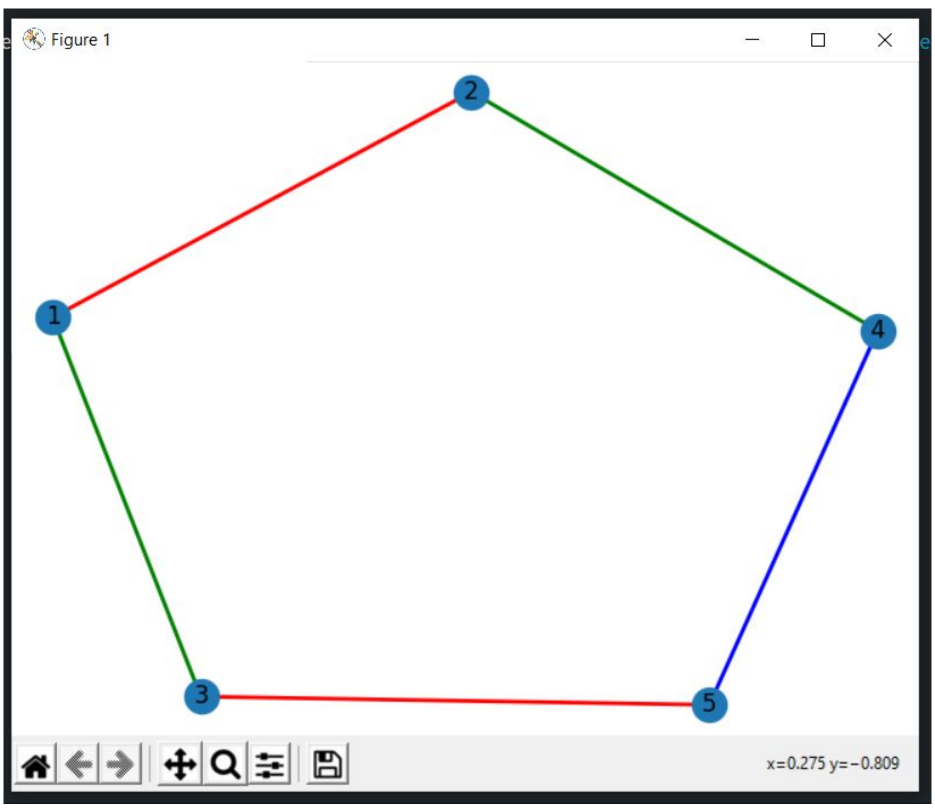
CODE - EDGE COLORING :

```
import matplotlib.pyplot as plt

import networkx as nx

from matplotlib.patches import Polygon

import numpy as np


G = nx.Graph()


colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}


G.add_nodes_from([1,2,3,4,5])

G.add_edges_from([(1,2), (1,3), (2,4), (3,5), (4,5)])


nodes = list(G.nodes)

edges = list(G.edges)


color_lists = []
```

```
color_of_edge = []

some_colors = ['red','green','blue','yellow']


for i in range(len(nodes) + 1):

    color_lists.append([])

    color_of_edge.append(-1)


def getSmallestColor(ls1,ls2):

    i = 1

    while(i in ls1 or i in ls2):

        i = i + 1

    return i


#iterate over edges

i = 0

for ed in edges:

    newColor = getSmallestColor(color_lists[ed[0]],color_lists[ed[1]])

    color_lists[ed[0]].append(newColor)

    color_lists[ed[1]].append(newColor)

    color_of_edge[i] = newColor

    i = i + 1


# Makin graph again
```

```
G = nx.Graph()


for i in range(len(edges)):
    G.add_edge(edges[i][0],edges[i][1],color=some_colors[color_of_edge[i]-1])


colors = nx.get_edge_attributes(G,'color').values()
nx.draw(G, edge_color=colors, with_labels=True, width=2)


plt.show()
```

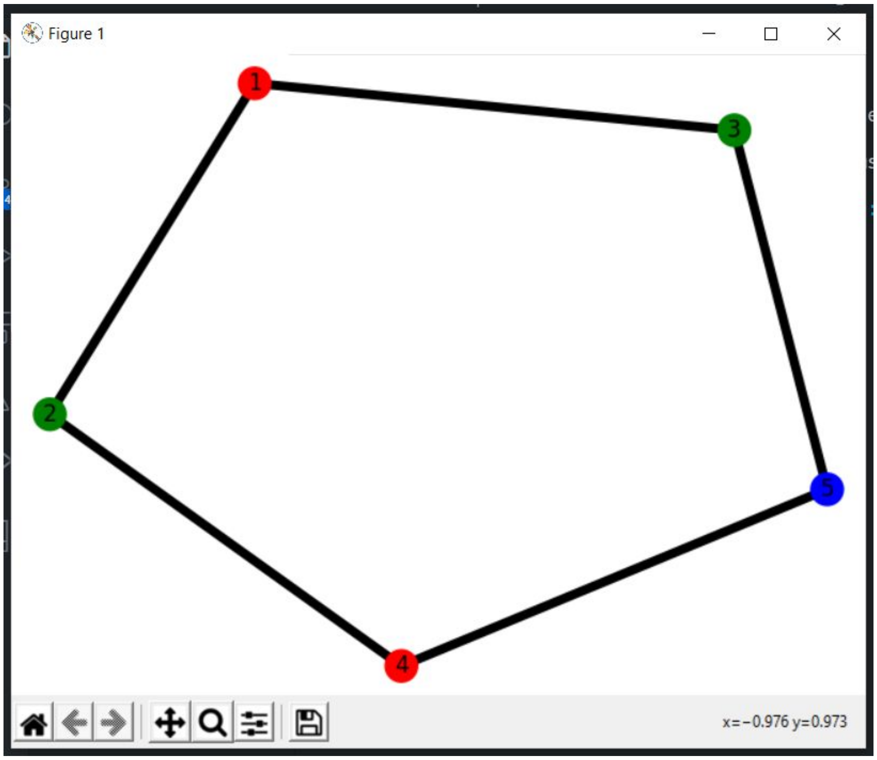
OUTPUT :



CODE - VERTEX COLORING :

```
import matplotlib.pyplot as plt
import networkx as nx


G = nx.Graph()


colors = {0:"red", 1:"green", 2:"blue"}


G.add_nodes_from([1,2,3,4,5])
G.add_edges_from([(1,2), (1,3), (2,4), (3,5), (4,5)])


d = nx.coloring.greedy_color(G, strategy = "largest_first")
node_colors = []
for i in sorted (d.keys()):
    node_colors.append(colors[d[i]])


nx.draw(G, node_color = node_colors, with_labels = True, width = 5)


plt.show()
```

OUTPUT :



CODE - FACE COLORING :

```
import networkx as nx

G = nx.Graph()

colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}

G.add_nodes_from([1,2,3,4,5])

G.add_edges_from([(1,2), (1,3), (2,4), (3,4), (4,5)])

nodes = list(G.nodes)

edges = list(G.edges)

some_colors = ['red','green','blue','yellow']

no_of_faces = len(edges)+2-len(nodes)-1
```

```
def regionColour(regions):
    print("NO OF FACES : "+str(regions))
    for i in range(1,regions+1):
        print("FACE 1 : "+some_colors[i%4])
regionColour(no_of_faces)
```

OUTPUT :

```
PS E:\Studies\SRM University\SEM 6\AI\week 2> python face_color.py
NO OF FACES : 4
FACE 1 : green
FACE 2 : blue
FACE 3 : yellow
FACE 4 : red
```

RESULT :

Edge, vertex and face coloring problem which are together known as graph coloring problem solved and visualized in an optimized way using greedy approach.

Exercise: 3

Date : 05-02-2021

CONSTRAINT SATISFACTION PROBLEM

1) SEND + MORE = MONEY

```
   5  4   3   2   1
      S   E   N   D
+     M   O   R   E
      c3  c2  c1
----------------------
  M   O   N   E   Y
```

1. From Column 5, M=1, since it is only carry-over possible from sum of 2 single digit number in column 4.
2. To produce a carry from column 4 to column 5 'S + M' is at least 9 so 'S=8or9' so 'S+M=9or10' & so 'O = 0 or 1'. But 'M=1', so 'O = 0'.
3. If there is c+rry from Column 3 to 4 then 'E=9' & so 'N=0'. But 'O = 0' so there is no carry & 'S=9' & 'c3=0'.
4. If there is no carry from column 2 to 3 then 'E=N' which is impossible, therefore there is carry & 'N=E+1' & 'c2=1'.
5. If there is carry from column 1 to 2 then 'N+R=E mod 10' & 'N=E+1' so 'E+1+R=E mod 10', so 'R=9' but 'S=9', so there must be c+rry from column 1 to 2. Therefore 'c1=1' & 'R=8'.

6. To produce carry 'c1=1' from column 1 to 2, we must h+ve 'D+E=10+Y'

as Y cannot be 0/1 so D+E is at least 12. As D is at most 7 & E is

At least 5 (D cannot be 8 or 9 as it is already assigned). N is at most 7

& 'N=E+1' so 'E=5or6'.

7. If E were 6 & D+E at least 12 then D would be 7, but 'N=E+1' & N would

also be 7 which is impossible. Therefore 'E=5' & 'N=6'.

8. D+E is at least 12 for that we get 'D=7' & 'Y=2'.

SOLUTION:

```
   9 5 6 7
+  1 0 8 5
-----------------
 1 0 6 5 2
```

VALUES:

S=9

E=5

N=6

D=7

M=1

O=0

R=8

Y=2

2. BASE + BALL = GAMES

Assuming numbers can't start with 0, G is 1 because two four-digit numbers can't sum to 20000 or more.

SE+LL=ES or 1ES.

If it is ES, then LL must be a multiple of 9 because SE and ES are always congruent mod 9. But LL is a multiple of 11, so it would have to be 99, which is impossible.

So SE+LL=1ES. LL must be congruent to 100 mod 9. The only multiple of 11 that work is 55, so L is 5.

SE+55=1ES. This is possible when E+5=S. The possibilities for ES are 27, 38, or 49.

BA+BA+1=1AM. B must be at least 5 because B+B (possibly +1 from a carry) is at least 10.

If A is less than 5, then A+A+1 does not carry, a and A must be even. Inversely, if A is greater than 5, it must be odd. The possibilities for A are 0, 2, 4, 7, or 9.

0 does not work because M would have to be 1.

2 and 7 don't work because M would have to be 5.

9 doesn't work because M would also have to be 9.

So A is 4, M is 9, and B is 7. This leaves 38 as the only possibility for ES. The ***full equation is:***

7483

+ 7455

14938

3. TWO + TWO = FOUR

F = 1 for carry over T >= 5.

'O' can't be 0 as R will be 0. So T can't be 5 so let T>=6

If T = 6, O = 2 and R = 4 and W + W = U for W can't be 1,2,6,4. W < 4 +s to

avoid carry over. W can't be 3 as U will be 6.

So T = 7, so, O can be 4 or 5 depending on whether or W + W > 10. If O is 4 then

R = 8. W can't be 1, 2. So W = 3

If W = 3 then U = 6 hence

Here is one + answer:

7 3 4

+ 7 3 4

----

1 4 6 8

4. CROSS + ROADS= DANGER

Solution:

C5C4C3C2C1

CROSS

+ROADS

DANGER

Since it is already mentioned that the carry value of resultaint cannot be 0 then

let's presume that the carry value of D is 1

We know that the sum of two similar values is even, hence R will have an even value

Hence S+S= R So R is an even number for sure.

So the value of R can b (0, 2, 4, 6, 8)

Value of R cannot be 0 as two different values cannot be allotted the same Digit, (if S=10 then their sum = 20 carry forward 2, then the value of R= 0) which is not possible.

IF S= 1 :

Not possible since D has the same value.

IF S = 2

Then R= 4 which is possible Hence S= 2 and R= 4

C4+C+R= A+10

C4+C+4= A+10

C4+C>5 (Being the v+lue of c+rry will be gener+ted when the v+lue of C is greater th+n 5

C= 9

C1+S+D= E

C1+2+1= E

Therefore E= 3

C4+C+R= A+10

C4+9+4= A+10

Therefore A= 3 but it c+nnot be possible +s E= 3

Now let's Consider S+D+C1= E

2+1+0= 3

Therefore E= 3 m+king C2= 0 since 2+1=3

Now let's consider the equation again:

C+R+C4= A+10

9+4+0= A+10

13= A+10

Therefore A= 3 but E= 3

So A is not equ+l to 3

Ag+in considering R= 6 So S= 3 C4= 0

C+R+C4= A+10

9+6+0= A+10

15= A+10

Therefore A= 5

And S+D+C1= E

3+1+0= E therefore E= 4 +nd C2= 0

Now considering the equ+tion

R+O+C3= N

6+0+C3= N

So 6+0+C3< or equ+l to 3

Let C3= 1

Then O< or equ+l to 2

Th+t is O= 0, 1, 2

Let O =2

Ag+in considering R+O+C3= N

6+2+1= N

Hence N= 9 but C= 9 so N c+nnot be equ+l to 9.

Now let N= 8 +nd C3= 0

Let us consider equ+tion

O+A+C2= G

Therefore G= 7

Hence D= 1 S= 3 A= 5 G= 7 C= 9 O= 2 E= 4 R= 6 N= 8

And C1= 0 C2= 0 C3= 0 C4= 0 C5= 1

Now verifying the above values in the equ+tion we get:

C5C4C3C2C1

CROSS

96233

ROADS

62513

Shape

DANGER

158746

5. If AA + BB = ABC

Explanation:

A A

B B +

C C

A B C

The digits are distinct and positive. Let's first focus on the value A, when we add three 2 digit numbers the most you get is in the 200's (ex: AA + BB + CC = ABC u 99 + 88 + 77 = 264). From this, we can tell that the largest value of A can be 2. So Either A = 1 or A = 2.

Now focus on value B, let's take the unit digit of the given question: A + B +C = C (units). This can happen only if A + B = 0 (in the units) u A and B add up to 10.

Two possibilities: 11 + 99 + CC = 19C u (1) or 22 + 88 + CC = 28C u (2)

Take equation (2), 110 + CC = 28C

Focus on ten's place, 1 + C = 8, here C = 7. Then 22 + 88 + 77 = 187

Thus, Equ+tion (2) is not possible.

From Equ+tion (1), 11+99+CC = 19C u 110 + CC =19C u 1 + C = 9, then C = 8.

11 + 99 + 88 = 198 u hence solved A = 1, B = 9 +nd C = 8

A + B + C = 18

6. N O + G U N + N O = H U N T

Solution:

N O

+ G U N

N O

————-

H U N T

————-

Here H = 1, from the NUNN column we must have “carry 1,” so G = 9, U = zero.

Since we have “carry” zero or 1 or 2 from the ONOT column, correspondingly we h+ve N + U = 10 or 9 or 8. But duplication is not allowed, so N = 8 with “carry 2” from ONOT. Hence, O + O = T + 20 – 8 = T + 12. Testing for T = 2, 4 or 6, we find only T = 2 +accept+able, O = 7. So we h+ve 87 + 908 + 87 = 1082.

HUNT = 1082

TOOLS : VS Code, Python 3.9.0

CODE :

```
def solutions():
    # letters = ('s', 'e', 'n', 'd', 'm', 'o', 'r', 'y')
    all_solutions = list()
    for s in range(9, -1, -1):
        for e in range(9, -1, -1):
            for n in range(9, -1, -1):
                for d in range(9, -1, -1):
                    for m in range(9, 0, -1):
                        for o in range(9, -1, -1):
                            for r in range(9, -1, -1):
                                for y in range(9, -1, -1):
                                    if len(set([s, e, n, d, m, o, r, y])) == 8:
                                        send = 1000 * s + 100 * e + 10 * n + d
                                        more = 1000 * m + 100 * o + 10 * r + e
                                        money = 10000 * m + 1000 * o + 100 * n + 10 * e + y

                                        if send + more == money:
                                            all_solutions.append((send, more, money))
    return all_solutions


print(solutions())
```

OUTPUT :

```
PS E:\Studies\SRM University\SEM 6\AI> python -u
[(9567, 1085, 10652)]
PS E:\Studies\SRM University\SEM 6\AI>
```

RESULT :

The constraint satisfying problem SEND + MORE = MONEY solved using the carry over technique and values for the alphabets obtained successfully.