Shriharsha A Vaidhyam

PROJECT 3

Problem 1:

Source code:

```
def grphC(graph, color):

    for i in range(6):
        for j in range(i + 1, 6):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True


def graphColoring(graph, m, i, color):

    if (i == 6):

        if (grphC(graph, color)):

            printSolution(color)
            return True
        return False

    for j in range(1, m + 1):
        color[i] = j
        if (graphColoring(graph, m, i + 1, color)):
            return True
        color[i] = 0
    return False



def printSolution(color):
    print("Solutions: " )
    for i in range(6):
        print(color[i], end=" ")


if __name__ == '__main__':

    graph = [
        [0,1,0,1,0,0],
        [1,0,1,0,1,0],
        [0,1,0,0,0,1],
```

```
        [1,0,0,0,1,0],
        [0,1,0,1,0,1],
        [0,0,1,0,1,0]
    ]
    m = 3


    color = [0 for i in range(6)]

    if (not graphColoring(graph, m, 0, color)):
        print("Solution does not exist")
```

Output:

harshavaidhyam@Harshas-MacBook-Pro Project 3 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 50840 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3/problem1.py

Solutions:
1 2 1 2 1 2
(red, green, red, green, red, green)
Total possible solutions: 3

The minimum no. of colors used would be 2 colors.
(1- Red, 2- green,3- white)

Problem 2:

Source code:

```
def knapSackalgo(W, wt, val, n):
    K = [[0 for w in range(W + 1)]
            for i in range(n + 1)]


    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i - 1] <= w:
                K[i][w] = max(val[i - 1]
                + K[i - 1][w - wt[i - 1]],
                            K[i - 1][w])
            else:
                K[i][w] = K[i - 1][w]

    res = K[n][W]
    print(res)

    w = W
    for i in range(n, 0, -1):
        if res <= 0:
            break

        if res == K[i - 1][w]:
            continue
        else:


            print(wt[i - 1])


            res = res - val[i - 1]
            w = w - wt[i - 1]


val = [20, 30, 35, 12, 3]
wt = [2 ,5, 7, 3, 1]
W = 9
n = len(val)


print("Maximum profit along with their weights:")
knapSackalgo(W, wt, val, n)
```

Output:

harshavaidhyam@Harshas-MacBook-Pro Project 3 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 54684 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3/problem2.py

Maximum profit along with their weights:
55
7
2

Problem 3:

Source Code:

```
def isomorphic_check(x, map):
    if check(x,map):
        if x == n:
            print(map)
            return True
        for i in range(n):
            flag = 0
            for j in range(x):
                if map[j] == i:
                    flag = 1
            if flag == 1:
                continue
            map[x] = i
            if isomorphic_check(x+1, map):
                return True
        return False
    else:
        return False

def check(x, map):
    for i in range(x-1):
        if graph1[i][x-1] != graph2[map[i]][map[x-1]]:
            return False
    return True

#test case 1:

n = 5
graph1 = [[0,1,0,1,0],
          [1,0,1,1,1],
          [0,1,0,1,1],
          [1,1,1,0,0],
          [0,1,1,0,0]]

graph2 = [[0,1,0,1,1],
          [1,0,0,1,0],
          [0,0,0,1,1],
          [1,1,1,0,1],
          [1,0,1,1,0]]

#test case 2:

# n = 6
```

```
# graph1 = [[0,1,1,0,1,0],
#           [1,0,1,0,0,1],
#           [1,1,0,1,0,0],
#           [0,0,1,0,1,1],
#           [1,0,0,1,0,1],
#           [0,1,0,1,1,0]]

# graph2 = [[0,1,0,1,1,0],
#           [1,0,1,0,0,1],
#           [0,1,0,1,0,1],
#           [1,0,1,0,0,1],
#           [1,0,0,1,0,1],
#           [0,1,0,1,1,0]]


#test case 3:

# n = 7
# graph1 = [[0,1,0,1,1,0,0],
#           [1,0,1,0,1,0,0],
#           [0,1,0,1,0,1,0],
#           [1,0,1,0,0,0,1],
#           [1,1,0,0,0,1,0],
#           [0,0,1,0,1,0,1]
#           [0,0,0,1,1,1,0]]

# graph2 = [[0,1,0,1,0,0,1],
#           [1,0,1,0,1,0,0],
#           [0,1,0,1,0,1,0],
#           [1,0,1,0,0,0,1],
#           [0,1,0,0,0,1,1],
#           [0,0,1,0,1,0,1]
#           [0,0,0,1,1,1,0]]

map = [0 for i in range(n)]
temp = isomorphic_check(0, map)
if temp == True:
    print("isomorphic")
else:
    print("Non isomorphic")
```

Output:

Test case 1:

harshavaidhyam@Harshas-MacBook-Pro Project 3 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 54744 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3/problem3.py

[1, 3, 4, 0, 2]
Isomorphic

Test case 2:

harshavaidhyam@Harshas-MacBook-Pro Project 3 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 54757 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3/problem3.py

Non isomorphic

Test case 3:

harshavaidhyam@Harshas-MacBook-Pro Project 3 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 54757 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/Project\ 3/problem3.py

Non isomorphic

Time complexity:

$O(N^2)$