Assessment – 1

MCSE 502L Design and analysis of Algorithm (Lab Component)

1. Explain how graph coloring is done. Implement it for the following graph. Paste the code as well as the output.

Solution 1 :

Graph coloring :

In graph theory, graph coloring is defined as the procedure of assignment of colours to each vertex of a graph G such that no adjacent vertices get same colour. It is a way of coloring the vertices of a graph such that no two adjacent vertices get same colour. This is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges are of the same color. Graph coloring problem is a NP complete problem.

chromatic number :

Chromatic number is define as the minimum number of colors required to color a graph such that no adjacent vertices get same color. For example, the following can be colored minimum 4 colors.

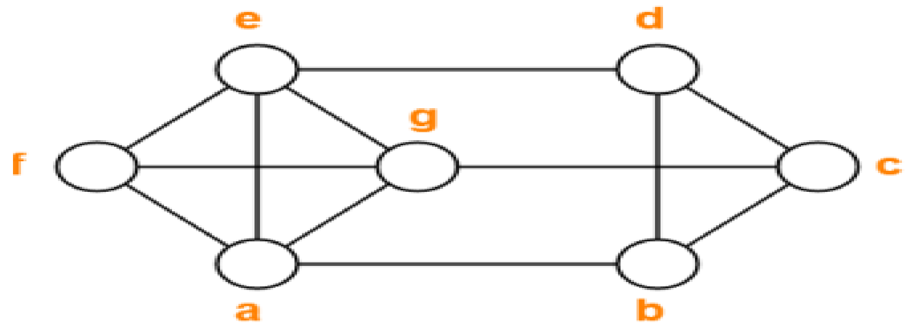
Procedure to Color a Graph :

Step 1 − Arrange the vertices of the graph in order.

Step 2 − Initially Choose the first vertex and color it with the very first color.

Step 3 − Choose the next vertex and color it with the minimum numbered color that has not been colored on any vertices adjacent to it. If all the adjacent vertices are colored with this color, assign a new color to it.

Step 4 - Repeat this step 3 until all the vertices are colored.

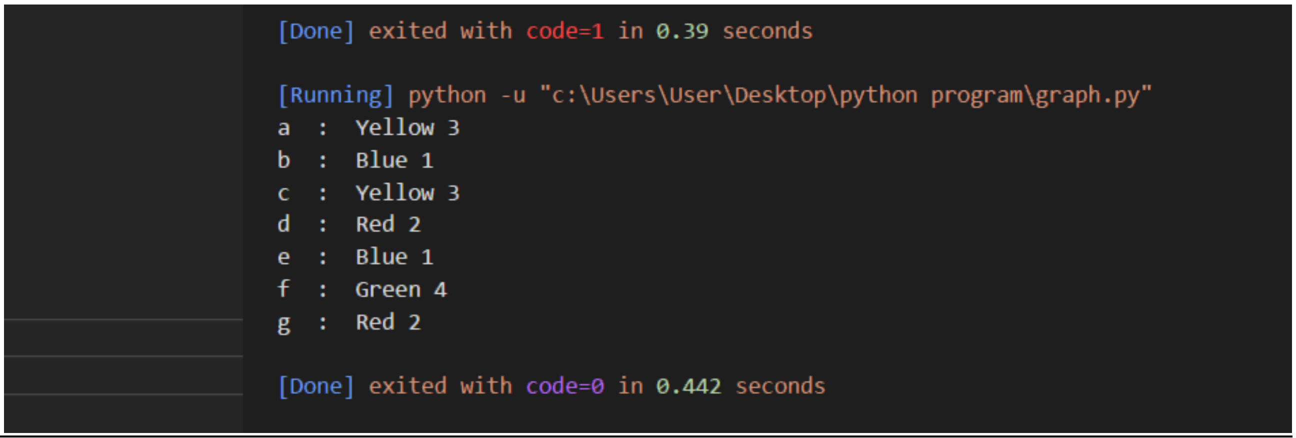


Code :

```
G = [[ 0, 1, 1, 1, 0, 0, 0], [ 1, 0, 1, 1, 1, 0, 0], [ 1, 1, 0, 1, 0, 0, 1],  [ 1, 1, 1, 0, 0, 1, 0], [ 0, 1, 0, 0,
0, 1, 1],  [ 0, 0, 0, 1, 1, 0, 1], [0, 0, 1, 0, 1, 1, 0]]
vertice = "fegadbc"
t_={}
for i in range(len(G)):
  t_[vertice[i]] = i
deg =[]
for i in range(len(G)):
  deg.append(sum(G[i]))
color = {}
for i in range(len(G)):
  color[vertice[i]]=["Blue 1","Red 2","Yellow 3","Green 4"]
Node=[]
indeks = []
for i in range(len(deg)):
  _max = 0
  j = 0
  for j in range(len(deg)):
    if j not in indeks:
      if deg[j] > _max:
        _max = deg[j]
        idx = j
  indeks.append(idx)
  Node.append(vertice[idx])
theSolution={}
for n in Node:
  setTheColor = color[n]
  theSolution[n] = setTheColor[0]
  adjacentNode = G[t_[n]]
  for j in range(len(adjacentNode)):
    if adjacentNode[j]==1 and (setTheColor[0] in color[vertice[j]]):
      color[vertice[j]].remove(setTheColor[0])

for t,w in sorted(theSolution.items()):
  print(t," : ",w)
```

Output :



```
[Done] exited with code=1 in 0.39 seconds

[Running] python -u "c:\Users\User\Desktop\python program\graph.py"
a  :  Yellow 3
b  :  Blue 1
c  :  Yellow 3
d  :  Red 2
e  :  Blue 1
f  :  Green 4
g  :  Red 2

[Done] exited with code=0 in 0.442 seconds
```

2. Scheduling: A set of jobs are given that need to be assigned to time slots. Each job needs one such slot. Jobs can be scheduled in any order, but pairs of jobs may be in conflict in the sense that they may not be assigned to the same time slot, for example, because they both rely on a shared resource.
Explain with an example how graph coloring can be useful for the above mentioned scheduling problem (in question no 2).

Solution 2 :
Job Sequencing With Deadlines-

Job Sequencing with Deadlines is the technique of scheduling jobs with deadlines on a single processor.Here-

- Several tasks are assigned to you.
- Every job has a set completion date and a profit attached to it.
- Only when a job is finished by the deadline is the profit awarded.
- All of the jobs can be completed by a single processor.
- A processor completes a task in one unit of time.

Solution Approach:

- A manageable solution would be a subset of tasks, each of which is accomplished by the deadline.
- The profit from all the jobs in the subset as a whole would represent the value of the workable solution.
- An effective solution to the issue that yields the greatest profit would be the ideal one.
- Greedy Algorithm: This algorithm is used to choose the next task in order to achieve the best result.

The job sequencing problem is always resolved optimally by the greedy approach given below.

Step 1 is to rank all of the offered jobs according to diminishing profit.

Step 2: Verify the maximum deadline's value. Create a Gantt chart with the maximum time equal to the maximum deadline.

Step-03 Take each task in turn. Put the task as far away from 0 on the Gantt chart as you can to insure that it is finished before the due date.

Problem-

Given the jobs, their deadlines and associated profits as shown-

Jobs	**J1**	**J2**	**J3**	**J4**	**J5**	**J6**
Deadlines	5	3	3	2	4	2
Profits	200	180	190	300	120	100

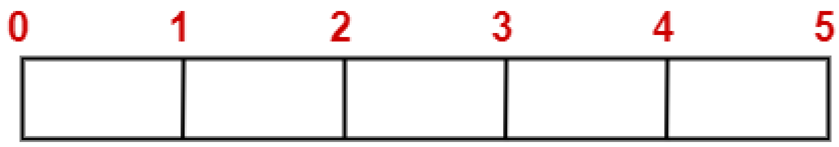
Solution-

Step-01: Sort all the given jobs in decreasing order of their profit-

Jobs	**J4**	**J1**	**J3**	**J2**	**J5**	**J6**
Deadlines	2	5	3	3	4	2
Profits	300	200	190	180	120	100

Step-02:
Maximum deadline value is 5.

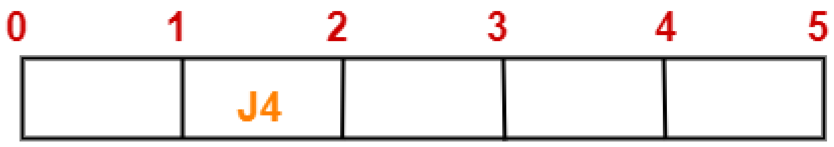
So, as indicated, build a Gantt chart with a maximum time of 5 units.



Gantt Chart

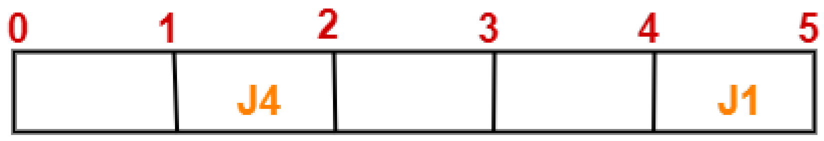
Now, in the order they occur in Step-01, we tackle each job individually.
• We position the task on the Gantt chart as far away from 0 as we can.

Step-03:
Since job J4's deadline is 2, we enter it in the first available cell before that deadline as-



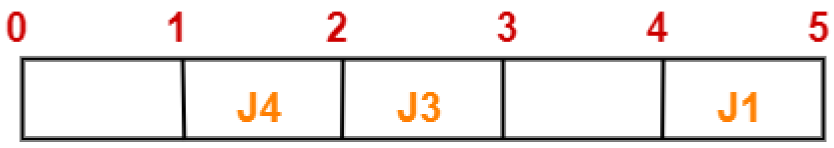
Step-04:

Job J1 is selected, and since it has a deadline of 5, it is inserted into the first available cell prior to that deadline as-

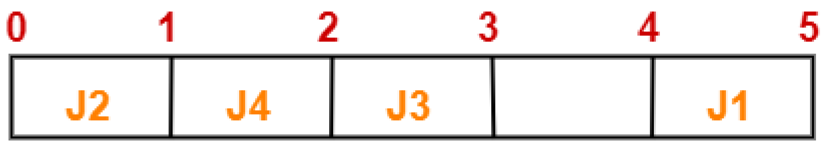


Step-05:

Since job J3's deadline is 3, we enter it in the first available cell before that deadline as-

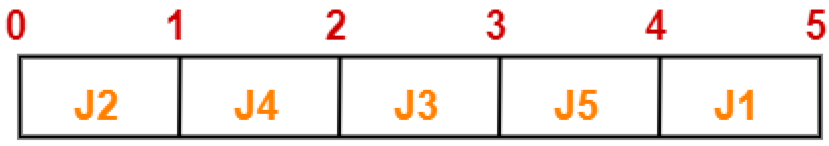


Step-06:
We choose job J2, and since it has a deadline of 3, we put it in the first available empty cell.
• Since the second and third cells are already full, job J2 is placed in the first cell as-



Step-07:

Job J5 is now being considered. Since it has a deadline of 4, we put it in the first empty cell before that deadline as-



- The only open job at the moment is J6, which has a deadline of 2.
- Every available slot prior to deadline 2 has already been taken.
- Job J6 cannot be finished as a result.

The optimal schedule is-

J2, J4, J3, J5, J1

This is the required order in which the jobs must be completed in order to obtain the maximum profit.
Due to job J6's inability to be completed by the deadline, all of the jobs were not completed according to the best possible timetable.
Maximum earned profit = Sum of profit of all the jobs in optimal schedule
= Profit of job J2 + Profit of job J4 + Profit of job J3 + Profit of job J5 + Profit of job J1
= 180 + 300 + 190 + 120 + 200
=990

Algorithm:

```
Job-Sequencing-With-Deadline (D, J, n, k)
D(0) := J(0) := 0
k := 1
J(1) := 1   // means first job is selected
for i = 2 … n do
  r := k
  while D(J(r)) > D(i) and D(J(r)) ≠ r do
    r := r – 1
```

```
if $D(J(r)) \le D(i)$ and $D(i) > r$ then
  for $l = k \dots r + 1$ by -1 do
    $J(l + 1) := J(l)$
    $J(r + 1) := i$
    $k := k + 1$
```

APPLYING THE CONCEPT OF GRAPH COLORING THEORY IN WORKER SCHEDULING PROBLEM :

Graph theory is often utilised in scheduling issues such as the scheduling of nurses, drivers, and other professions. With no conflicts, the graph theory vertex colouring notion is helpful in properly allocating resources like people, machines, and cars. Due to time constraints, we are allowing workers with varying levels of expertise to work together during the same shift. This allows the senior most workers to address any issues as they arise and ensures that everyone's tasks are completed on schedule. Let's assume that there are 30 workers in an industry in order to solve this scheduling problem utilising graph theory. According to the previous description, any senior staff member must define the schedule throughout three shifts. Suppose that the eight senior-most employees were sorted into two groups, X1 and X2, respectively. The group of employees is divided by the senior-most member of staff in the manner shown below, with knowledgeable employees making up each group. The workers are identified as w1, w2, w3, w4, w5,... w30. The group of 10 employees is being divided. With the help of the concept of suitable k-coloring provided above, we may divide the group of workers into three disjoint subgroups that correspond to the three areas.

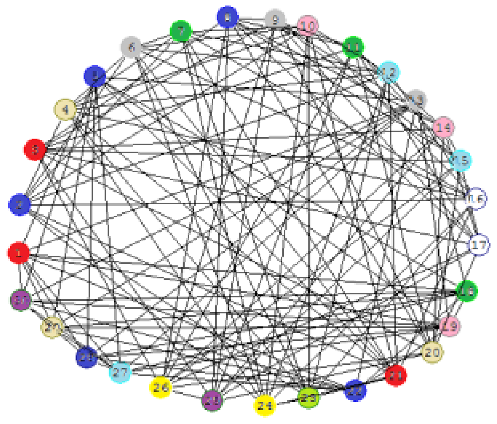
The first group X1 = {w1, w5, w13, w18, w19, w22, w27, w28, w29, w30}

Second group X2 = {w2, w4, w6, w7, w21, w23, w24, w16, w12, w10}

Third group X3 = {w3, w8, w9, w11, w14, w15, w17, w20, w25, w26}

A matrix was made for the workers by utilising the aforementioned sets. The matrixes measured 30 x 30. Worker names are assumed to be I and j in that matrix. The ij th entry is then placed in accordance with the settings mentioned above.

If any two employees are in the same group, then the entry for ij is equal to either 1 or 0. (For convenience, let's refer to w1, w2, w3, and w30 as just 1, 2, 3, and 4 through 30.) Adjacency matrix for the worker set.



The I j) th item is 1 if I j since I and j are next to each other. We can denote A (G) as A because the I j) element of A (G) is 0 for I = 1, 2, 3.......n. The adjacency matrix for the group of employees is shown in the matrix below and indicates which employees can work together during the same shift. Specifically, the adjacency matrix's 1 indicates that the points can be connected (vertices). 0 means that we are unable to connect those places. Workers are used as the vertices of a graph that is created using the matrix . (points). The related vertices are combined using edges if two workers are in the same group based on the disjoint sets X1, X2, and X3 mentioned above.

The aforementioned matrix is known as an adjacency matrix in graph theory. A m x m matrix with the number of edges connecting vertex I and vertex j as the ij-th member. The graph's vertices were coloured in the next section. According to the preceding definition of vertex colouring in graph theory, adjacent vertices may be coloured differently.
regarding the meaning of vertex colouring. After the graph was drawn, nearby vertices were given various colours. The information below demonstrates how the greedy approach is used to determine the colours for the vertices (points).
Point 1 2 3 4 5 6 7 8 9 10
Color 1 2 1 8 2 3 4 2 3 5
Point 11 12 13 14 15 16 17 18 19 20
Color 4 6 3 5 6 7 7 4 5 8
Point 21 22 23 24 25 26 27 28 29 30
Color 1 2 9 10 9 10 6 7 8 9
NetWork : wsp grin.next, Type : undirNet ,Number of Points : 30,Number of Edges : 135 ,Chromatic Number = 10

Group of workers details after applying vertex coloring techniques :

Group	Names of the workers
X1	w1, w5, w13, w18, w19, w22, w27, w28, w29, w30
X2	w2, w4 , w6, w7, w21, w23, w24, w16, w12, w10
X3	w3, w8, w9, w11, w14, w15, w17, w20, w25, w26

CONCLUSION ::

We applied the concept of vertex coloring to schedule workers facing continuously operated organizations. Here, we divided the workers into her three groups using the appropriate k-coloring definitions and theorems, and colored the graph using the adjacencies we created. According to the table above, w1 and w27 are members of the same group X1, so w1 and w27 can work together in the same shift. But w1 and w2 cannot work together because they belong to different groups (w1 € X1 and w2 € X2). Multiple groups can work together in the same shift, but workers must belong to the same group. Otherwise, some restrictions may be violated. If you have a scheduling problem, you can apply graph theory vertex coloring to solve the organizational scheduling problem.