SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL504 **DATE: 26/10/2021**

COURSE NAME: Artificial Intelligence Laboratory **CLASS:TY-IT**

EXPERIMENT NO.05

CO/LO: Formulate the problem as a state space and select appropriate technique from blind, heuristic, or adversarial search to generate the solution.

AIM / OBJECTIVE: To implement the constraint satisfaction problem.

DESCRIPTION OF EXPERIMENT:

- Student should implement the state space for a CSP problem [graph coloring].
- The traversal path for obtaining the solution should be displayed.

Explanation/Solutions (Design):

Constraint satisfaction problem:

A constraint satisfaction problem (CSP) consists of

- a set of variables,
- a domain for each variable, and
- a set of constraints.

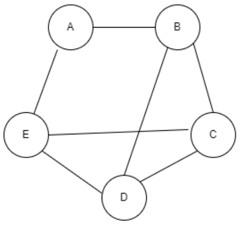
The aim is to choose a value for each variable so that the resulting possible world satisfies the constraints; we want a model of the constraints.

A finite CSP has a finite set of variables and a finite domain for each variable

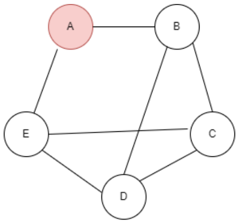
Graph coloring:

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 same color.
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 minimum 2 colors.

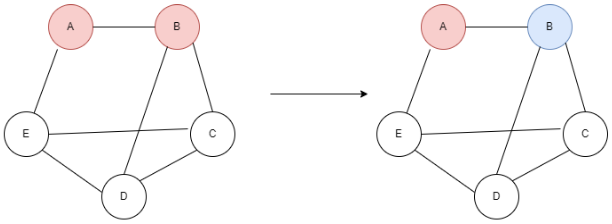
Example :



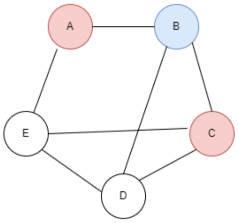
Step 1)



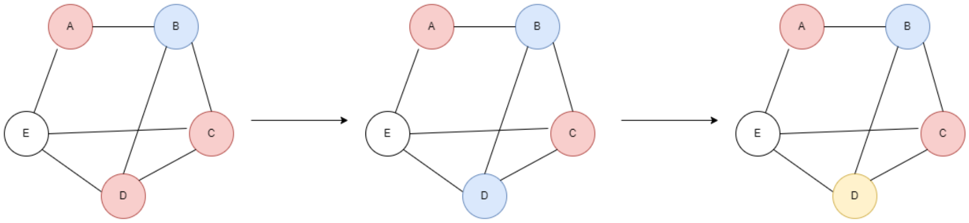
Step 2)



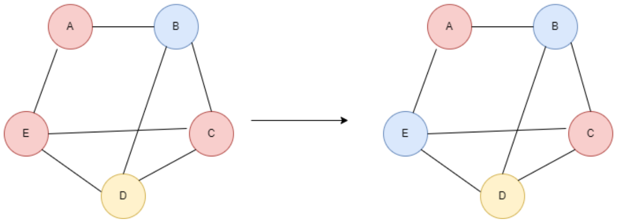
Step 3)



Step 4)

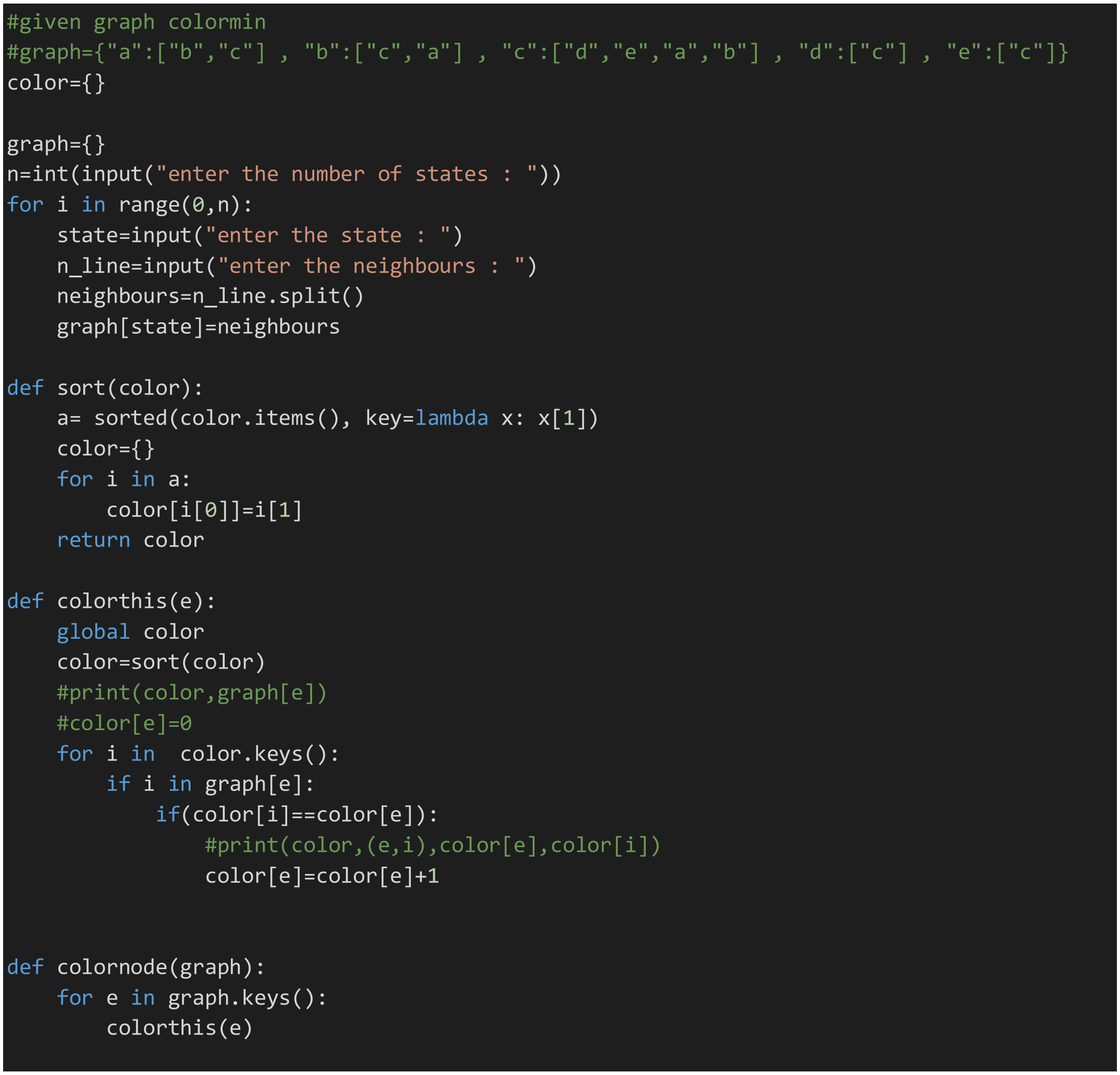


Step 5)



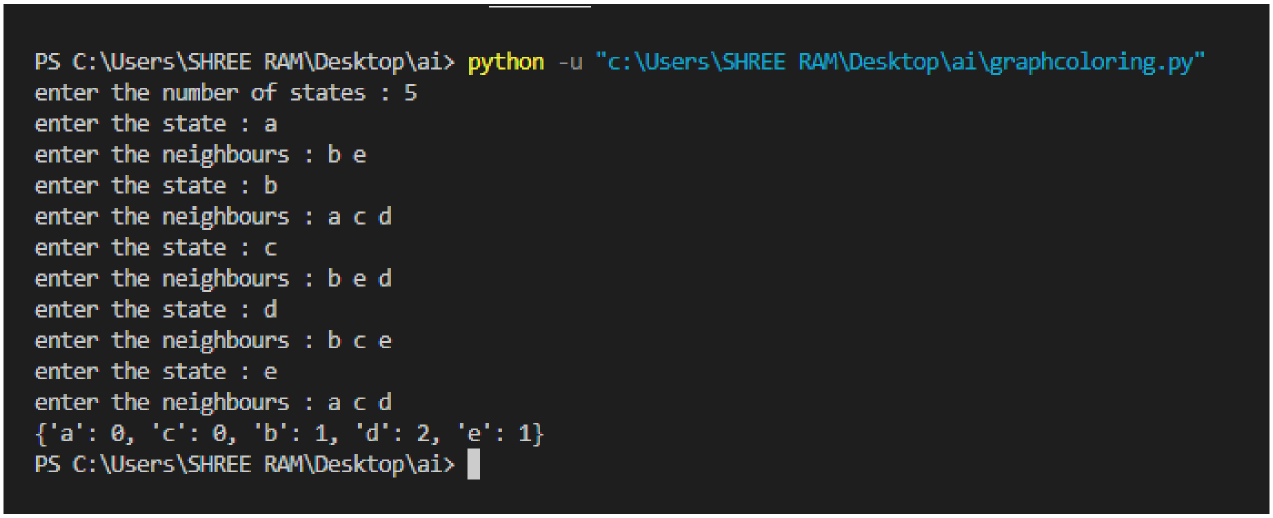
```
Algorithm graphColoring(graph):
    For all nodes in graph do:
        Color the current node with color “0”
        Check for all neighbours of current node:
            If colored same as current node then increment color by 1
```

Source code:



```
#given graph colormin
#graph={"a":["b","c"] , "b":["c","a"] , "c":["d","e","a","b"] , "d":["c"] , "e":["c"]}
color={}

graph={}
n=int(input("enter the number of states : "))
for i in range(0,n):
    state=input("enter the state : ")
    n_line=input("enter the neighbours : ")
    neighbours=n_line.split()
    graph[state]=neighbours

def sort(color):
    a= sorted(color.items(), key=lambda x: x[1])
    color={}
    for i in a:
        color[i[0]]=i[1]
    return color

def colorthis(e):
    global color
    color=sort(color)
    #print(color,graph[e])
    #color[e]=0
    for i in  color.keys():
        if i in graph[e]:
            if(color[i]==color[e]):
                #print(color,(e,i),color[e],color[i])
                color[e]=color[e]+1


def colornode(graph):
    for e in graph.keys():
        colorthis(e)
    print(color)
    #print("given graph can be colored in ",max(color.values)+1 ," colors")

#print(graph.keys())
colornode(graph)
```

Output:



```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\graphcoloring.py"
enter the number of states : 5
enter the state : a
enter the neighbours : b e
enter the state : b
enter the neighbours : a c d
enter the state : c
enter the neighbours : b e d
enter the state : d
enter the neighbours : b c e
enter the state : e
enter the neighbours : a c d
{'a': 0, 'c': 0, 'b': 1, 'd': 2, 'e': 1}
PS C:\Users\SHREE RAM\Desktop\ai>
```

CONCLUSION:

Thus we have implemented graph coloring constraint satisfaction problem(csp).