Artificial Intelligence  
Lab Exercise 2  
Graph Coloring Problem

short line

Ashwin Prakash  
RA1911026010048

**The Graph Coloring Problem:**

Graph colouring is one of the most important concepts in graph theory and it has a huge number of applications in daily life. Various colouring methods are available and can be used on a requirement basis.

The proper colouring of graph G is the colouring of the vertices and edges with a minimal number of colours such that no two vertices should have the same colour. The minimum number of colours is called the chromatic number χ(G) and the graph G is called a properly coloured graph.

**Aim:**

To solve Graph Colouring Problem.

**Algorithm:**

**Greedy Algorithm**

1. Color the first vertex with the first colour.

2. Do the following for the remaining V-1 vertices.

a) Consider the currently picked vertex and colour it with the

lowest numbered colour that has not been used on any previously

coloured vertices adjacent to it. If all previously used colours

appear on vertices adjacent to v, assign a new colour to it.

**Program:**

**Construction of Graphs**

def greedyColoring(adj, V):

    result = [-1] \* V

    result[0] = 0

    available = [False] \* V

    for u in range(1, V):

        for i in adj[u]:

            if (result[i] != -1):

                available[result[i]] = True

        cr = 0

        while cr < V:

            if (available[cr] == False):

                break

            cr += 1

        result[u] = cr

        for i in adj[u]:

            if (result[i] != -1):

                available[result[i]] = False

    for u in range(V):

        print("Vertex", u, " =====>  Color", result[u])

def addEdge(adj, v, w):

    adj[v].append(w)

    adj[w].append(v)

    return adj

if \_\_name\_\_ == '\_\_main\_\_':

    g1 = [[] for i in range(5)]

    g1 = addEdge(g1, 0, 1)

    g1 = addEdge(g1, 0, 2)

    g1 = addEdge(g1, 1, 2)

    g1 = addEdge(g1, 1, 3)

    g1 = addEdge(g1, 2, 3)

    g1 = addEdge(g1, 3, 4)

    print("Graph A")

    greedyColoring(g1, 5)

    g2 = [[] for i in range(5)]

    g2 = addEdge(g2, 0, 1)

    g2 = addEdge(g2, 0, 2)

    g2 = addEdge(g2, 1, 2)

    g2 = addEdge(g2, 1, 4)

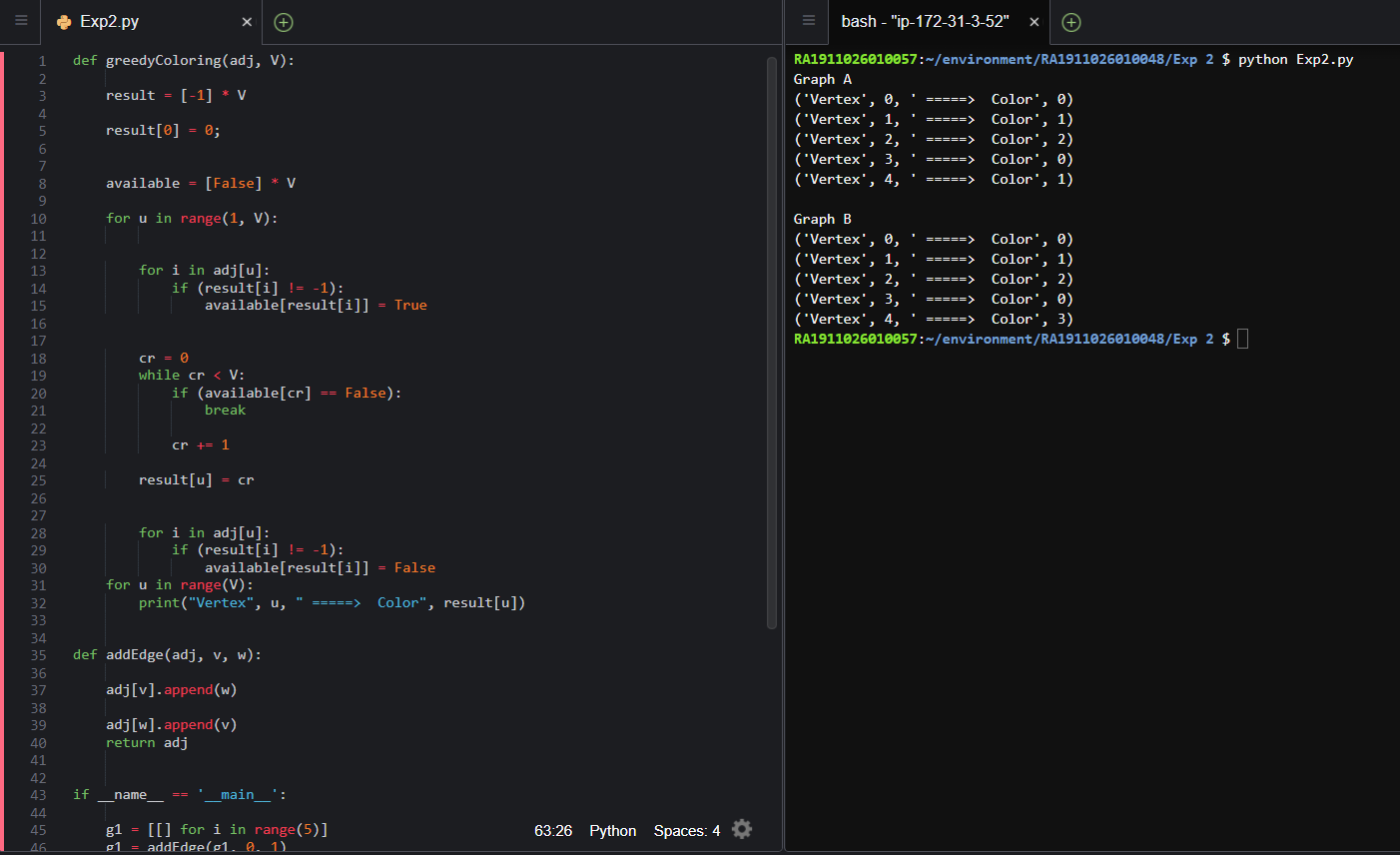
    g2 = addEdge(g2, 2, 4)

    g2 = addEdge(g2, 4, 3)

    print("\nGraph B")

    greedyColoring(g2, 5)

**Output:**

****

**Graph Coloring in Real-Life Scenarios**

To color 4 neighboring houses.

**Code:**

colors = ['Red', 'Blue', 'Orange', 'Green', 'Yellow']

houses = ['Worstershire House', 'The Shining', 'Paradise Villa', 'Creek Valley']

neighbors = {}

colors\_of\_houses = {}

neighbors['Creek Valley'] = ['The Shining', 'Paradise Villa']

neighbors['The Shining'] = ['Worstershire House', 'Paradise Villa', 'Creek Valley']

neighbors['Paradise Villa'] = ['Worstershire House', 'The Shining', 'Creek Valley']

neighbors['Worstershire House'] = ['The Shining', 'Paradise Villa']

def get\_neighbor(house, color):

    for neighbor in neighbors.get(house):

        color\_of\_neighbor = colors\_of\_houses.get(neighbor)

        if color\_of\_neighbor == color:

            return False

    return True

def get\_color\_for\_house(house):

    for color in colors:

        if get\_neighbor(house, color):

            return color

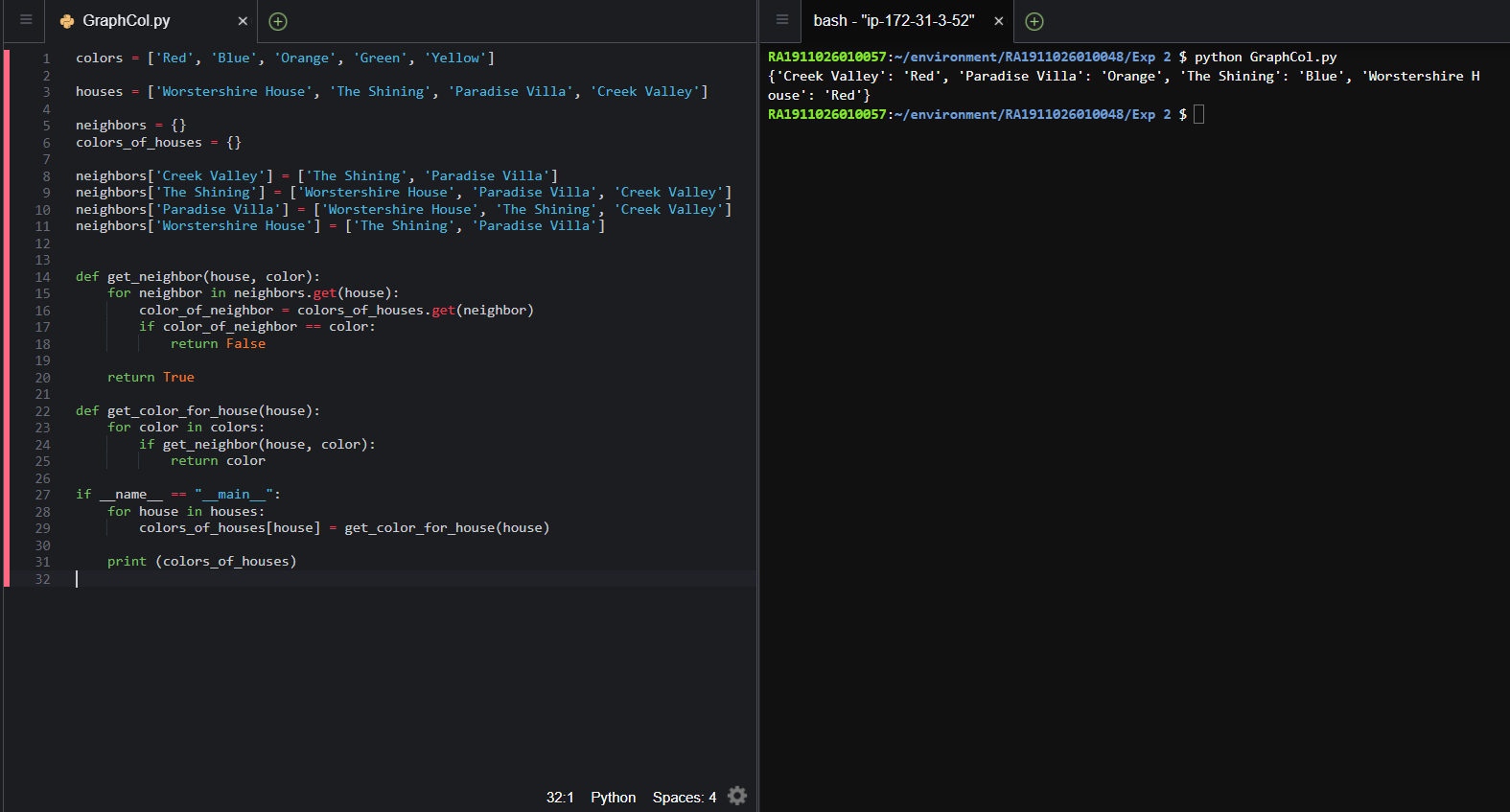
if \_\_name\_\_ == "\_\_main\_\_":

    for house in houses:

        colors\_of\_houses[house] = get\_color\_for\_house(house)

    print (colors\_of\_houses)

**Output:**

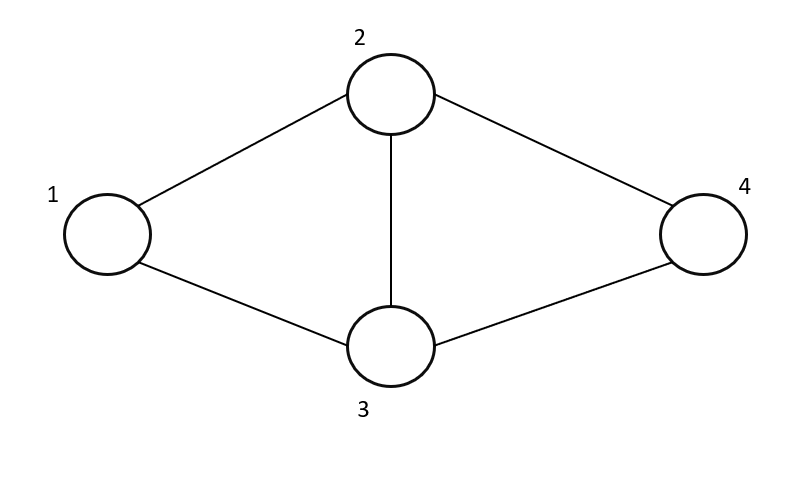
****

**Applications of Graph Colouring:**

* Map Coloring
* Scheduling the tasks
* Preparing Time Table
* Assignment
* Conflict Resolution
* Sudoku

**Observations:**

The chromatic number of a Special Graph χ(G) =n for odd vertices and χ(G)=2 for every vertex.



Here Worstershire House (1) and Creek Valley (4) are coloured red, Paradise Villa (3) is coloured orange and The Shining(2) is coloured blue.

Here the graph contains 4 vertices which denote states, with degrees 2,2,3,3. As vertex 2 and vertex 3 are connected hence they must be coloured with different colours. As there is no connection between 1 and 4 therefore they can be coloured with the same colour. This makes the χ(G)=3.

**Result:**

The graph colouring problem has been solved using a greedy algorithm and a real-life scenario has been discussed.