Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Spring , Year:2024), B.Sc. in CSE (Day)

LAB REPORT NO #02

Course Title: Artificial Intelligence Lab

Course Code: CSE - 316 Section: 213_D5

Lab Experiment Name: Constraint satisfaction problems .

Student Details

Name	ID
MD Dulal Hossain	213902116

Lab Date	:	**07– 05 - 2024**
Submission Date	:	**14– 05 - 2024**
Course Teacher's Name	:	**Sagufta Sabah Nakshi**

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: ………………………………… **Signature:.....................**

Comments:... **Date:..............................**

1. TITLE OF THE LAB EXPERIMENT

- To understand Constraint satisfaction problems CSP.
- To understand how to model Graph coloring problem as CSP.
- To understand how to represent a graph using adjacency list.
- To understand how Graph Coloring Algorithm works.

2. OBJECTIVES/AIM

To implement a greedy coloring algorithm to color the vertices of a given graph with the minimum number of colors, ensuring no adjacent vertices share the same color. The code iteratively assigns colors to vertices while considering neighboring vertices' colors, aiming for an efficient coloring solution.

3. PROCEDURE / ANALYSIS / DESIGN

Algorithm :

1. Initialize the data structures:
    a. Create an empty dictionary result to store the assigned colors for each vertex.
    b. Create an empty dictionary available to keep track of available colors for each vertex.
    c. Define a list of color names (e.g., ‘red’, ‘green’, etc.).
2. Assign the first color:
    a. Assign first color (let’s say color index 0) to first vertex (e.g., ‘Western Australia’).
3. Color the remaining vertices:
    a. For each remaining vertex u (from 1 to V-1, where V is the total number of vertices):
        i. Mark colors of adjacent vertices of u as unavailable in available dictionary.
        ii. Find the first available color for u that is not used by its adjacent vertices.
        iii. Assign the found color to vertex u.
4. Print the result:
    a. Print the assigned colors for each vertex.
5. Calculate the total number of unique colors used:
    a. Count the unique colors assigned to all vertices.

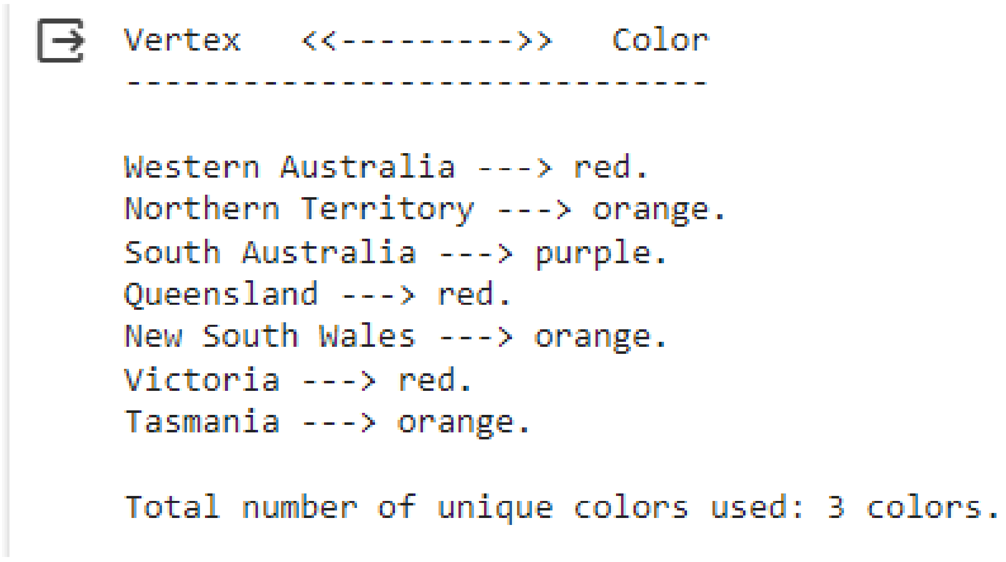
4. IMPLEMENTATION

Source Code :

```
graph = {
    'Western Australia': ['Northern Territory', 'South Australia'],
    'Northern Territory': ['Western Australia', 'South Australia', 'Queensland'],
    'South Australia': ['Western Australia', 'Northern Territory', 'Queensland', 'New South Wales',
'Victoria'],
    'Queensland': ['Northern Territory', 'South Australia', 'New South Wales'],
    'New South Wales': ['Queensland', 'South Australia', 'Victoria'],
    'Victoria': ['New South Wales', 'South Australia', 'Tasmania'],
    'Tasmania': ['Victoria']
}
def greedy_coloring(adj):
    result = {}
    available = {}
    Color_names = ['red','green','blue','yellow','purple','orange']
    for vertex in adj:
        result[vertex] = None
        available[vertex] = set(color_names)
    for vertex in adj:
        for neighbor in adj[vertex]:
            if result[neighbor] is not None:
                available[vertex].discard(result[neighbor])
        result[vertex] = available[vertex].pop()

    print("Vertex   <<--------->>   Color")
    print("------------------------------\n")
    for vertex, color in result.items():
        print(f"{vertex} ---> {color}.")
    unique_colors = len(set(result.values()))
    print(f"\nTotal number of unique colors used: {unique_colors} colors.\n")
greedy_coloring(graph)
```

5. TEST RESULT / OUTPUT



```
Vertex   <<--------->>   Color
------------------------------

Western Australia ---> red.
Northern Territory ---> orange.
South Australia ---> purple.
Queensland ---> red.
New South Wales ---> orange.
Victoria ---> red.
Tasmania ---> orange.

Total number of unique colors used: 3 colors.
```

Figure 1 : Output Successfully .

Explain :

- The regions are colored as follows:
    - ❖ Western Australia: red
    - ❖ Northern Territory: orange
    - ❖ South Australia: purple
    - ❖ Queensland: red
    - ❖ New South Wales: orange
    - ❖ Victoria: red
    - ❖ Tasmania: orange
- Only three unique colors (red, orange, purple) are useds.
- The total number of unique colors used is 3.

6. ANALYSIS AND DISCUSSION

The code applies a greedy coloring algorithm to color the vertices of a given graph. It efficiently assigns colors to vertices, ensuring no adjacent vertices share the same color. This approach is simple to implement and works well for small to medium-sized graphs. However, its greedy nature may not always produce the optimal coloring for larger and more complex graphs. Additionally, the algorithm's reliance on local decisions might lead to suboptimal solutions in certain scenarios. Despite its limitations, the code provides a practical and quick solution for graph coloring problems, suitable for various applications where exact optimality is not a strict requirement but rather a good heuristic solution suffices.