Q1:

(i) Algorithm:

Step 0: Iterate over each node.

Step 1: Go through the neighbouring nodes and mark the colours used.

Step 2: In the remaining colours, assign the first unused colour by the neighbours to the given node.

Step 3: Repeat steps 2 and 3 for each node.

(ii) Pseudo Code:

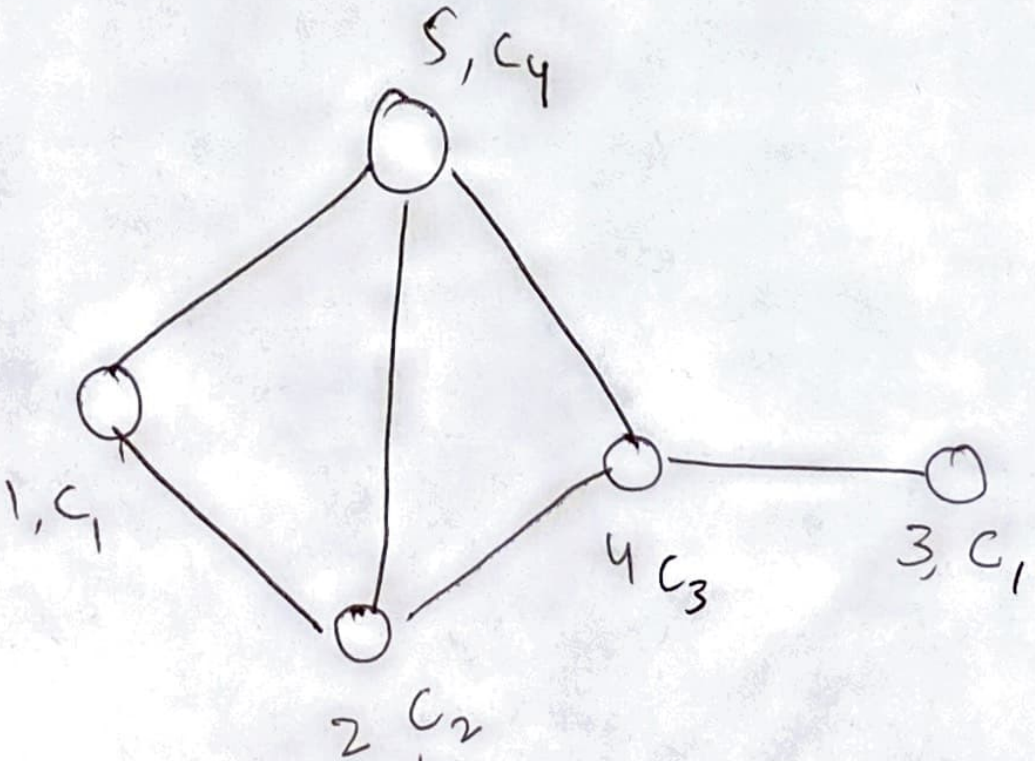
```
def greedy_graph_coloring(Graph, colors):
    nodes of Graph = list(Graph.nodes())
    for node in nodesof Graph:
        neighbours of Node = list(Graph(node).neighbours())
        mark_colors = [True] * len(colors)
    for neighbour in neighbours of Node:
        if Graph(neighbour)['colour'] != empty /uncolored
            mark_colour[Graph(neighbour['color']] = False

i = 0
while (mark_color[i] == False):
    i = i + 1
Graph(node)['colour'] = colors[i]

return Graph, colors.
```

Time Complexity:

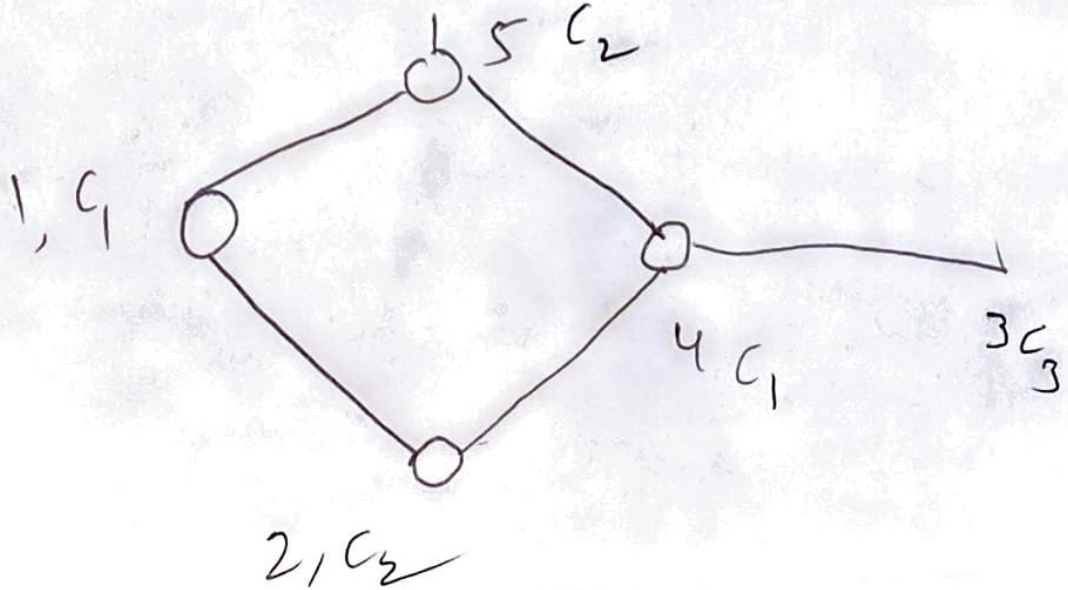
- Let N be the Number of vertices and M be the number of edges in the Graph.
- Iterating over all nodes of the graph is O(N)
- Iterating over all neighbours of a given node is O(max-degree of Graph)
- For finding the first ~~unmarked~~ unmarked color of a given node it is O(max-degree of Graph)
- Final Time Complexity = $O(N^2 + M)$.

Q2

- No, the algorithm is highly dependent on the order of the nodes chosen or their arrangement in the graph which may lead to solutions with more colors than an optimal answer.
- The upper bound of the colours is decided by the maximum degree (d) of the graph because if all the neighbours of the node maximum degree are colored with distinct colors (d) then we require one more color d+1 to color it and for any other node with degree less than or equal to d, we cannot require ~~more~~ more than d+1 colors. Hence, the upper bound on the required numbers of colours is maximum degree (d) of the graph + 1.



optimal for 1



Q4

(i) Take each cell in the game as a node of the Graph.

(ii) Draw edges from a node to its neighbouring nodes corresponding to its row, column and box.

(iii) Try to color all the nodes with atmost d+1 colours where d is the maximum degree of the Duko Graph.

(iv) Now run greedy graph coloring algorithm on the given Graph.

Q5

(i) The Welsh Powell Algorithm uses the same Greedy Algorithms approach but with an addition that it requires the nodes to be traversed in decreasing order of their degree. This makes it possible for us to eliminate the nodes with the greater number of neighbours hence conflicts as early as possible. Thus, this would result in a more optimal solution than Greedy.

(ii) In the worst case Greedy takes at max d+1 colours where d is the maximum degree of the Graph whereas Welsh Powell Algorithm takes at max $\max_i \min\{d+1, i\}$ which is mostly strictly less than 1.