Q5.4) For any graph G, does your algorithm always use exactly $\chi(G)$ colors? If so, explain why. If not, provide a graph G for which your algorithm requires more than $\chi(G)$ colors.

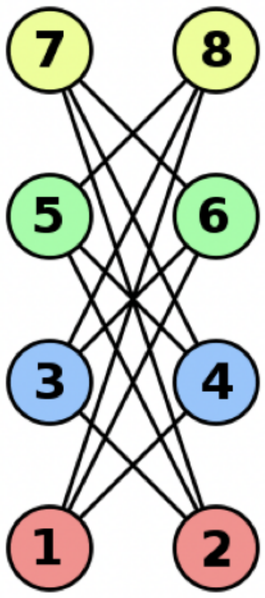
Greedy coloring considers the vertices of the graph in sequence and assigns each vertex its first available color, i.e., vertices are considered in a specific order v1, v2, … vn, and vi and assigned the smallest available color which is not used by any of vi's neighbors.
So, what we seek is a ***k-coloring*** of our graph with k as small as possible.

A simple greedy algorithm for creating a proper coloring is shown below. The basic idea is do a single pass through all vertices of the graph in some order and label each one with a numeric identifier. The procedure requires us to number consecutively the colors that we use, so each time we introduce a new color, we number it also. A vertex will labeled/colored with the lowest value that doesn't appear among previously colored neighbors.

However, Graph Coloring using Greedy algorithm doesn't always use the minimum number of colors possible to color a graph. For a graph of maximum degree x, greedy coloring will use at most x+1 color since a graph may contain cycles and you might not be aware of certain edges to nodes until you reduce your space to a few nodes and you're left with incompatible colors.

For instance, Greedy coloring does not work in cases of **crown graphs or bipartite graphs** if the vertices of a crown graph are presented to the algorithm in the order u0, v0, u1, v1, etc., then a greedy coloring uses n colors, whereas the optimal number of colors is two. Two greedy colorings of the same crown graph using different vertex orders can give different $\chi(G)$ colors.

The graph G' shown below generalizes to 2-colorable graphs with n vertices which means the chromatic number of the graph is 2 and requires 2 colors to color all the vertices, whereas using the greedy algorithm uses n/2 colors to color all the vertices which means the chromatic number is 4.



Explanation of the Greedy Algorithm (Counter Example)

The example provided above is of a crown graph (a complete bipartite graph) which does not provide the optimal results using Greedy algorithm as the algorithm traverses through the vertices in ascending order to proper color the graph. The order in which vertices are chosen and graph is k-colored is explained below.

1. Starting with vertex 1, the vertex is assigned the first color red.
2. Next, vertex 2 which does not share an edge with vertex 1 is also assigned the red color.
3. Next, vertex 3 shares an edge with 2, so it cannot be assigned color red. A new color blue is assigned to vertex 3 from the set of k-colors.
4. Vertex 4 which shares an edge with 1 and not 3. So it also assigned color blue.
5. Next, vertex 5 which shares an edge with 2 and 4. Since, it cannot be assigned color red and blue as it shares edges with previously-colored vertices, a new color green is assigned to vertex 5 from the set of k-colors.
6. Vertex 6 which shares an edge with 1 and 3. Since, it cannot be assigned color red and blue as it shares edges with previously-colored vertices, it is also assigned color green.
7. Next, vertex 7 which shares an edge with 6, 2 and 4. Since, it cannot be assigned color green, red and blue as it shares edges with previously-colored

vertices, respectively, a new color yellow is assigned to vertex 7 from the set of k-colors.

8. Vertex 8 which shares an edge with 5, 1 and 3. Since, it cannot be assigned color green, red and blue as it shares edges with previously-colored vertices, it is also assigned color yellow.

As we can see, the graph is k-colored and k is 4 which means that we require 4 colors to color all the vertices of the graph such that no two adjacent vertices share a color.

However, the optimal solution to color the graph is using 2 colors. Hence, the bipartite graph is 2-colored.

We can assign red color to vertices 1, 3, 5, 7 (left part of the problem) and blue color to vertices 2, 4, 6, 8 (right part of the problem). The condition still holds true in this case, as no adjacent vertices which share an edge will have same color. So, the solution for this problem is $\boldsymbol{\chi(G) = 2}$. However, greedy algorithm solved in $\boldsymbol{\chi(G) = 4}$.

Hence, Greedy algorithm does not always use minimum number of colors as number of colors used sometimes depend on the order in which the vertices are processed.

Nonetheless, suppose that d is the largest degree of any vertex in our graph, that is, all vertices have d or fewer edges attached, and at least one vertex has precisely d edges attached. As we go about coloring, when we color any particular vertex v, it is attached to at most d other vertices, of which some may already be colored. Then, there are at most d colors that we must avoid using. We use the lowest-numbered color which is not prohibited. That means that we use some color numbered d + 1 or lower, because at least one of the colors 1, 2, . . ., d+1 is not prohibited. So we never need to use any color numbered higher than d+1.

So, if d is the largest of the degrees of the vertices in a graph G, then G has a proper coloring with d+1 or fewer colors, i.e., the chromatic number of G is at most d+1.