**CSE2046/CSE2246**

**Assignment 2**

**Graph Coloring Problem**

**Problem:**

Graph Coloring (or Vertex Coloring) problem is defined as assignment of the smallest number of colors to the vertices of an undirected graph, such that no adjacent vertices are of the same color. A coloring using at most k colors is called a (proper) k-coloring. The smallest number of colors needed to color a graph G is called its chromatic number and is often denoted χ(G). The goal is to color all vertices of the graph, such that the number of colors used (k) is as close as possible to the optimal result, i.e. χ(G). Since this problem is NP-Hard, it is very difficult to find the optimal solution especially for large instances.

**Solution:**

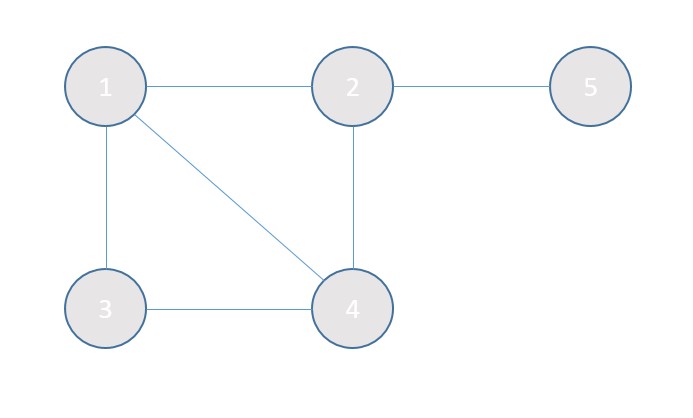
To solve the problem, we have used an improved version of the greedy algorithm. First we will examine the greedy algorithm to solve the Graph Coloring problem and then we will tell about our improved greedy approach which gives a better solution to the Graph Coloring problem by examples.

Greedy Approach:

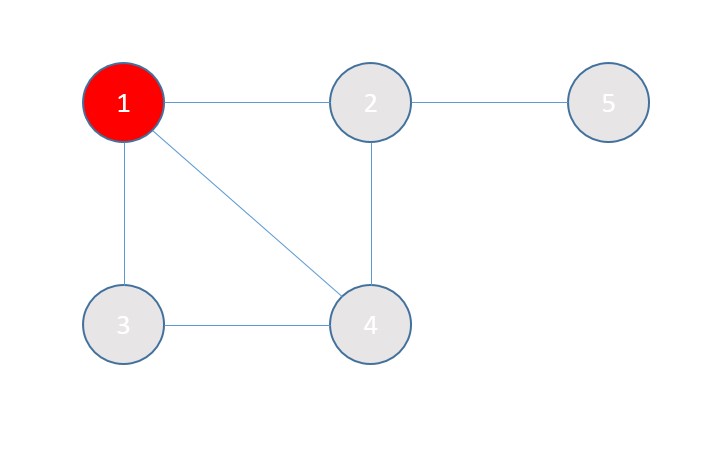
Greedy approach on graph coloring works as follows:

Take the first vertex and color with first color, then continue for the next V-1 vertex and assign the first possible color which is not assigned in the neighbors. (if not color remains generated a new color and assign it to the vertex) Time Complexity: **O(|V|2 + |E|)**

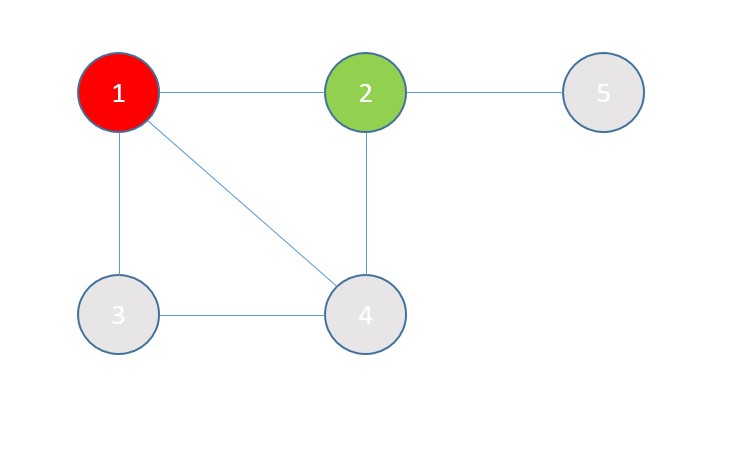
*Example 1:*



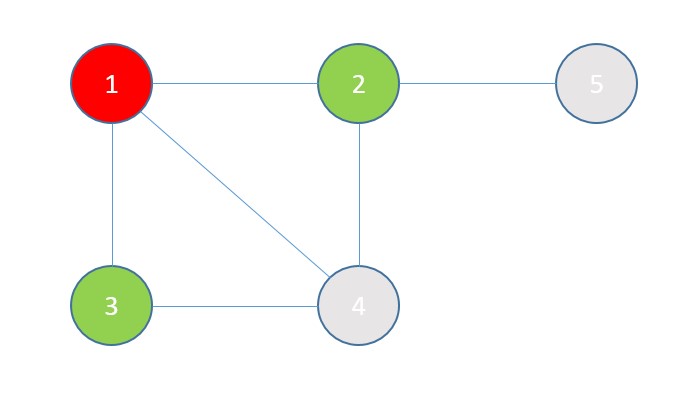
Assign vertex 1 as red color:



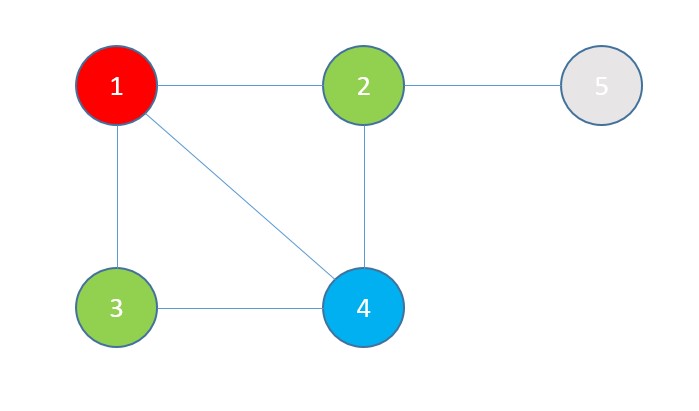
For vertex 2, because red is used on its neighbor vertex 1, assign new color green:



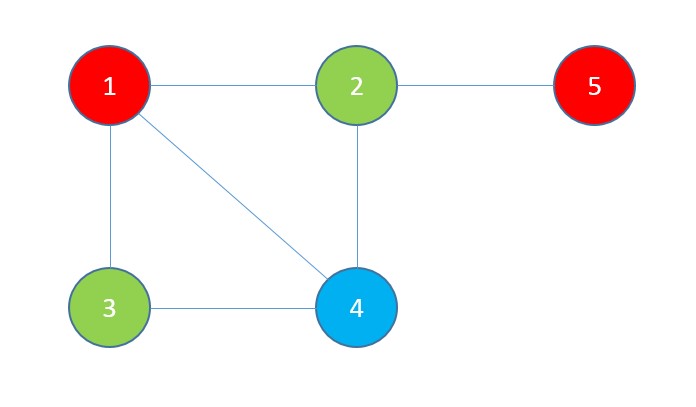
For vertex 3, its neighbor vertex 1 has already used red, so next color green is assigned.



For vertex 4, its neighbors, vertex 2 and 3 are green, vertex 1 is red, so assign a new color blue:



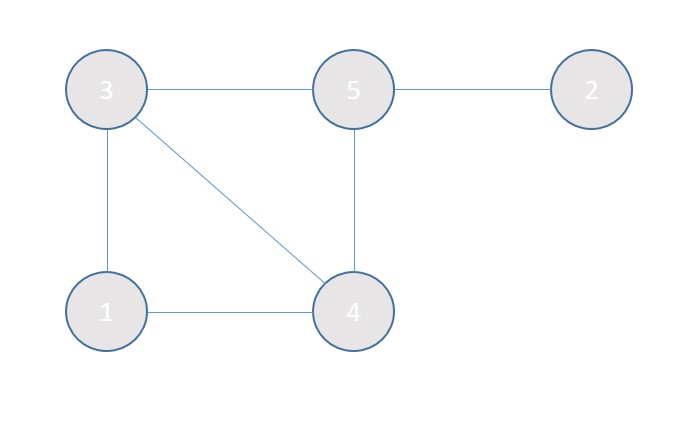
Last vertex 5, assigned first unused color red because its neighbor vertex 2 is assigned to green:



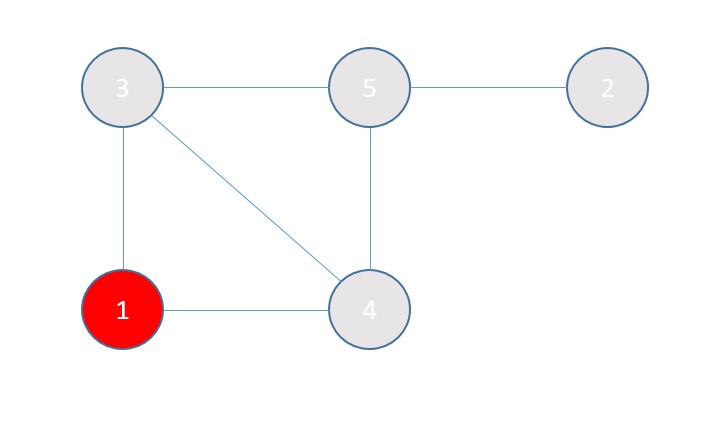
Totally **three** colors were used.

*Example 2:*

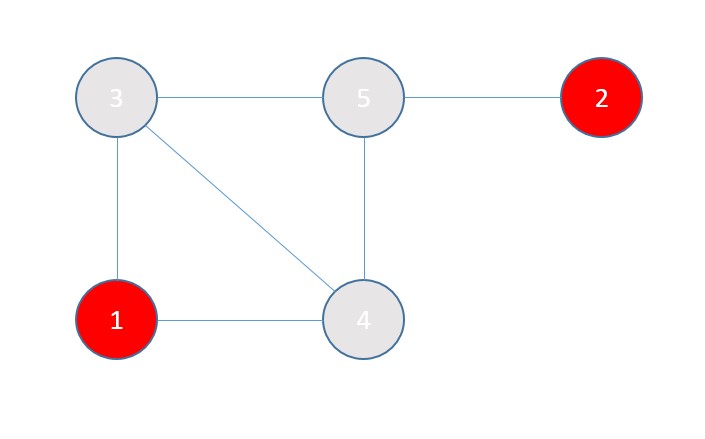
We change only some indexes of the vertices in the graph of the above example.



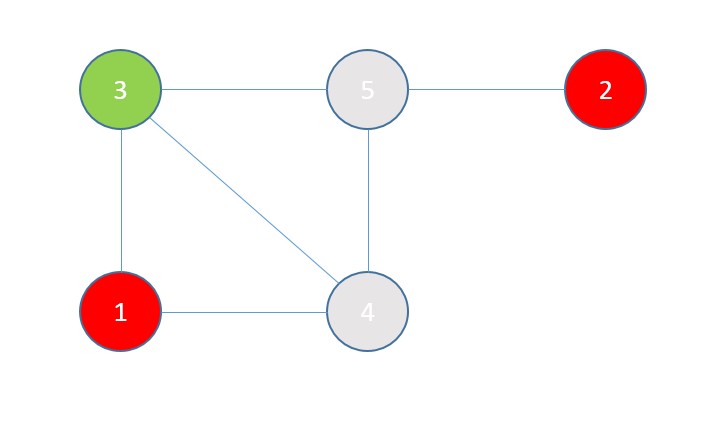
Assign vertex 1 as red color:



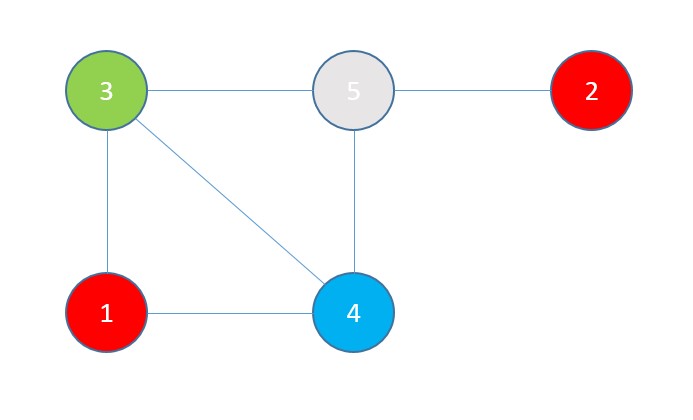
For vertex 2, assign first unused color red:



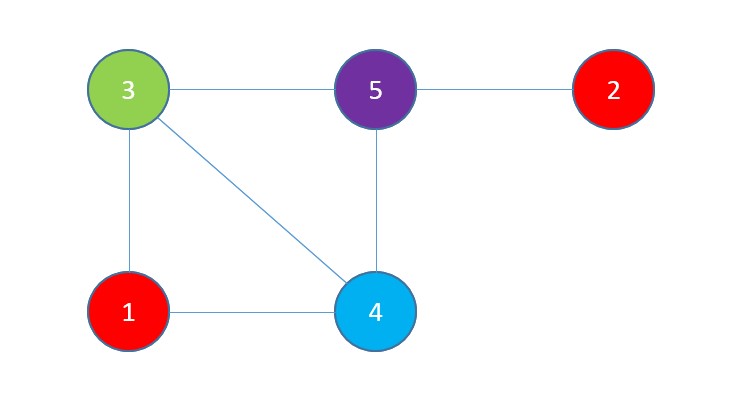
For vertex 3, its neighbor vertex 1 has already used red so the next color green is assigned.



For vertex 4, its neighbors, vertex 1 has used red, vertex 3 has used green, so assign a new color blue:



Last vertex 5, it neighbors uses red, green, blue, so assigned to a new color purple:



Totally **four** colors were used.

As we have seen, the order of the nodes which we are applying differs the result in a greedy approach. To get better results we have made an improvement on the greedy algorithm.

Improved greedy approach:

First order the node indexes according to the degree of the nodes (number of connections they have made) in descending order. After that apply the greedy approach with the new order of the indexes.

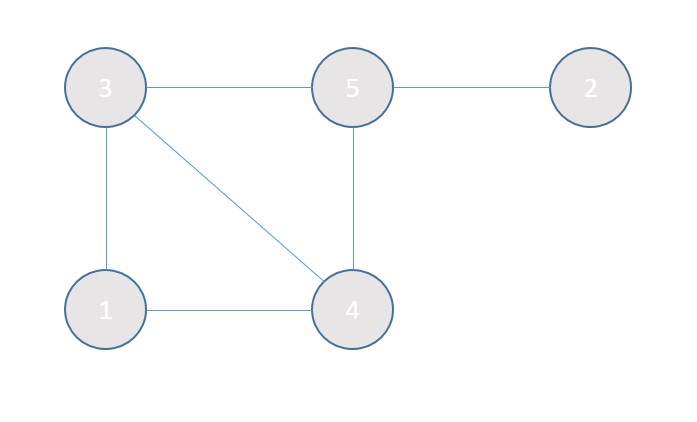
Time Complexity:

O(|V|log|V|) for sorting with quick sort or heap sort, or O(|V|2) with bubble sort.

O(|V|2 + |E|) for coloring with greedy approach

|  |  |
| --- | --- |
| Vertex | Degree |
| 1 | 2 |
| 2 | 1 |
| 3 | 3 |
| 4 | 3 |
| 5 | 3 |

Worst case: O(|V|log|V|) + O(|V|2 + |E|) = **O(|V|2 + |E|)** Let’s apply new approach to the above example:

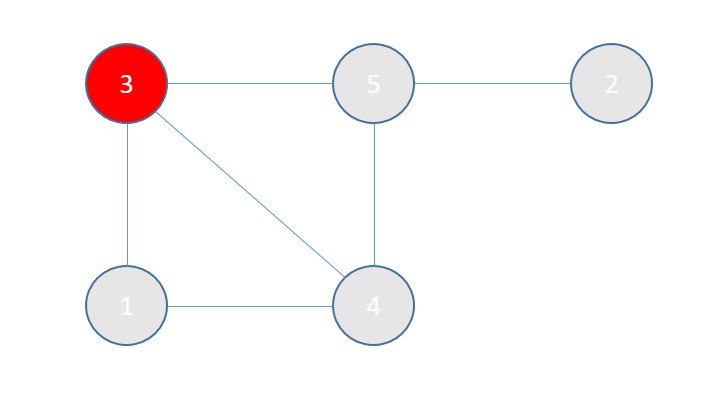


Nodes’ degrees are:

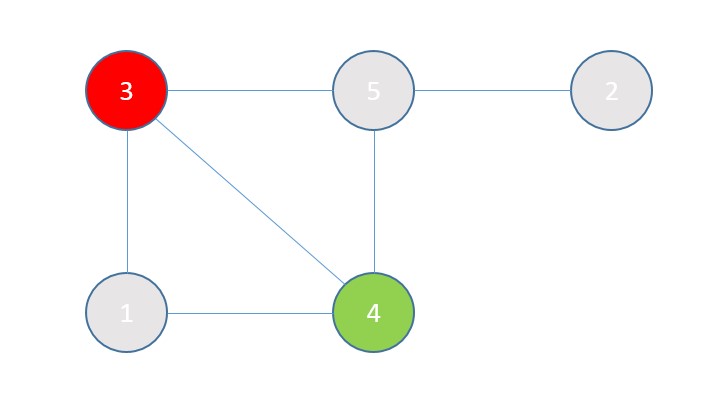
When we order them in descending order:

|  |  |
| --- | --- |
| Vertex | Degree |
| 3 | 3 |
| 4 | 3 |
| 5 | 3 |
| 1 | 2 |
| 2 | 1 |

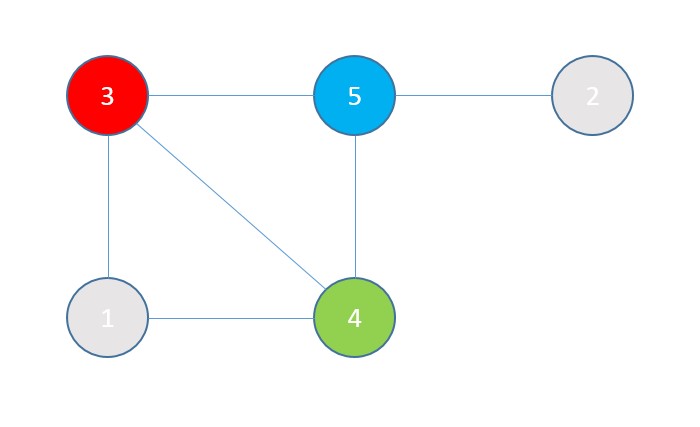
So we will apply greedy approach in this order: 3, 4, 5, 1, 2 Assign for first vertex 3 as red color:



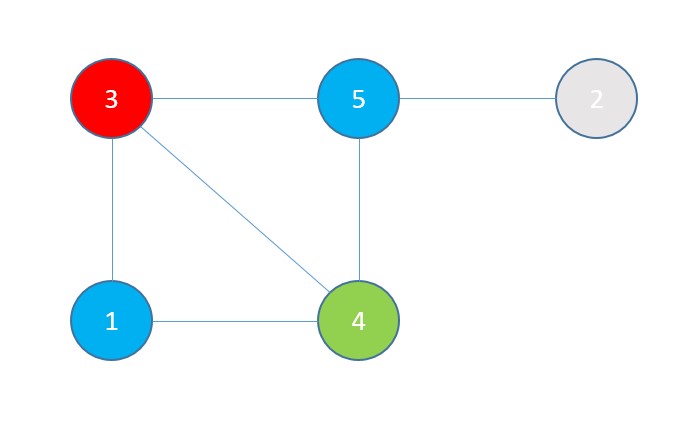
For next vertex 4, assign a new color green, because its neighbor vertex 3 is using red:



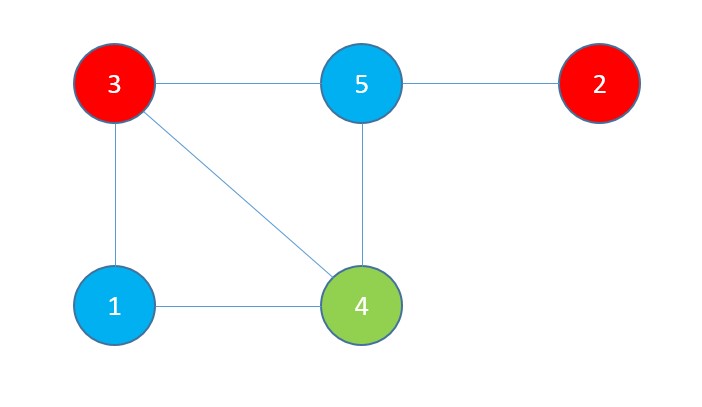
For next vertex 5, its neighbor vertex 3 has used red, and vertex 4 green, so new color blue is assigned to it:



For next vertex 1, its neighbor vertex 3 has used red, and vertex 4 green, so color blue is assigned to it:



Last vertex 2, it neighbor vertex 5 uses blue, so assigned to red:



Totally **three** colors used which is one less than using the basic greedy approach.