Graph Coloring (Vertex Coloring) – Part 2

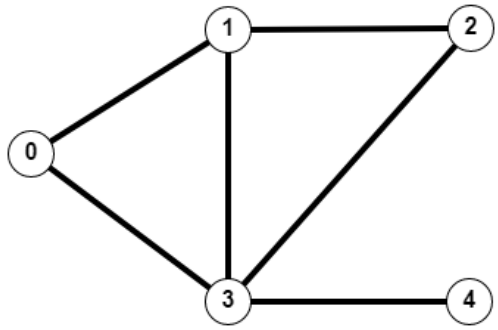
Approach

We will start with coloring vertex 0 (V_0), and then try to color remaining vertices one by one,

While trying to color a vertex we will start with first color, and if its not

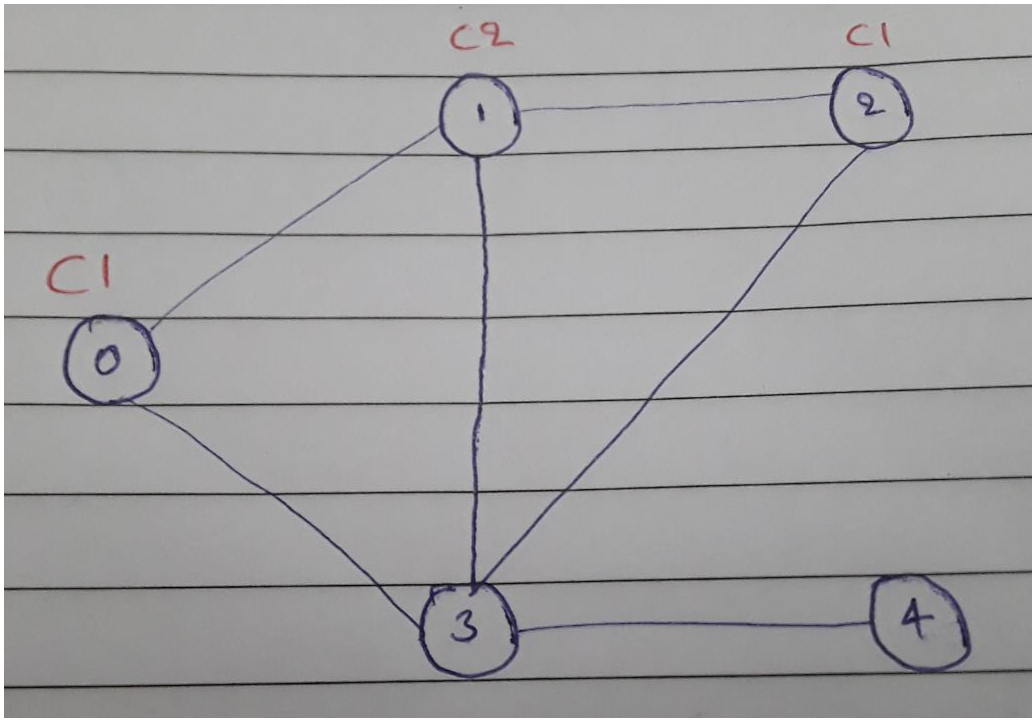
feasible to color the vertex with first color we will try the second color and so on.

While coloring a vertex, we will always make sure that, no adjacent vertices have the same color.

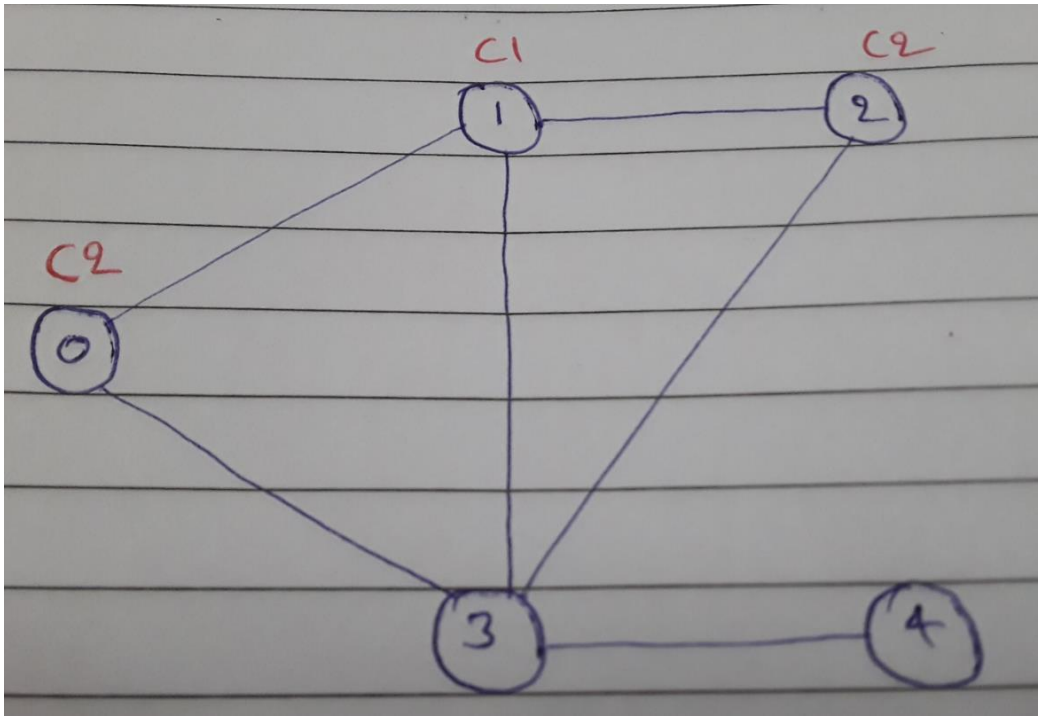



```
int[][] adjacencyMatrix = new int[][] {
                    { 0, 1, 0, 1, 0 },
                    { 1, 0, 1, 1, 0 },
                    { 0, 1, 0, 1, 0 },
                    { 1, 1, 1, 0, 1 },
                    { 0, 0, 0, 1, 0 } };
```

Vertex Coloring with 2 colors

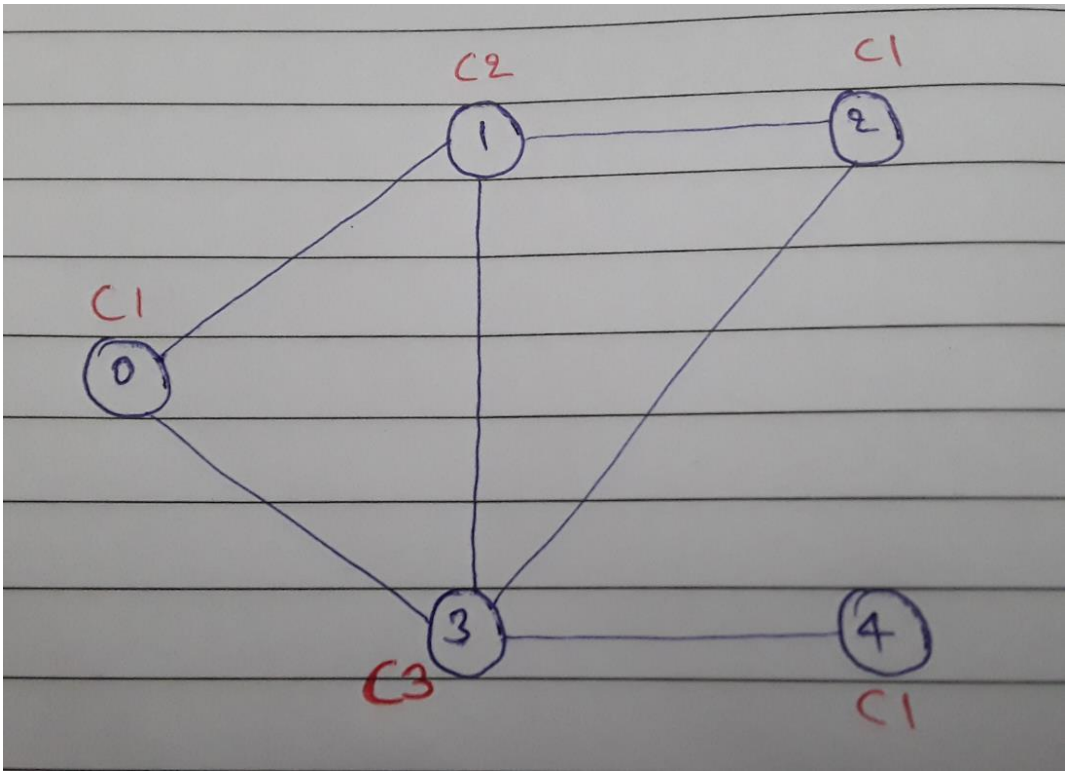


Backtrack



Vertex Coloring with 2 colors is not possible

Vertex Coloring with 3 colors




```
public class App {

	public static void main(String[] args) {

		int[][] adjacencyMatrix = new int[][] {
			{ 0, 1, 0, 1, 0 },
			{ 1, 0, 1, 1, 0 },
			{ 0, 1, 0, 1, 0 },
			{ 1, 1, 1, 0, 1 },
			{ 0, 0, 0, 1, 0 } };

		GraphColoring graphColoring = new GraphColoring(adjacencyMatrix);
		graphColoring.setNumberOfColors(3);
		graphColoring.solveVertexColoring();
	}
}
```

```
public class GraphColoring {

	private int[][] adjacencyMatrix;
	private int numOfVertices;

	private int numOfColors;
	private int[] colors;

	public GraphColoring(int[][] adjacencyMatrix) {
		this.adjacencyMatrix = adjacencyMatrix;
		this.numOfVertices = adjacencyMatrix.length;
		this.colors = new int[numOfVertices];
	}

	public void setNumberOfColors(int numOfColors) {
		this.numOfColors = numOfColors;
	}

	…
}


public void solveVertexColoring() {
		if(	colorVertex(0) ) {
			printVertexColoring();
		} else {
		System.out.println("Not possible to color entire graph with just "
					+ this.numOfColors + " color");
		}
}
```

```java
public boolean colorVertex(int vertexIndex) {
// Base case: If we were able to color all the vertices, it means we got our solution
            if (vertexIndex == numOfVertices) {
                  return true;
            }

            //try all colors, starting from first color
            for (int colorIndex = 1; colorIndex <= numOfColors; colorIndex++) {

                  //try to assign the color to the node
                  if (isColorValid(vertexIndex, colorIndex)) {
                        //If color is valid, assign that color to the vertex
                        colors[vertexIndex] = colorIndex;
                        // Color the next Vertex
                        if( colorVertex(vertexIndex + 1) ) {
                              return true;
                        }
                        // !!! Backtrack
                        colors[vertexIndex] = 0;
                  }
            }
// return false, as it was not feasible to color the vertex using any of the colors
            return false;
}
```

```java
public boolean isColorValid(int vertexIndex, int colorIndex) {
  for (int i = 0; i < numOfVertices; i++) {
       //if the adjacent vertex has the same color then return false
        if (adjacencyMatrix[vertexIndex][i] == 1 && colors[i] == colorIndex) {
                        return false;
        }
     }
  return true;
}
```