Code for Graph Coloring Algorithm

import java.util.\*;

public class EXPT\_09\_Graph\_Coloring {

    public static void main(String args[]) {

        Scanner in = new Scanner(System.in);

        int num\_vert;

        System.out.println("Enter the number of vertices in the graph: ");

        num\_vert = in.nextInt();

        // int graph[][] = new int[num\_vert][num\_vert];

        /\*

         \* for (int i = 0; i < num\_vert; i++) {

         \* for (int j = 0; j < num\_vert; j++) {

         \* System.out.print("\nEnter value for [" + i + "][" + j + "]: ");

         \* graph[i][j] = in.nextInt();

         \* }

         \* }

         \*/

        // It is tedious to enter each element of a graph many times for

        // different test cases. So, initializing graph at once before running.

        /\*

         \* int graph[][] = { { 0, 1, 1, 0, 0, 1, 1, 1 },

         \* { 1, 0, 0, 1, 0, 1, 0, 1 },

         \* { 1, 0, 0, 1, 1, 0, 1, 1 },

         \* { 0, 1, 1, 0, 1, 0, 0, 1 },

         \* { 0, 0, 1, 1, 0, 0, 0, 0 },

         \* { 1, 1, 0, 0, 0, 0, 0, 0 },

         \* { 1, 0, 1, 0, 0, 0, 0, 0 },

         \* { 1, 1, 1, 1, 0, 0, 0, 0 } };

         \*/

        int graph[][] = { { 0, 1, 1, 0 },

                { 1, 0, 1, 1 },

                { 1, 1, 0, 1 },

                { 0, 1, 1, 0 } };

        System.out.print("\nDisplaying entry:\n ");

        for (int i = 0; i < num\_vert; i++) {

            for (int j = 0; j < num\_vert; j++) {

                System.out.print("\t" + graph[i][j]);

            }

            System.out.print("\n");

        }

        System.out.print("\nEnter the maximum chromatic number for the graph: ");

        int m = in.nextInt();

        long start = System.currentTimeMillis();

        graphColoring(num\_vert, graph, m);

        long end = System.currentTimeMillis();

        System.out.print("\nTime taken for Graph Coloring Algorithm for graph with " + num\_vert + " vertices is "

                + (end - start) + " milliseconds.");

        in.close();

    }

    static void graphColoring(int num\_vert, int graph[][], int m) {

        Scanner in = new Scanner(System.in);

        int color[] = new int[num\_vert];

        color[0] = 0; // Assign first color for the first node

        boolean colorUsed[] = new boolean[num\_vert]; // Used to check whether color is used or not

        int col\_assigned = 0;

        for (int i = 1; i < num\_vert; i++)

            color[i] = -1; // initialize all other vertices are unassigned

        for (int i = 0; i < num\_vert; i++)

            colorUsed[i] = false; // initially any colors are not chosen

        for (int u = 1; u < num\_vert; u++) { // for all other NODE - 1 vertices

            for (int v = 0; v < num\_vert; v++) {

                if (graph[u][v] == 1) {

                    if (color[v] != -1) // when one color is assigned, make it unavailable

                        colorUsed[color[v]] = true;

                }

            }

            int col;

            for (col = 0; col < num\_vert; col++)

                if (!colorUsed[col]) // find a color which is not assigned

                    break;

            color[u] = col; // assign found color in the list

            col\_assigned++;

            for (int v = 0; v < num\_vert; v++) { // for next iteration make color availability to false

                if (graph[u][v] == 1) {

                    if (color[v] != -1)

                        colorUsed[color[v]] = false;

                }

            }

        }

        System.out.print("The total number of colors assigned is " + col\_assigned);

        if (col\_assigned <= m) {

            System.out.print(

                    "\nIt is possible to color the graph as the number of colors assigned is less than or equal to maximum chromatic number.");

        } else {

            System.out.print(

                    "\nIt is not possible to color the graph as the number of colors assigned greater than the maximum chromatic number.");

        }

        for (int u = 0; u < num\_vert; u++) {

            System.out.print("\nVertex " + u + " is assigned with color " + color[u]);

        }

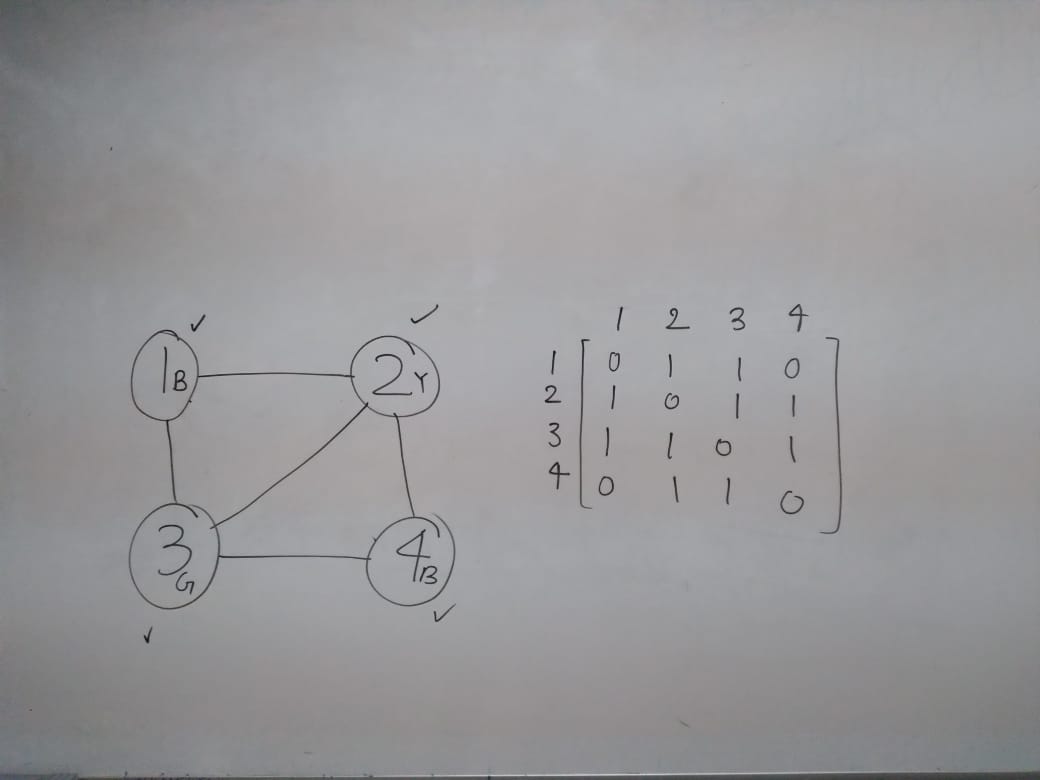
        in.close();

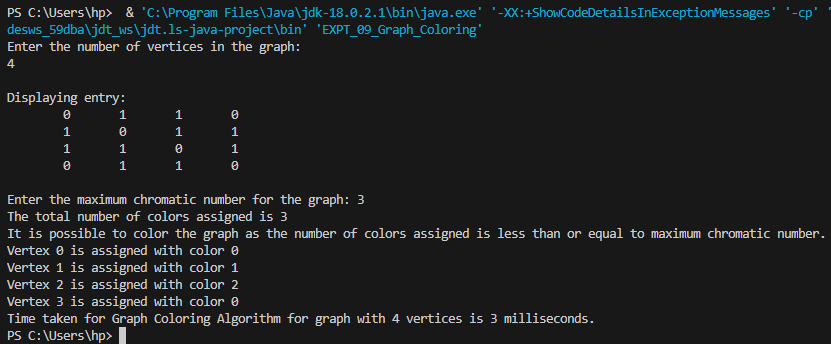
    }

}

Case 1: When the number of vertices is 4

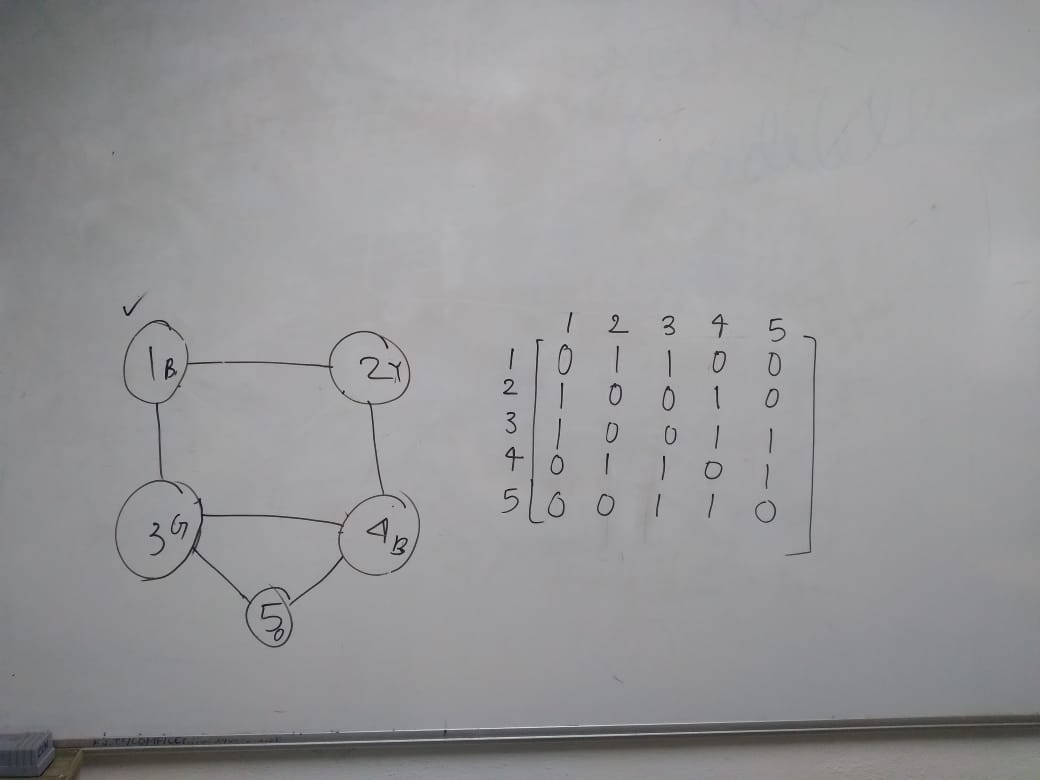
Time taken is 2 milliseconds.

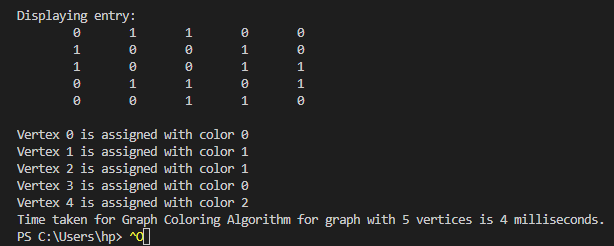




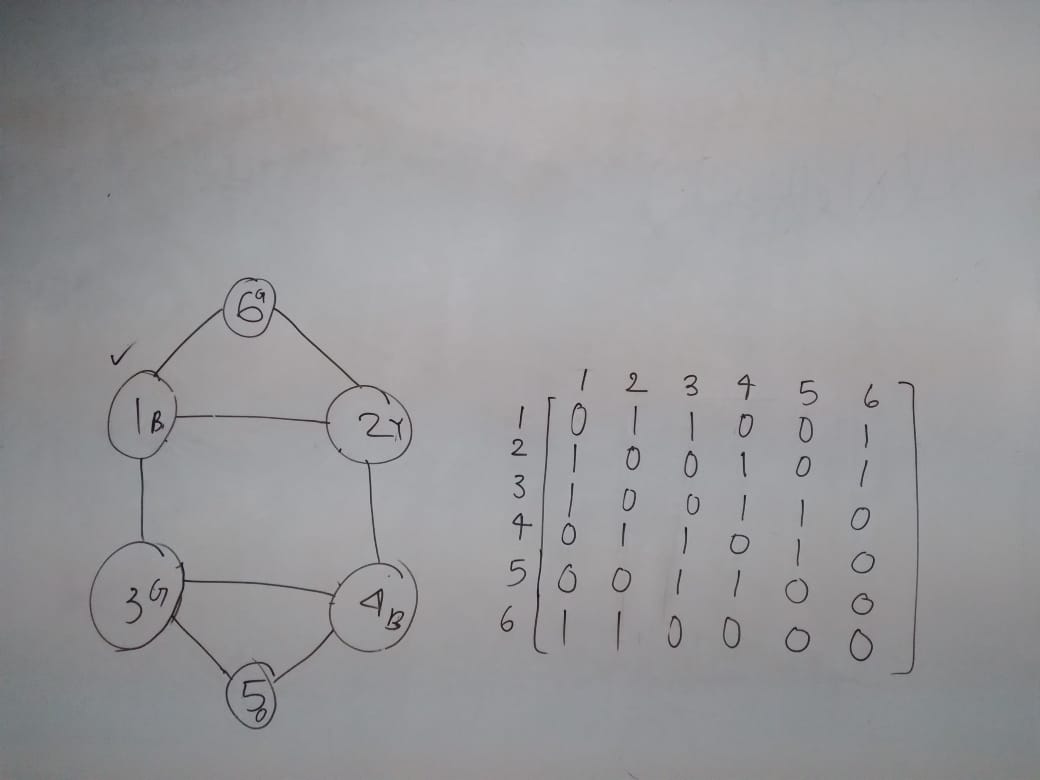
Case 2: When the number of vertices is 5.

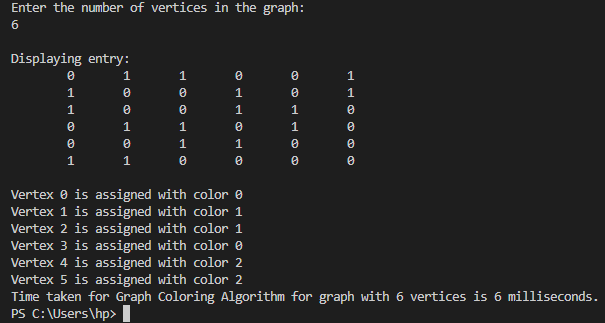
Time taken is 4 milliseconds.



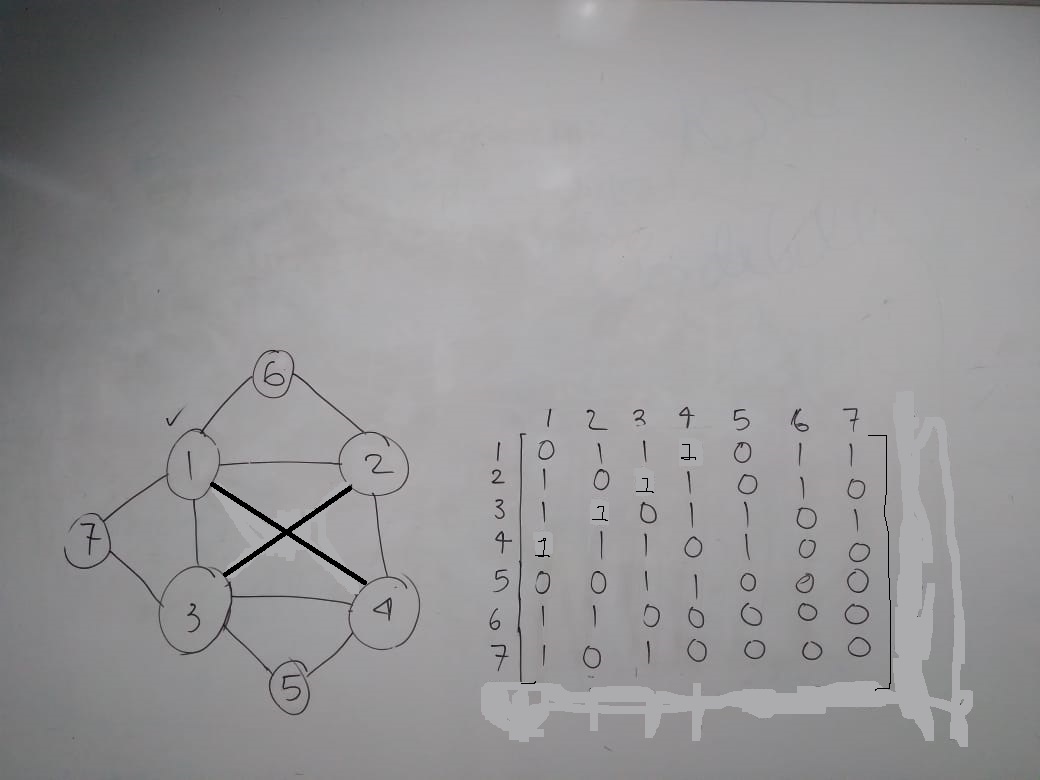


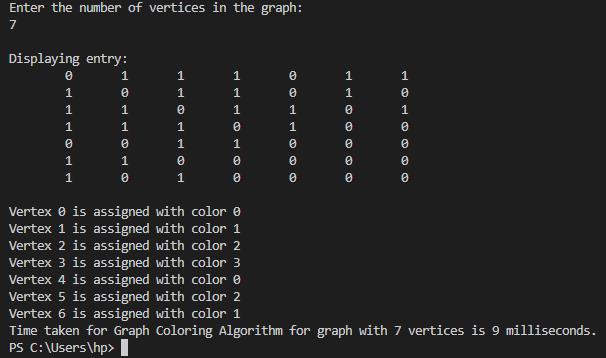
Case 3: When the number of vertices is 6.

Time taken is 6 milliseconds. 



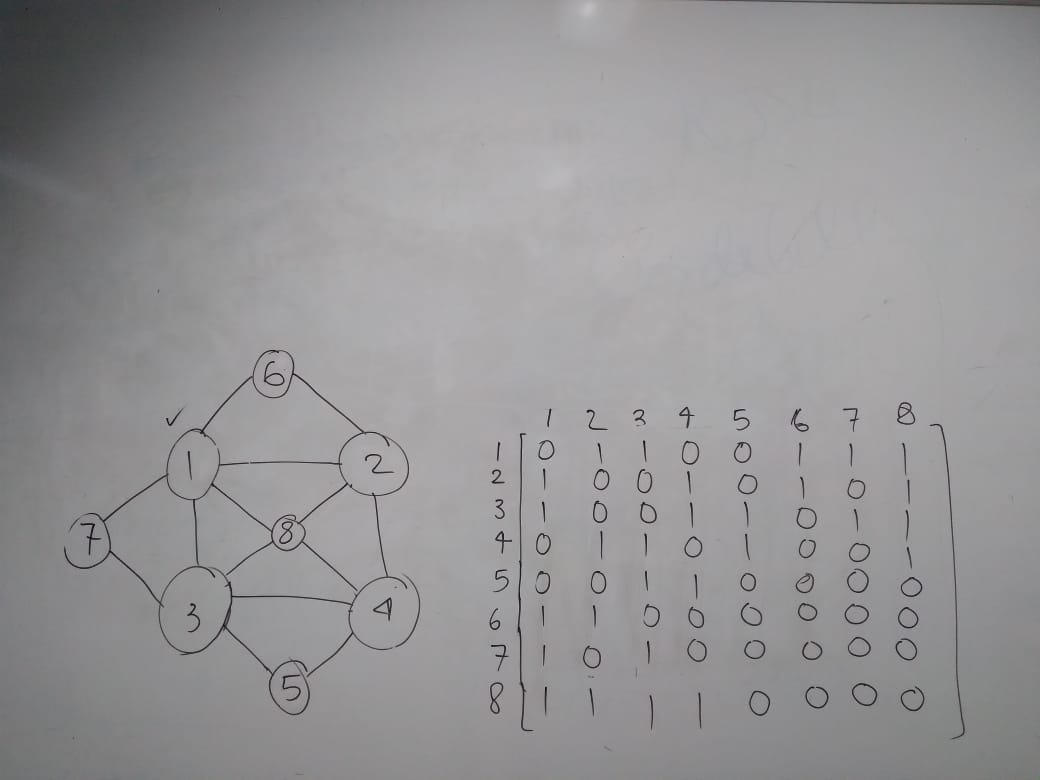
Case 4: When the number of vertices is 7.

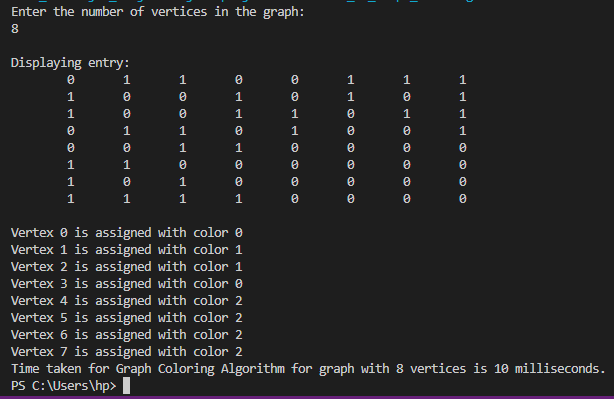




Time taken is 9 milliseconds.

Case 5: When the number of vertices is 8.





The time taken is 10 milliseconds.