Complejidad Computacional 2024-2
Facultad de Ciencias UNAM
Practica 1: Esquemas de Codificacion

Dominguez Cruz Emiliano: 315335075
Miñon Velazquez Jose Alberto: 314202226

Fecha de entrega: Viernes 1 de Marzo de 2024

1. Investigar en que consiste el problema de optimizacion de coloracion en grafos, dar su definicion como problema de optimizacion.

   Consiste en dado un grafo, cual es la minima cantidad de colores que necesitamos para colorear todos sus vertices de tal manera que ningun par de vertices adyacentes sean del mismo color.

   Entendamos que dos vertices sean adyacentes como que exista una arista que los una.

2. Describir k -*coloracion* como problema de decision.

   Consiste en dado un grafo, y un entero k si existe una manera de colorear sus vertices con k colores de tal manera que ningun vertice sea adyacente a otro del mismo color.

3. Describir e implementar un esquema de codificacion para ejemplares del problema del k - *coloracion* en grafos (en su version de decision).

   A continuación, se describe e implementa un esquema de codificación utilizando matrices de adyacencia:

   a) Crear una matriz de adyacencia de tamaño NxN, donde N es el número de vértices del grafo. Inicialmente, todas las entradas de la matriz se establecen en cero.

   b) Para cada arista (u, v) en el conjunto de aristas E del grafo, donde u y v son los vértices adyacentes, establecer la entrada correspondiente en la matriz de adyacencia en 1. Esto indica la existencia de una arista entre los vértices u y v.

   c) La matriz de adyacencia resultante representa la estructura de conexiones del grafo. Cada fila y columna de la matriz corresponde a un vértice del grafo, y los valores no nulos indican las aristas entre los vértices.

   En términos de implementación, se puede utilizar una matriz bidimensional o una lista de listas para representar la matriz de adyacencia. Cada entrada de la matriz se puede representar como un valor booleano (True o False) para indicar la presencia o ausencia de una arista.

```
public class Grafo {
    private boolean[][] matrizAdyacencia;

    // Constructor que recibe el nombre de un archivo
```

```
    public Grafo(String archivo) throws FileNotFoundException {
        try {
            // Lee el contenido del archivo y lo convierte a una cadena de texto
            byte[] contenidoBytes = Files.readAllBytes(Paths.get(archivo));
            String contenidoTexto = new String(contenidoBytes, Charset.defaultCharset());

            // Divide el contenido en partes utilizando el carácter '#'
            String[] partes = contenidoTexto.split("#");

            // Crea una matriz booleana basada en las partes del archivo
            matrizAdyacencia = new boolean[partes.length][partes[0].length()];
            for (int i = 0; i < matrizAdyacencia.length; i++) {
                for (int j = 0; j < matrizAdyacencia[i].length; j++) {
                    matrizAdyacencia[i][j] = partes[i].charAt(j) == '1';
                }
            }
        } catch (IOException e) {
            System.err.println("Error al leer el archivo: " + e.getMessage());
        }
    }

    // Método para verificar si dos vértices son adyacentes
    public boolean esAdyacente(int vertice1, int vertice2) {
        return this.matrizAdyacencia[vertice1][vertice2];
    }

    // Método para obtener la matriz de adyacencia completa
    public boolean[][] getMatrizAdyacencia() {
        return matrizAdyacencia;
    }
}
```

4. Describir e implementar un algoritmo eficiente para determinar si una grafica es 2 - coloreable; la implementacion del algoritmo debe hacer uso del esquema de codificacion implementado en el inciso anterior.

   El programa implementado debe recibir como entrada (desde consola) el nombre del archivo que contiene el ejemplar a probar, y como salida debera imprimir:

   - Numero de Vertices
   - Numero de Aristas
   - Vertice de mayor grado (nombre o etiqueta del vertice y valor del grado)
   - ¿La grafica del ejemplar es 2 - coloreable? [Responder con un **SI** o un **NO**]
   - En caso de que la grafica sea 2 - coloreable, lista de vertices indicando el color de cada vertice.

   El algoritmo propuesto toma la matriz de adyacencia obtenida de la codificacion, y crea una lista para representar el color (0,1) donde todas las entradas son -1 inicialmente para denotar que aun no son coloreadas.

Itera sobre la matriz, tomando una linea y revisando si el vertice correspondiente a dicha linea ya esta coloreado, si no lo esta le asigna el color 0 arbitrariamente, si lo esta le deja su color. Despues se revisan sus adyacencias en la matriz y cada que encontremos una revisamos el color del vertice, si no tiene color le asignamos uno opuesto al de la linea, si ya tiene el color opuesto no hacemos nada, sin embargo si ya esta coloreado del mismo color inmediatamente sabemos que la grafica no es 2-coloreable ya que si lo dejamos como esta tendriamos dos vertices adyacentes del mismo color, pero si lo cambiamos hariamos que fuera del mismo color del vertice que orignalmente le asigno ese color.

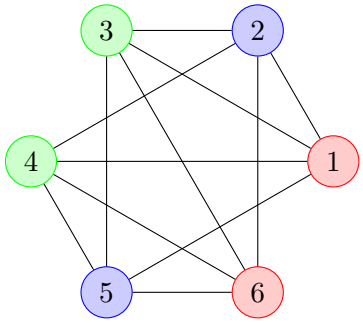
Repetimos este proceso hasta terminar, si logramos iterar sobre toda la matriz sin encontrar un fallo es porque pudimos hacer la 2-coloracion sin encontrar problemas. Devolvemos entonces verdadero e imprimimos la lista de coloracion que fuimos construyendo.

En el peor de los casos este algortimo tendra que iterar sobre una matriz tamaño n x n por lo que podemos afirmar su complejidad es de

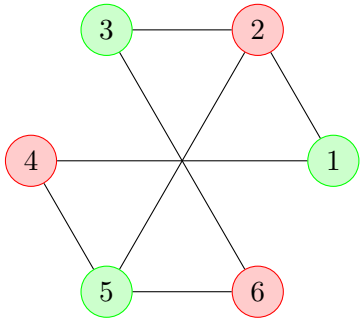
$$O(n^2)$$

5. Incluir ejemplos de ejecucion, asi como ejemplos graficos de los ejemplares con los que prueben su programa. Al menos deben agregar ejemplos de los siguientes casos (con al menos 6 vertices en cada caso)

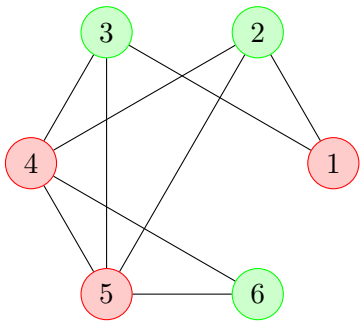
   - ejemplar 3 - *coloreable*



   - ejemplar 2 - *coloreable*



   - ejemplar que **NO** sea 2 - *coloreable*


3
2
4
1
5
6