**Trabajo Práctico 4 - Coloreo de Grafos**

**Introducción:**

El coloreo de grafos consiste básicamente, en obtener la menor cantidad de colores para pintar un grafo. Se debe tener en claro que, dos nodos adyacentes no pueden tener el mismo color, y que no deben quedar nodos sin pertenencia a algún grupo de color.

En esta tarea se comprenderán el uso de generadores de grafos (de diverso tipo), la codificación de 3 algoritmos de coloreo (secuencial aleatorio, Welsh-Powell y Matula) para realizar el coloreo, y un programa probador de algoritmo de coloreo.

Documentación de clases

Clase: Archivo

Contiene los métodos de lectura y escritura de archivos.

Metodos:

**public static GrafoNDNP leerGrafo(String ruta)**. Lee un archivo de entrada almacenado en la dirección “ruta”, arma el grafo y lo devuelve.

**public static void escribirGrafo(GrafoNDNP g, String ruta).** Escribe un archivo de entrada en la dirección “ruta” a partir del grafo “g”.

**public static void escribirGrafoColoreado(GrafoNDNP g, String ruta).** Escribe un archivo de salida en la dirección “ruta” a partir del grafo “g”.

**public static GrafoNDNP leerGrafoColoreado(String ruta).** Lee un archivo de salida almacenado en la dirección “ruta”, rearma y retorna el grafo coloreado.

Clase: Generador

Contiene los métodos para generar grafos aleatorios y regulares.

Metodos:

**public static GrafoNDNP aleatorioPorcAdy(int cantNodos, double porcAdy).** Genera y retorna un grafo no dirigido y no ponderado con “cantNodos” nodos y un porcentaje de adyacencia “porcAdy”.

**public static GrafoNDNP aleatorioProbAristas(int nodos, double prob).** Genera y retorna un grafo no dirigido y no ponderado con “cantNodos” nodos y una probabilidad “prob” para las aristas.

**public static GrafoNDNP regularPorcAdy(int nodos, double porc).** Genera y retorna un grafo no dirigido y no ponderado donde todos los nodos tienen la misma cantidad de aristas con “cantNodos” nodos y una un porcentaje de adyacencia “porc”.

**public static GrafoNDNP regularGrado(int nodos, int grado)**. Genera y retorna un grafo no dirigido y no ponderado donde todos los nodos tienen la misma cantidad de aristas (grado) con “cantNodos” nodos.

**private static void genGrado1(GrafoNDNP g, int nodos).** Método utilizado por el generador regular para los grafos cuyo grado es 1.

**private static void genGrado2(GrafoNDNP g, int nodos).** Metodo utilizado por el generador regular para los grafos cuyo grado es 2.

**private static void genConSalto(GrafoNDNP g, int nodos, int grado, int salto).** Metodo utilizado por el generador para generar grafos regulares a partir de la cantidad de nodos (nodos), el grado del grafo (grado) y un valor “salto” que se utilizará para la construcción.

**private static void genGradoMultiplo(GrafoNDNP g, int nodos, int grado).** Metodo utilizado por el generador para generar grafos regulares cuya cantidad de nodos es múltiplo del grado.

**public static void partito(GrafoNDNP g, int nodos, int part).** Método utilizado para generar grafos n-partitos (part).

Clase: Coloreo

Contiene los métodos y atributos necesarios para emplear los algoritmos de coloreo.

Atributos:

private int [][] elementos. Matriz que guarda los elementos originales a colorear.

private int [][] mat. Matriz copia que se utiliza para mezclar y colorear sin perder el orden del grafo original.

private int [] colores. Vector que almacena los colores para poder contarlos.

Metodos:

**public Coloreo(GrafoNDNP g).** Constructor que inicializa las matrices utilizadas para colorear.

**public int colorear(GrafoNDNP g).** Metodo que ejecuta el coloreo del grafo.

**public void mezclar().** Métodos utilizado para mezclar el orden de coloreo.

**public void ordenMayorAMenor().** Metodo utilizado para establecer el orden de coloreo según el grado de cada nodo.

**public void ordenMenorAMayor().** Metodo utilizado para establecer el orden de coloreo según el grado de cada nodo.

**public int [] getColores().** Metodo que devuelve un vector con todos los colores que se utilizaron sobre el grafo.

Clase: GrafoNDNP

Contiene los métodos y atributos necesarios para implementar un grafo no dirigido y no ponderado.

Atributos:

protected int[] colores. Colores utilizados en el coloreo.

protected int cantColores. Cantidad de colores utilizados en el coloreo.

protected int cantAristas. Cantidad de aristas que contiene el grafo.

protected int porcAdy. Porcentaje de adyacencia del grafo.

protected int gradoMax. Grado maximo de los nodos del grafo.

protected int gradoMin. Grado mínimo de los nodos del grafo.

Metodos:

**public GrafoNDNP(int nodos).** Constructor que crea un grafo con la cantidad “nodos” de nodos.

**public void secuencial().** Método que colorea el grafo en orden secuencial.

**public void welshPowell().** Método que colorea el grafo según el algoritmo de Welsh-Powell.

**public void matula().** Método que colorea el grafo según el algoritmo de Matula.

Clase: MatrizSimetrica.

Contiene los métodos y atributos necesarios para implementar un grafo no dirigido y no ponderado sobre un vector.

Atributos:

protected boolean[] vector. Vector que almacena la adyacencia entre nodos.

protected int tam. Cantidad de nodos del grafo.

Metodos:

**public MatrizSimetrica(int n).** Constructor que inicializa el vector de aristas a partir de la cantidad “n” de nodos.

**public boolean todosMismoGrado(int grado).** Metodo utilizado por el generador regular para verificar si fue posible formar un grafo con todos los nodos del mismo grado esperado (grado).

Clase: Probador.

Contiene los métodos necesarios para verificar si el programa de coloreo funciona correctamente.

**private static boolean test(GrafoNDNP in, GrafoNDNP out).** Compara la entrada y la salida del programa y determina si el resultado obtenido es correcto.