TRABAJO

PRACTICO

N°2

INTEGRANTES: TILCA GONZALO – LEON PATRICIO

**Trabajo Práctico 2: Diseñando Regiones**

**Introducción**

El objetivo de este trabajo práctico es hallar una clique de peso máximo, utilizando Algoritmos golosos para poder obtener dicha clique del conjunto de vertices.

**Decisiones de Diseño**

Para organizar el desarrollo de la aplicación, decidimos dividirlo en 6 secciones: AlgoritmoGoloso, Grafo, GrafoGUI, Main, TestAlgoritmoGoloso y Vertice.

**1. AlgoritmoGoloso**

Esta sección corresponde a la parte lógica donde calculamos la clique maxima. Los metodos que se encuentran dentro de esta clase (obviando el constructor) son:

* List<Vertice> encontrarCliquePesoMaximo(): Encuentra y devuelve una clique de peso máximo en el grafo utilizando un algoritmo goloso. Obteniendo todos los vertices del grafo y ordenandolos de forma descendente con respecto al peso. Luego inicializa una lista vacia "clique", e itera sobre los vertices ordenados, si la clieu esta vacia, añade el vertice actual. Si no está vacía, añade el vértice actual si es vecino de todos los vértices ya en la clique. Por ultimo, devuelve la lista de vértices que forman la clique de peso máximo.
* String obtenerResultadoCliquePesoMaximo(): Calcula y devuelve una representación en formato de cadena de la clique de peso máximo y su peso total. Primero llama al método encontrarCliquePesoMaximo() para obtener la clique de peso máximo. Despues inicializa un StringBuilder para construir la representación de la cadena, iterando sobre los vertices de la clique, añadiendo la información de cada a la cadena. Luego calcula y añade el peso total de clique, finalizando por devolver el toString().

**2. Grafo**

Esta sección contiene crea un grafo a partir de un HashMap, y contiene metodos para poder agregar los vertices y los arcos del mismo, (obviando el constructor) estos son:

* public void agregarVertice(String id, int peso): Agrega un nuevo vertice al grafo con su identificador y peso dados.  
   Comprueba si el vertice dado no exista en dicho grafo, si no existe, lo agrega como un nuevo objeto "Vertice", con los parametros pasados y añadiendolo al HashMap.
* public void agregarArco(String id1, String id2): Agrega un arco (o arista) entre dos vértices especificados por sus identificadores. Primero obtiene los vértices correspondientes a los identificadores id1 y id2 desde el HashMap de vértices. Si ambos vértices existen, añade v2 como vecino de v1 y viceversa, haciendo que la relación sea bidireccional.
* public Collection<Vertice> getVertices(): Devuelve una colección de todos los vértices en el grafo.

**3. GrafoGUI**

Esta sección es la interfaz grafica, que tambien contiene metodos que cargan los vertices, con su peso y sus aristas, para poder cargar un grafo y calcular su clique mas adelante :

* public GrafoGUI(): Inicializa la interfaz gráfica de usuario (GUI) para manipular un grafo. Configurando la ventana principal con un título, tamaño, comportamiento de cierre y posición.Luego crea y organiza componentes de la interfaz gráfica, incluyendo etiquetas, campos de texto y botones, en un panel con un layout de GridLayout y añade listeners a los botones para manejar eventos de acción.
* private class AgregarVerticeListener implements ActionListener: Maneja el evento de agregar un vértice al grafo cuando se presiona el botón "Agregar Vértice". Obtieniendo el identificador del vértice y su peso desde los campos de texto correspondientes para luego llamar al método agregarVertice() para añadir el vertice. Por último limpia los campos de texto y muestra un mensaje de confirmación.
* private class AgregarArcoListener implements ActionListener: Maneja el evento de agregar un arco entre dos vértices cuando se presiona el botón "Agregar Arco". Obtiene los identificadores de los dos vértices desde los campos de texto correspondientes. Llama al método agregarArco del grafo para añadir el arco entre los vértices. Limpia los campos de texto y muestra un mensaje de confirmación.
* private class CalcularCliqueListener implements ActionListener: Maneja el evento de calcular la clique de peso máximo cuando se presiona el botón "Calcular Clique" llamando al método obtenerResultadoCliquePesoMaximo de AlgoritmoGoloso para obtener el resultado en formato de cadena. Por último el resultado en un cuadro de diálogo.

**4. Main**

Inicia la aplicación GUI y asegura que la GUI se maneje de manera segura en el hilo de despacho de eventos de Swing.

**5. TestAlgoritmoGoloso**

Esta sección contiene los test para la clase "AlgoritmoGoloso", la cual comprueba que funcionen los metodos de dicha clase, ante algunos casos bases planteados.

* void testEncontrarCliquePesoMaximo(): Verifica que el método encontrarCliquePesoMaximo encuentra la clique de peso máximo en un grafo con varios vértices y arcos.
* void testCliqueUnVertice(): Verifica que el método encuentra correctamente la clique de peso máximo cuando hay un solo vértice.
* void testCliqueNingunVertice(): Verifica que el método no encuentra ninguna clique en un grafo con un solo vértice aislado.
* void testCliqueConTodosLosVertices(): Verifica que el método encuentra correctamente una clique completa donde todos los vértices están conectados entre sí.

**6. Vertice**

Esta clase contiene la estrucutra de un objeto **Vertice.**

* El constructor inicializa un vértice con un identificador, un peso y un conjunto de vecinos vacío.
* Los métodos public String getId() y public int getPeso() devuelven el identificador y el peso del vértice respectivamente.
* El método public Set<Vertice> getVecinos() devuelve el conjunto de vecinos del vértice.
* El método public void agregarVecino(Vertice vecino) agrega un vértice vecino al conjunto de vecinos del vértice actual y viceversa.