



Trabajo Práctico

“Clustering Humano”

Programación III

Comisión 1

Alumnos

- Krujoski, Melanie Belén – 40913717/2016
- Gomez, Leandro Antonio – 39586115/2018
- Rages, Daniela – 38795763/2016

Profesores

- Javier Marengo
- Patricia Bagnes

Análisis

El enunciado que nos fue dado para este segundo trabajo práctico, consta de crear un Grafo donde se asigna el nombre y un puntaje para cuatro intereses de cada persona: deportes, música, espectáculos y ciencia. Estos datos se deben mostrar en una ventana de aplicación, donde se calcula la similaridad entre dichos intereses de cada persona y a partir de ello, se los divide en dos grupos.

Para crear esta aplicación, desarrollamos una parte lógica y una interfaz gráfica. En la lógica, definimos los distintos objetos que pertenecen al grafo: Vértices y Aristas. Los vértices serán las personas especificadas por el usuario, quien se encarga de llenar los datos de estas. Las aristas serán la unión entre cada persona, cuyo peso es el índice de similaridad entre cada par de personas unidas. Además, se desarrolla un árbol generador mínimo, utilizando como concepto el algoritmo de Robert C. Prim para eliminar las uniones (aristas) más pesadas entre personas (vértices), y para recorrer el grafo, aplicamos el concepto de BFS (Breadth First Search), que nos ayudó a encontrar los vecinos de cada vértice, obteniendo de esta manera los dos grupos que buscábamos formar.

Además, decidimos implementar test unitarios por cada entidad y método creado en la lógica de la aplicación. Esto nos fue de mucha ayuda para detectar errores o fallas lógicas de la implementación y lograr resolverlos. Creemos que es una buena práctica para verificar cada parte del código y asegurarse una correcta implementación de un proyecto.

En cuanto a la interfaz, decidimos crear una ventana (frame) principal en la que se pide la cantidad de personas que el usuario quiere almacenar. Luego utilizamos dos paneles: El primero se encarga de pedir los datos de cada persona, guardarlos para crear el grafo y los muestra en forma de tabla para mejor visualización de dichos datos. El segundo panel muestra una tabla dividida en los dos grupos que se forman, listando debajo los nombres de cada persona correspondiente a cada grupo.

Desarrollo de la aplicación

Comenzamos el desarrollo de esta aplicación analizando que entidades debe contener. Encontramos que principalmente para formar un grafo necesitamos Vértices y Aristas, por lo cual creamos dichas clases.

A la entidad Vértice la llamamos Persona, ya que el valor de cada Vértice serán los datos de las personas: Nombre, intereses en deporte, música, espectáculos, ciencia. En dicha clase construimos una Persona con estos datos, y calculamos el índice de similaridad, que toma otra persona con sus datos y compara solo sus intereses, devolviendo un entero, que es el “peso” de la unión o arista de estas dos personas. También sobrescribimos el método equals para poder comparar dos personas con sus atributos específicos.

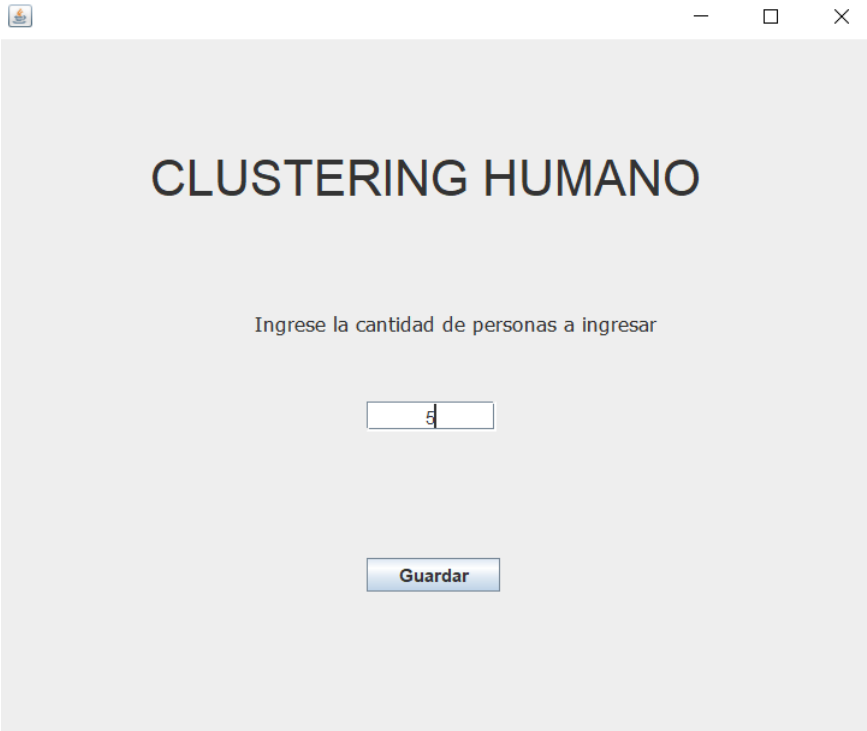
La entidad Arista, consta de dos personas y un peso, ya que la unión de ellas dará una arista con el peso igual al índice de similaridad entre ellas. Para esta clase también decidimos sobrescribir el método equals que permite comparar dos aristas con dichos atributos.

Habiendo creado las dos entidades principales, construimos un grafo. Este tiene una lista de vértices y otra de aristas, y para construirlo, le pediremos al usuario que especifique la cantidad de personas que quiere almacenar. En esta clase se agregan los vértices y las aristas, y además crean métodos para eliminar aristas, verificar si existe un vértice o una arista, eliminar la arista de mayor peso, comparar si dos aristas son idénticas, buscar los vecinos de un vértice determinado, verificar que no se repitan vértices. Todas estas acciones ayudan a generar un grafo correctamente, sin repeticiones, y con la capacidad de consultar, agregar o eliminar datos cuando se lo requiera.

Ya obtenido el grafo, queremos generar los dos grupos. Para ello creamos una clase Prim, en donde se genera un Árbol Generador Mínimo que elimina las aristas más pesadas, dejando las de menor índice de similaridad. Luego, mediante el algoritmo BFS buscamos los vértices alcanzables de cada uno, sus vecinos, y los agregamos a una lista para luego crear los dos grupos que buscamos según sus intereses definidos.

Luego de haber desarrollado la lógica y verificar que el algoritmo creado funciona correctamente mediante el uso de test unitarios, creamos una interfaz grafica donde mostramos dichos datos.

La primera ventana de la aplicación, muestra el título/nombre de la misma: Clustering Humano, y pregunta cuantas personas desea almacenar. Mediante un JTextField lee el entero ingresado y con un botón lo guarda como parámetro para luego generar el grafo.



Esta ventana, cuando guarda el entero al presionar el botón Guardar, llama al siguiente panel (optamos por usar un frame y dos paneles).

Este panel se encarga de pedir los datos de cada persona al usuario y mostrarlos en forma de tabla. Al pedir un nombre, restringe a que este tenga mas de dos letras, no puede ser vacío ni menor a dos caracteres, tampoco puede contener números. Para cada interés se presenta un comboBox con opciones del 1 al 5 para que se elija el puntaje de cada interés de la persona. Cuando se cargan la cantidad de personas especificadas, se deshabilita la edición y muestra la tabla con todas las personas y sus intereses almacenados. A partir de ello, puede presionar el botón Generar Grafo, que lleva al segundo panel.

Ingrese nombre y apellido:

Seleccione un valor del 1 al 5 para las siguientes característic...

Interés por los deportes:

Interés por la música:

Interés por las noticias del espectáculo:

Interés por la ciencia:

Nombre	Deportes	Música	Espectáculo	Ciencia
Melanie Krujoski	3	5	2	4
Leandro Gomez	5	2	3	4
Daniela Rages	2	5	2	5

Generar Grafo

El segundo panel muestra una tabla con dos columnas: Grupo 1 y Grupo 2. Bajo cada una, se listan los nombres de las personas, utilizando un JList para visualizarlas. Aquí es donde vemos los grupos formados según las personas que pide el usuario.

GRUPO 1	GRUPO 2
Melanie Krujoski Daniela Rages	Leandro Gomez

Dificultades y resoluciones

Una de las dificultades que se nos presentó al momento de comenzar a analizar este trabajo, fue qué estructura de datos era mas conveniente utilizar, para lograr una implementación rápida y eficaz de la aplicación. Al utilizar un arreglo estático, nos limitaba a crear un grafo de una cantidad fija de personas, sin posibilidad de modificar dicho largo (ya que eso resultaría en un código poco eficiente). Pero para este trabajo, encontramos que resultaba más fácil y conveniente pedirle al usuario cuantas personas desea almacenar para luego, a partir de ello, crear el grafo. Por lo tanto optamos por utilizar ArrayList en ambas listas: Vértices y Aristas.

Por otro lado, desde que empezamos la carrera, solemos utilizar la consola para imprimir datos y de esta manera verificar que el código está bien implementado. Este trabajo nos ayudó a seguir familiarizándonos con los test unitarios, e impulsó a que los incluyamos. Creemos que es una buena práctica para continuar utilizando en futuros proyectos, ya que realmente fue de gran ayuda para detectar errores en la lógica y tener la oportunidad de corregirlos.