

UNIVERSIDAD RAFAEL LANDÍVAR

FACULTAD DE INGENIERÍA

LICENCIATURA EN INGENIERÍA EN INFORMÁTICA Y SISTEMAS

MATEMÁTICA DISCRETA II

CATEDRÁTICO: INGENIERO JUAN CARLOS SOTO SANTIAGO

MANUAL TÉCNICO ISOMORFISMO-DISCRETO

INTEGRANTES: DEREK ANDRÉ MENENDEZ URIZAR

MYNOR OTTONIEL XICO TZIAN

BRYAN ESTUARDO MACARIO CORONADO

JOSÉ CARLOS MANSILLA MILLIAN

03 DE DICIEMBRE DE 2016

INTRODUCCIÓN

En el presente documento, se presenta el Análisis y Diseño del programa correspondiente al proyecto del curso de Matemática Discreta. El proyecto consistió en el desarrollo de un programa que permitiera reducir los tiempos de procesamiento para el análisis de isomorfismo entre dos grafos cargados por medio de la lectura de un archivo de texto.

El principal objetivo del programa, es facilitar la validación de relación de isomorfismo entre dos grafos y en caso de que exista una, encontrar al menos una función de isomorfismo entre los mismos. "Para el proceso de cómputo, se hizo uso de listas simples y matrices.

Para la solución del problema, fue necesario usar el lenguaje de alto nivel C# y el IDE Visual Studio 2015. Para comprobar que el programa funcionara como debiese, se creó una batería de pruebas con los casos más particulares que pudiera un usuario promedio ingresar.

En este reporte también se introducen los conceptos básicos sobre los temas mínimos que fueron necesarios conocer para poder desarrollar el proyecto de una manera efectiva, alcanzando así un mejor nivel de abstracción y obteniendo conocimientos de seguro serán útiles a lo largo de la carrera de ingeniería en informática y sistemas.

La aplicación desarrollada puede ser utilizada fácilmente por cualquier persona interesada en conocer si existen o no funciones de isomorfismo entre dos grafos, ya que resulta ser realmente cómoda y amigable con el usuario.

TEORÍA DE GRAFOS

Los grafos son una estructura de datos no lineal, la cual se puede usar para modelar diversas aplicaciones. Es una parte importante de la teoría combinatoria de matemáticas discreta, lo cual le da un carácter bastante amplio y complejo.

Los grafos se representan con frecuencia en la vida real, tal es el caso de una red de carreteras que enlace a cierto grupo de ciudades; aquí los nodos de la red o las ciudades representan los vértices del grafo, las carreteras que unen las ciudades representan los arcos o las aristas.

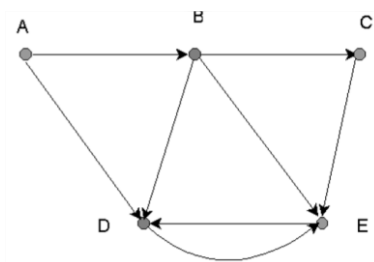


Figura 1: Representación gráfica de un grafo.

ISOMORFISMO

Dos grafos tendrán la misma “forma matemática” cuando la única diferencia entre ambos, en cuanto a su estructura, sea que hemos usado cualquier tipo de representación distinta para sus vértices, mientras que las conexiones entre ellos sean las mismas. En este caso diremos que son isomorfos.

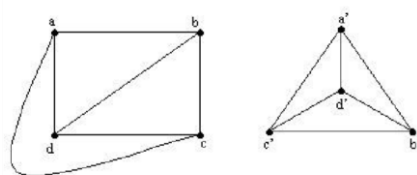


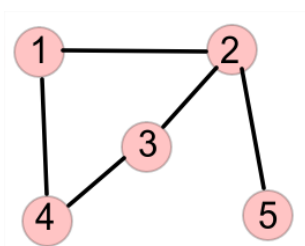
Figura 2: Ejemplo de isomorfismo

MATRIZ DE ADYACENCIA

La información de un grafo puede representarse mediante una matriz de n filas * n columnas (siendo n el número de vértices).

Para construir la matriz de adyacencia de un grafo, cada elemento de la matriz a_{ij} vale 1 siempre que haya una arista que conecte a los vértices i y j . De lo contrario el elemento a_{ij} vale 0.

Por ejemplo, si se desea construir la matriz de adyacencia del siguiente grafo:



La matriz será de 5×5 dado que el grafo contiene 5 vértices.

En la primera fila (la del vértice 1), se marca la columna del vértice 2 y la del vértice 4 con un uno por que son los vértices que conectan con 1.

Se procede de la misma forma con el resto de filas y se obtiene la matriz de adyacencia.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

EJEMPLO DE BÚSQUEDA DE FUNCIÓN DE ISOMORFISMO

Sean G_1 y G_2 el grafo de la figura de la figura izquierda y derecha, respectivamente.



Sea V_1 , el conjunto de vértices de G_1 .

$$V_1 = \{1, 2, 3, 4, 5\}$$

Sea V_2 , el conjunto de vértices de G_2

$$V_2 = \{1, 2, 3, 4, 5\}$$

Sea C el conjunto de permutaciones de V_2

$$C = \{\{1, 2, 3, 4, 5\}, \{1, 2, 3, 5, 4\}, \dots, \{5, 4, 1, 3, 2\}, \dots, \{5, 4, 3, 2, 1\}\}$$

1. Se toma la primera permutación y se genera la matriz de V_1 mapeado en algún elemento tomado de C .

Permutación tomada: $\{1, 2, 3, 4, 5\}$

Matriz generada (Filas representan los elementos de V_1 y columnas los elementos de la permutación tomada.

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

2. Se genera la matriz de adyacencia del segundo grafo.

$$A^H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

3. Se genera la matriz transpuesta de la matriz generada en el paso 1

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Se realiza el producto punto de $P * P^H * P$. En este caso se obtiene A^H por ser haber sido multiplicada por la matriz identidad.
5. Si la matriz que se obtiene del producto es igual a la matriz de adyacencia del primer grafo, entonces se ha encontrado una función de isomorfismo. Dada por los elementos del conjunto $V1$ y la permutación tomada del conjunto C .
6. Se procede con la siguiente permutación en C hasta encontrar una función, de existir alguna.

ANÁLISIS

NOMBRE DEL PROGRAMA

El nombre que recibe el programa desarrollado es **ISOMORFISMO-DISCRETO**. El nombre se debe al título con el cuál fue asignado el proyecto y el objetivo del mismo. El nombre se divide en dos partes *ISOMORFISMO* palabra con origen griego cuyo significado se encuentra estrechamente relacionado con la igualdad en forma de dos cosas, y *DISCRETO*, haciendo referencia al nombre del curso del cuál era el proyecto (Matemática Discreta II).

OBJETIVO DEL PROGRAMA

Agilizar el análisis de isomorfismo entre dos grafos brindando al usuario una manera fácil y cómoda de visualizar los dos grafos a comparar y las funciones de isomorfismo existentes entre ambos.

ALCANCE DEL PROGRAMA

El programa fue desarrollado y diseñado para que cualquier persona pueda visualizar la información de cualquier grafo y si existe o no alguna función de isomorfismo entre dos grafos; lo cual puede ser útil tanto en el ámbito académico como investigativo.

El programa incluye varias características que resultan de gran utilidad en el análisis de la información de grafos. Entre las funciones con las que el programa cuenta, se mencionan:

- **DIBUJAR**

Esta función permite al usuario ver de una forma gráfica las conexiones (para cada uno de los grafos) existentes entre los vértices.

- **BUSCAR FUNCIÓN**

Esta es, quizá la piedra angular del proyecto. Por medio de esta función se lleva a cabo un proceso de búsqueda iterativo por medio del cual se puede identificar al menos una función de isomorfismo para grafos con una cantidad de vértices mayor a 9 y todas las funciones para un número de vértices menor a 9.

- **GENERAR PDF**

La aplicación da al usuario la opción de generar un reporte en formato PDF con las funciones generadas por el mismo.

RESOLUCIÓN DEL PROBLEMA (PASOS DE POLYA)

1 PLANTEAMIENTO DEL PROBLEMA

Se requiere que se implemente una aplicación que sea capaz de validar si 2 grafos (no multi-grafos, no dirigidos) son isomorfos, en cuyo caso deberá generar una función de isomorfismo, en caso contrario indicar que no son isomorfos y el criterio que no cumplen:

- ✓ No tienen la misma cantidad de vértices
- ✓ No tienen la misma cantidad de aristas
- ✓ No concuerdan los grados de los vértices
- ✓ No es posible determinar una función de isomorfismo

Los grafos deberán ser leídos desde archivos de texto con el siguiente formato, uno por cada grafo.

➤ FORMATO DE ENTRADA (.TXT):

de vértices (n)
de vértice inicial A (0...n-1), # de vértice final A (0...n-1)
de vértice inicial B (0...n-1), # de vértice final B (0...n-1)
de vértice inicial C (0...n-1), # de vértice final C (0...n-1)

2 COMPRENDER EL PROBLEMA

El primer paso y el más importante fue entender de qué trataba el problema y los resultados que debían obtenerse.

Basados en el problema planteado se desglosó una lista de ideas importantes:

- i. Se necesita una aplicación que sea capaz de dar solución al problema.
- ii. Como se necesita una aplicación se debe hacer uso de la programación en algún lenguaje. (Queda a nuestro criterio elegir sobre que lenguaje programar).
- iii. De entrada, se tienen 2 grafos.
- iv. La cantidad de vértices y como se conectan los vértices entre sí, están almacenados en un archivo de texto con el formato establecido por el problema
- v. Se tiene un archivo de texto por cada grafo.
- vi. Los grafos son no dirigidos y no son multi-grafos.
- vii. La cantidad de vértices y aristas no tienen restricciones
- viii. Se necesita validar que estos 2 grafos fueran isomorfos o no.
- ix. De salida se tendrán 2 casos, en caso sean isomorfos y en el caso de que no lo sean.
- x. Si los grafos son isomorfos se debe generar alguna función de isomorfismo.
- xi. Si los grafos no son isomorfos se debe indicar porque no lo son.

3 FORMULAR UN PLAN

“Divide y vencerás”

Maquiavelo (Julio Cesar)

Se decidió tomar el problema general y dividirlo en sub-problemas más sencillos de resolver, de este modo tendríamos varias partes para resolver y de este modo es más fácil reconocer patrones, encontrar errores, etc.

4 EJECUTAR EL PLAN

Para llevar a cabo la solución se decidió utilizar el lenguaje de alto nivel C#, en el cual por medio del uso de POO se crearon las clases y métodos necesarios para llevar a cabo la solución al problema.

Basándonos en el problema general, se ha dividido en los siguientes sub problemas:

1. **Interacción con el Usuario:** Tiene la tarea de tener un entorno en el cual el usuario pueda interactuar con la aplicación, para cargar los archivos y mostrar el resultado del problema.
 - **Solución:** Para ello se hizo uso de una aplicación creada en Windows Forms, la cual posee un entorno gráfico que permite interactuar con la app por medio de accesorios, como lo son los botones, labels, pictureBox, etc. Por medio de los botones permite cargar los archivos de los dos grafos que se van a analizar, por medio de un botón ejecuta la sección de código que se encarga de determinar si los grafos son isomorfos o no, y muestra el resultado usando dataGridView o MessageBox, además se le agregó una función para que por medio de Picture Box muestre una representación gráfica de los grafos ingresados.
2. **Lector de grafos:** Se debe encargar de leer de los archivos de texto y de ellos obtener la información de cada grafo, la información que contienen los archivos son: número de vértices y la configuración de cómo se conectan entre sí los vértices.
 - **Solución:** Haciendo uso de POO se creó una clase llamada CargarArchivo que posee métodos que permiten ingresar un archivo que el usuario elija, luego recorre el archivo obteniendo la información del grafo, si en caso el archivo no posee el formato correcto, genera error al hacer la carga, esta información obtenida es almacenada en una clase llamada Grafo y dentro de esta clase existe una lista del tipo Vértice (Vértice es una clase creada que permite almacenar la información el ID del vértice y con que otros vértices conecta) y otra lista del tipo Arista (Arista es una clase creada que permite almacenar el vértice de origen y a que vértice se dirige), en estas listas se almacena la información obtenida de los archivos de texto.
3. **Analizar grafos:** Se encarga de recorrer la nuevamente la información que esta almacenada en las clases Grafo, y va realizando comparaciones, la primera comparación es en la cantidad de vértices de cada grafo, la segunda comparación es en la cantidad de aristas de cada grafo, la tercera comparación es en el grado de cada vértice, si las anteriores comparaciones son iguales entonces se dice que existe la posibilidad de que los grafos sean isomorfos. Por lo que se pasa a analizar si los grafos poseen una función de isomorfismo, de ser esto verdadero se muestra que los grafos son isomorfos y se muestran las posibles funciones de isomorfismo, de lo contrario se mostrara porque no son isomorfos los grafos.
 - **Solución:** Se creó una clase isomorfismo que entra en ejecución cuando el usuario presiona el botón encargado de generar la ejecución en el entorno gráfico, esta clase isomorfismo posee métodos que realizan las comparaciones mencionadas en el problema, para ello se hizo uso de la función foreach, e if's para realizar los recorridos de los grafos e ir comparando contra la información de un grafo contra otro grafo. Al llegar a determinar una función de isomorfismo se realizan otras operaciones. Para mostrar el resultado, se utilizan MessageBox para mostrar con un mensaje lo que ha sucedido, si fue isomorfo y si no lo fue, muestra las razones por la cual no.
4. **Determinar función de isomorfismo:** Entra en ejecución cuando las comparaciones anteriores se cumplieron, por lo tanto, es necesario analizar si los grafos poseen una función de isomorfismo.
 - **Solución:** Para la solución se implementaron varias clases, la primera llamada PermutadorVertice, la segunda PermutadorArista, la función de estas clases es obtener todas las combinaciones posibles de vértices y aristas, llamándose así mismo recursivamente, para luego ir comparando contra el otro grafo para verificar si existen similitudes, en este caso de existir similitudes, es posible que exista una función de isomorfismo, por lo que pasa a apoyarse de la clase Matriz que se encarga de generar listas de vértices adyacente y luego comparar con la lista del otro grafo para determinar si existe una función de adyacencia entre los grafos, de ser esto verdadero se dice que existe una función de isomorfismo, por lo tanto los grafos son isomorfos y al usuario se le muestra en un DataGridView las funciones de isomorfismo encontradas.
 - * **Nota:** el proyecto por el momento genera todas las funciones de isomorfismo para un número de vértices menor o igual a 9. Para una cantidad mayor, se genera únicamente una. El proyecto aún está en proceso de construcción y si desea contribuir puede hacerlo al siguiente repositorio: <https://github.com/MynorXico/ProyectoIsomorfismo> en el que se encuentra toda la información y código fuente del proyecto.

5. **Dibujar grafos ingresados (*):** Como grupo se decidió agregar una función a la aplicación para que además de mostrar si son isomorfos o no los grafos, el usuario tenga la opción de ver una representación gráfica de los grafos ingresados. Es importante mencionar que la app dibuja los grafos basándose en colocar los vértices en una forma circular, para luego conectar los vértices según indique la información cargada de los archivos de texto. Para dibujar aristas que van del mismo vértice de origen al mismo vértice de llegada, se hizo utilizando otro color y cambiando el grosor de la línea de esta arista

➤ **Solución:** Utilizando mapas de bits y la clase Graphics de C# se pudo representar el grafo en un PictureBox, para ello se utilizó una clase llamada Coordenadas, que almacena la coordenada tanto en el eje X como en el eje Y de un vértice. Para determinar en qué posición del PictureBox estaría cada vértice, se utilizó la idea de dibujar los vértices en un círculo, para esto se determinó el radio en base a la altura del PictureBox, luego se determinó el centro del PictureBox tanto en X como en Y, luego se dividió 360 entre la cantidad de vértices del grafo, esto indicaba el grado en el cual debía ser dibujado el grafo, con el ángulo y el radio por medio de ley de senos se determinó la coordenada de Y y de X respectivamente. También se hizo uso de la librería Math, ya que era necesario usar la función matemática Sin(), además de la constante PI para hacer transformaciones de grados a radianes.

La función se habilita por medio de un botón que se encuentra en el entorno gráfico, el usuario puede decidir si ver o no la representación gráfica del grafo.

(*): Esta parte han sido extras que se le decidió agregarle al proyecto

5 EXAMINAR LA SOLUCIÓN OBTENIDA

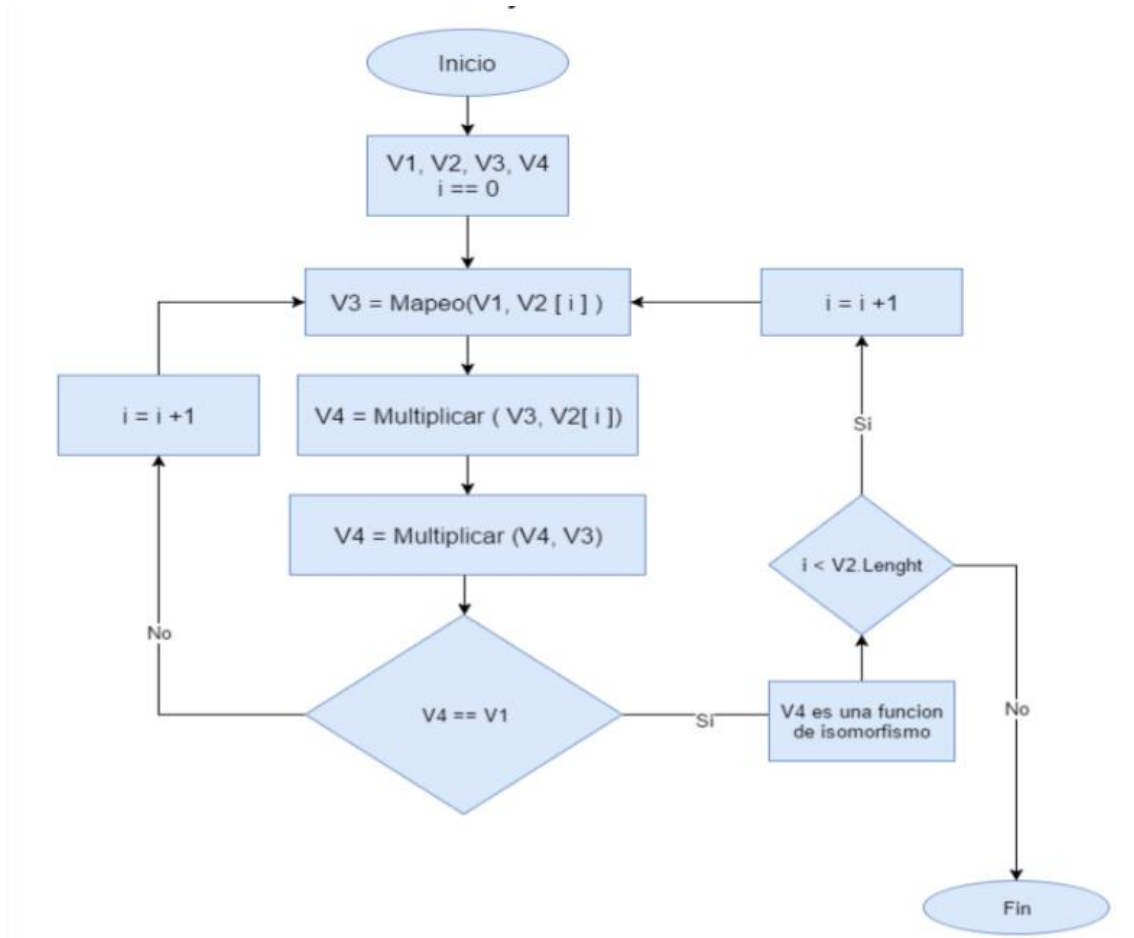
Una vez teníamos ya todo finalizado era necesario generar todos los posibles casos y escenarios para verificar respuestas, para este paso fue necesario usar el libro de “Matemática Discreta y Combinatoria” del autor Ralph Grimaldi, específicamente en el capítulo 11 que es donde se toca el tema de grafos. Tomamos varios grafos de ese libro en la parte de isomorfismo y comparamos nuestra respuesta con la que el libro plantea.

DIAGRAMA DE CLASES

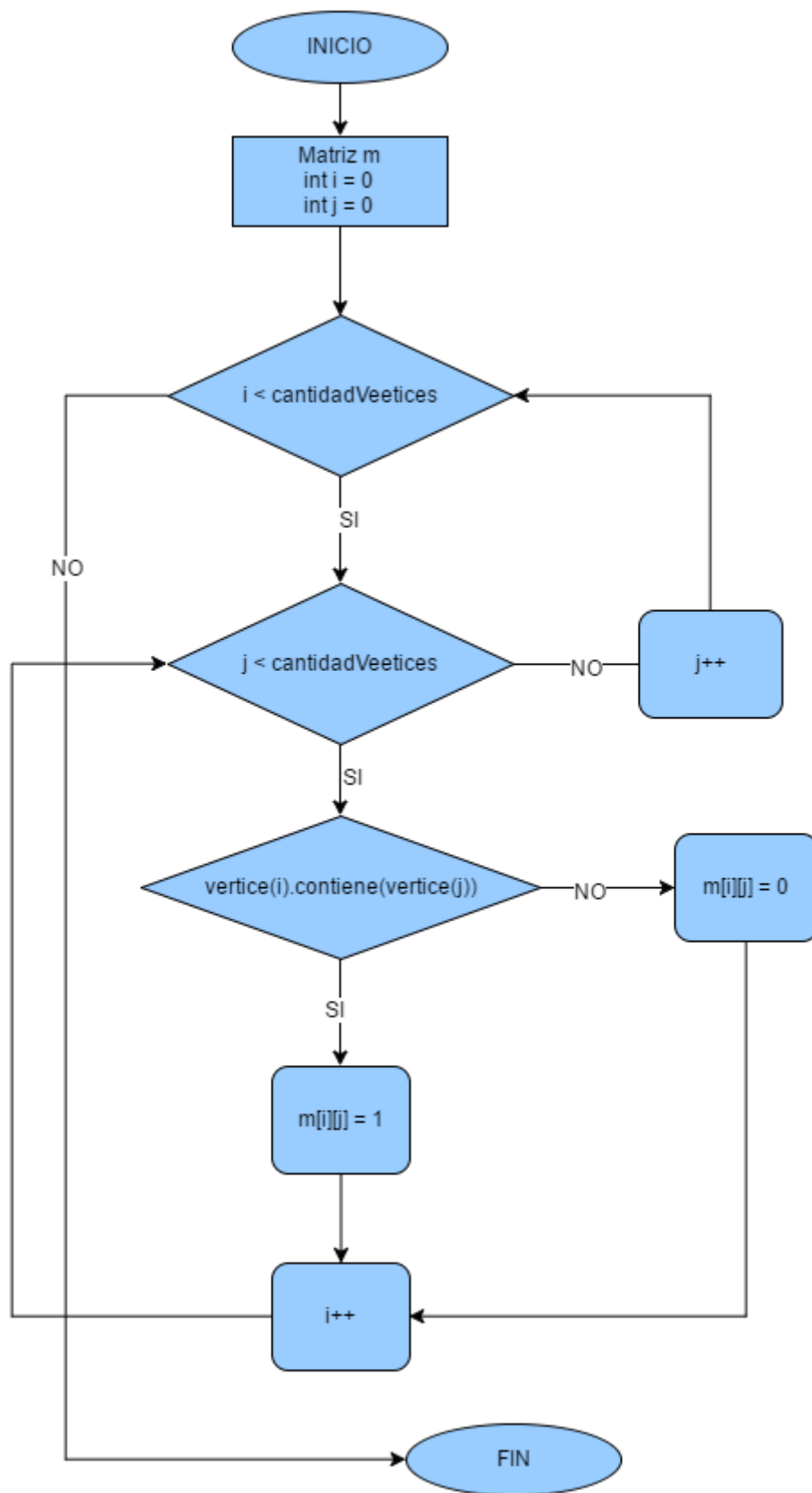
*El diagrama de clases se adjunta a los manuales Técnicos y de Usuario por motivos de conservación de la resolución de la imagen.

DIAGRAMAS DE FLUJO

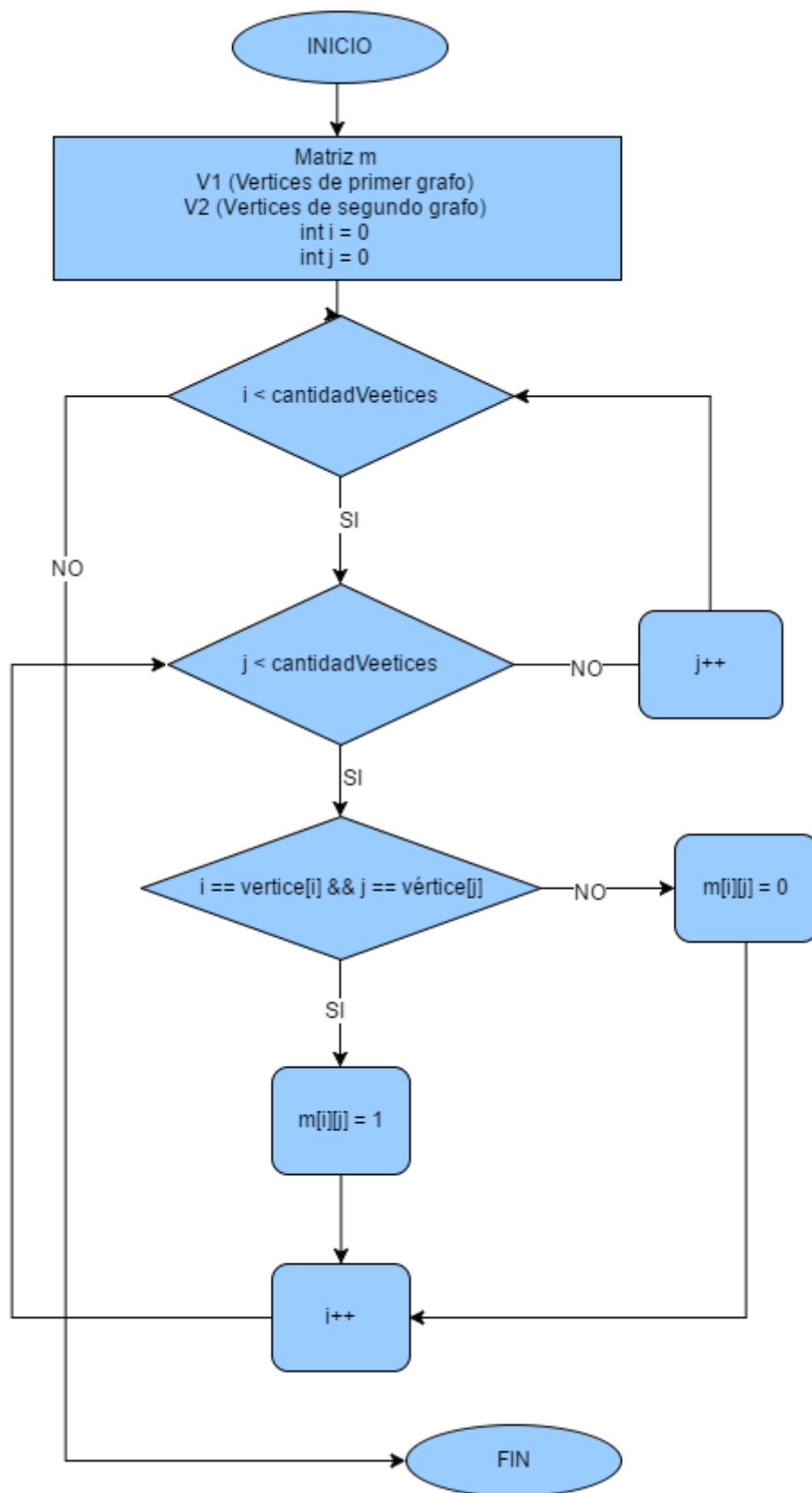
Buscar función de isomorfismo



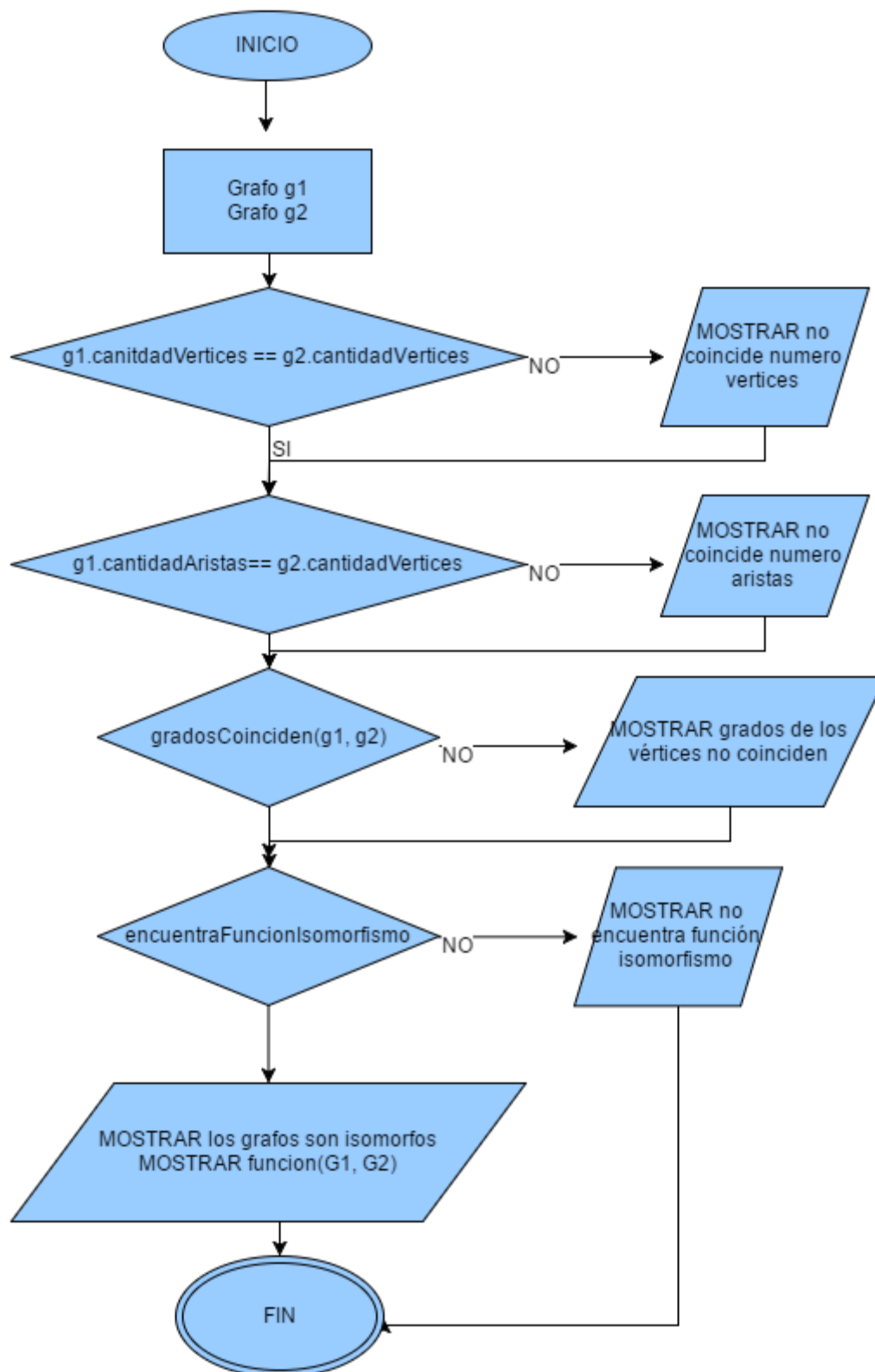
Generar Matriz de Adyacencia



Generar Matriz de Mapeo



Secuencia General del Programa



DESCRIPCIÓN DE LAS CLASES

✓ CLASE ARISTA

En esta clase se guarda información básica de cada arista como lo son:

- ID
- From (De donde viene)
- To (Hacia donde va)

✓ CLASE CARGA DATOS

Se encarga de realizar todo el proceso de carga de datos y creación de los grafos y los vértices y aristas contenidos en cada uno de ellos.

✓ COORDENADAS

Esta clase se encarga del modelado de las coordenadas rectangulares de cada uno de los vértices en un plano.

✓ GRAFO

Contiene las propiedades que tendría cualquier grafo:

- Cantidad de aristas
- Cantidad de vértices
- Lista de vértices
- Lista de aristas

También contiene algunos métodos:

- Generar matriz de adyacencia
- Obtener un vértice según la etiqueta

✓ ISOMORFISMO

Contiene todas las validaciones necesarias para determinar si dos grafos son o no isomorfos.

Las verificaciones que realiza son:

- ¿Tienen la misma cantidad de vértices?
- ¿Tienen la misma cantidad de aristas?
- ¿Existe alguna coincidencia de grados de vértices entre los grafos?
- ¿Existe alguna función de isomorfismo entre los grafos?

✓ MATRIZ

Contiene únicamente dos cosas.

- Una lista de lista de enteros.
- Un método para verificar que otra matriz es idéntica a ella.

✓ OPERACIONES MATRIZ

Contiene métodos útiles para operar las matrices

- Multiplicación de hasta tres matrices.
- Obtener columna por número
- Obtener el producto punto entre dos vectores
- Obtener la transposición de una matriz

- Generación de una matriz de mapeo posible para encontrar la función de isomorfismo.

✓ VERTICE

Esta clase contiene las algunas propiedades y métodos características de los vértices en un grafo.

- Propiedades:
 - ✓ Etiqueta
 - ✓ Vértices conectados
 - ✓ Grado
 - ✓ ID
- Métodos
 - ✓ Asignar ID
 - ✓ Calcular Grado
 - ✓ Lista de vértices por etiqueta
 - ✓ Verificar si un vértice ya existe en el grafo

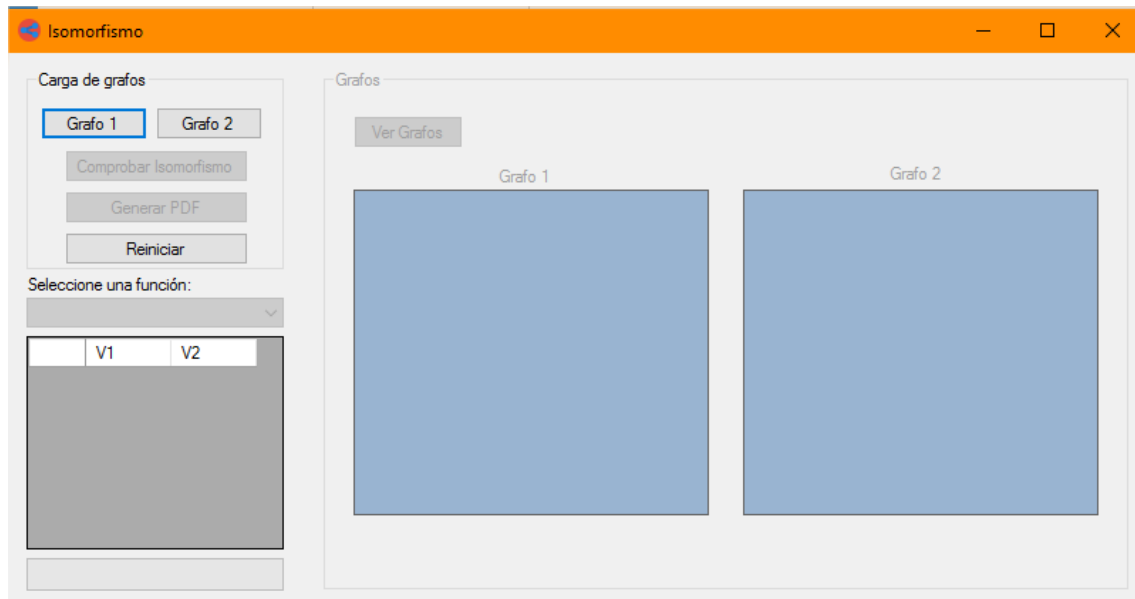
✓ PERMUTADOR

Contiene los métodos encargados de generar todas las permutaciones de vértices dada una lista de vértices.

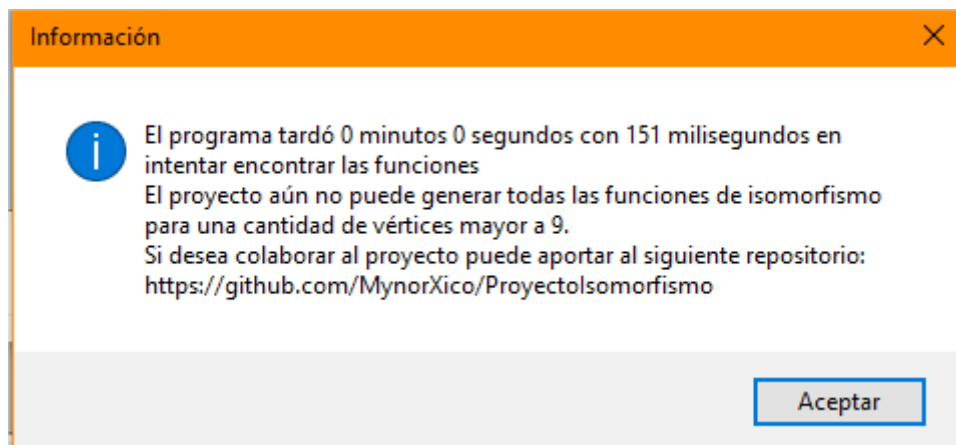
✓ PERMUTADOR UTILITIES

Contiene métodos auxiliares de la clase Permutador y también es el encargado de agregar las permutaciones a una nueva lista que guardará las todas las listas de vértices permutadas.

PANTALLA DE INICIO



CUADRO DE DIÁLOGO INDICANDO EL TIEMPO QUE TARDÓ EN ENCONTRAR LA FUNCIÓN



VISUALIZACIÓN DE LOS GRAFOS

Isomorfismo

Carga de grafos

Grafo 1 Grafo 2

Comprobar Isomorfismo

Generar PDF

Reiniciar

Seleccione una función:

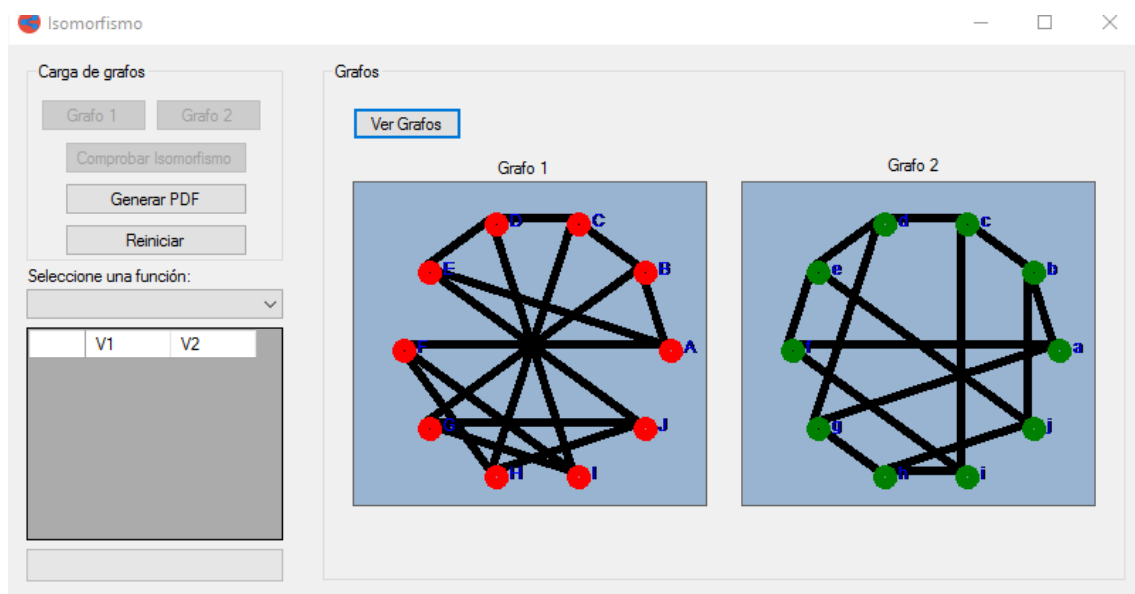
V1 V2

Grafos

Ver Grafos

Grafo 1

Grafo 2



VISUALIZACIÓN DE FUNCIÓN DE ISOMORFISMO

Isomorfismo

Carga de grafos

Grafo 1 Grafo 2

Comprobar Isomorfismo

Generar PDF

Reiniciar

Seleccione una función:

Funcion #1

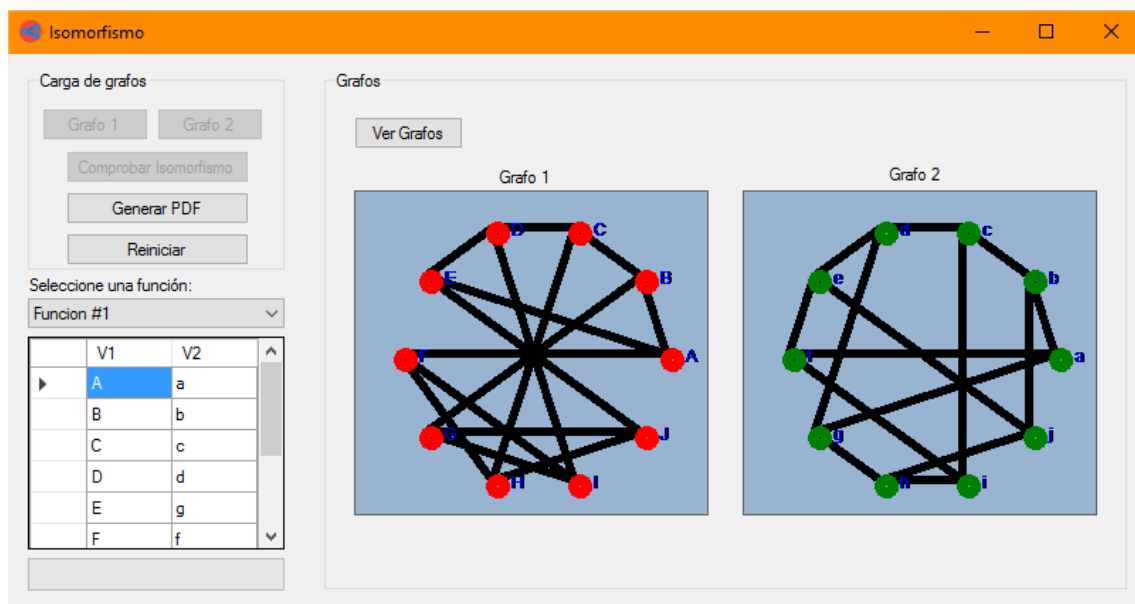
	V1	V2
▶	A	a
	B	b
	C	c
	D	d
	E	g
	F	f

Grafos

Ver Grafos

Grafo 1

Grafo 2



Lista de funciones isomórficas - Matemática Discreta II

Función Isomórfica No. 1

Grafo 1	Grafo 2
A	a
B	b
C	c
D	d
E	g
F	f
I	e
H	i
G	j
J	h

CONCLUSIONES

- ✓ El programa cumple con todos los objetivos propuestos al inicio de este documento (comprobación de isomorfismo entre grafos).
- ✓ El programa de Isomorfismo-Discreto, es una herramienta que permite realizar varias tareas relacionadas a la información proveída de los grafos.
- ✓ La aplicación correcta de la teoría de grafos fue esencial para optimizar el tiempo de ejecución de los algoritmos acá presentados.
- ✓ Mientras los grafos tienen más vértices, es mayor el tiempo de espera para saber si dos grafos son isomorfos debido a que las permutaciones crecen por $n!$ siendo n el número de vértices. Debido a este problema, se aplicó un algoritmo eficaz a tal magnitud de permutaciones. Este algoritmo evalúa directamente cada permutación de vértices haciendo que se compare directamente con la matriz de adyacencia del otro grafo. Al encontrar una permutación válida, termina el algoritmo.

BIBLIOGRAFÍA

- Discrete and Combinatorial mathematics: and applied introduction. Addison-Wesley, Ralph Grimaldi. 1994
- Graphs and algorithms. Wiley. 1984
- The Stanford GraphBase: a platform for combinatorial computing. ACM Press. 1993
- Matrix algebra. Cambridge University Press. 2005
- Applications of combinatorial matrix theory to Laplacian matrices of graphs. CRC Press. 2012