

Universidad Rafael Landívar  
Facultad de Ingeniería  
Licenciatura en Ingeniería en Informática y Sistemas  
Matemática Discreta II  
Catedrático: Ing. Juan Carlos Soto

S-2

**“Proyecto  
Isomorfismo”**

Estrada Rodríguez Marcela Margarita 1010419  
De León Chang José Daniel 1170419  
Villeda Navarro Estuardo José 1003519  
Roches Revolorio Marlon Andrés 1250918

Guatemala, 4 de noviembre del 2019

# Introducción

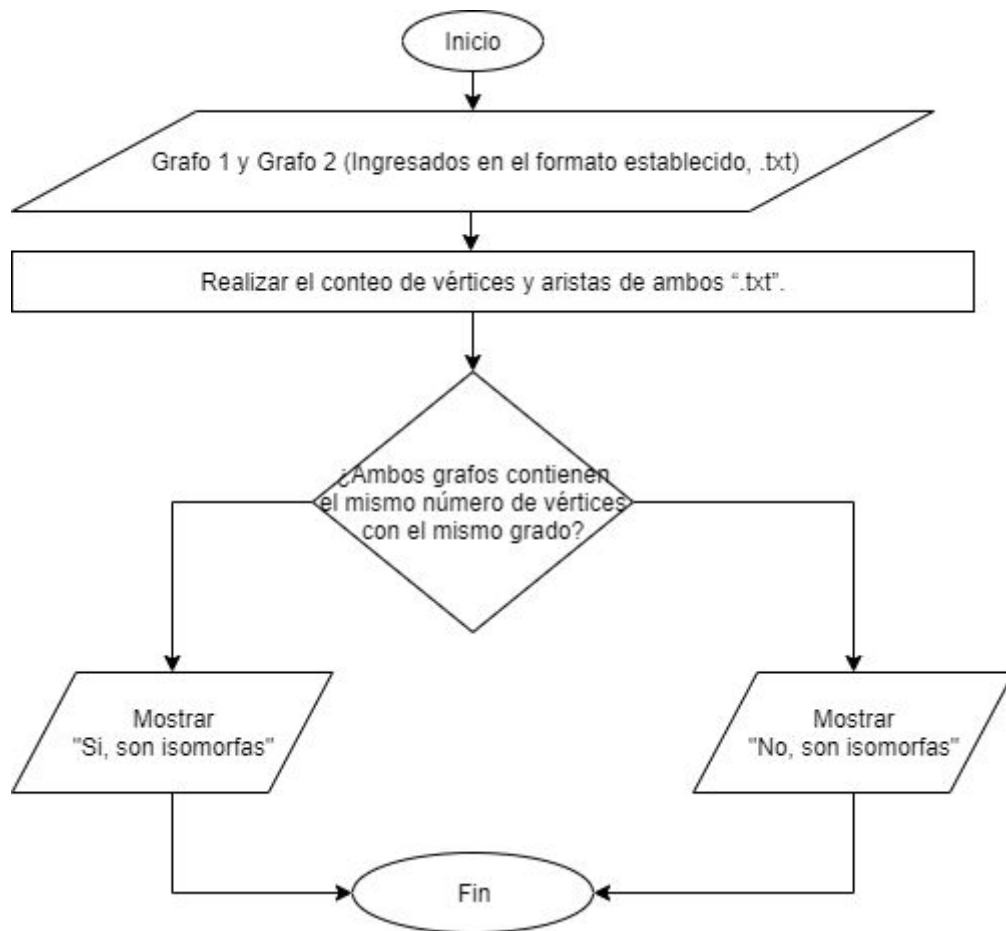
En el siguiente documento se dan a conocer detalles fundamentales sobre el proyecto de la signatura “Matemática Discreta II”, el cual consta de una aplicación de consola en c#, la cual consta del ingreso de 2 archivos .txt en un formato específico para luego descomponerlos por la información que contienen en objetos que son representaciones del grafo que contienen por coordenadas y vértices.

De esta manera se compararan ambos grafos en vértices, aristas y grado de vértices para determinar si son isomorfos entre ellos de una manera rápida y eficaz sin necesidad de graficar de alguna manera, solamente obteniendo la meta data del grafo obtenida por el txt.

## Análisis

Entradas	Procesos	Salidas	Restricciones
Dos grafos con sus respectivos vértices y aristas, ingresadas en un documento de texto. (Dos archivos .txt distintos)	<ol style="list-style-type: none"><li>1. Lectura De lineas y separacion de datos.</li><li>2. Validaciones de propiedades mediante un deccionario en c#.</li><li>3. Comparacion de los dos grafos vitrualizados en objetos para trabajar con los datos del txt y validar isomorfismo.</li></ol>	Verificacion si el grafo es isomorfo o no. Diferencias de por qué no son isomorfos..	Formato de entrada del txt. Ingreso del txt para leerlo.

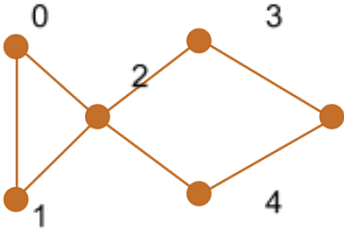
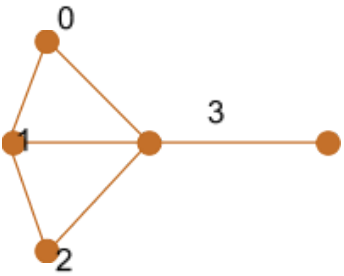
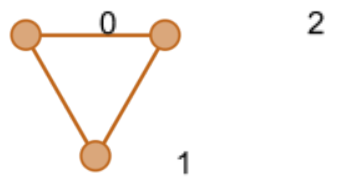
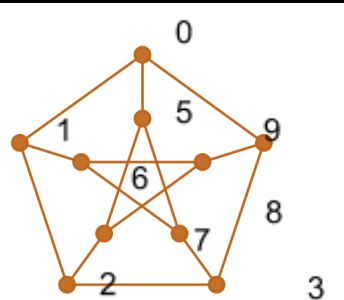
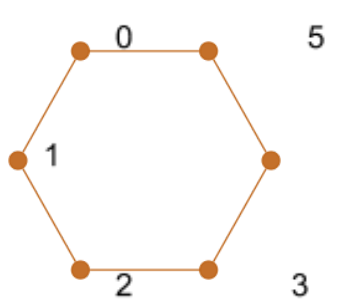
## Diagrama de flujo

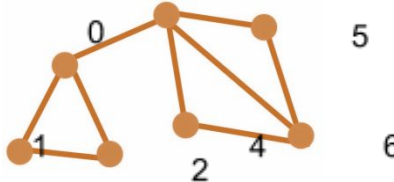
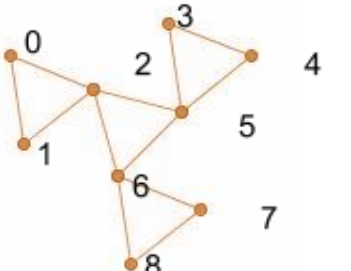
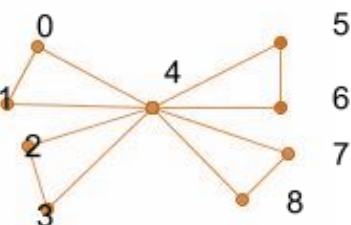
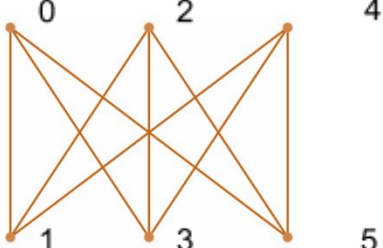
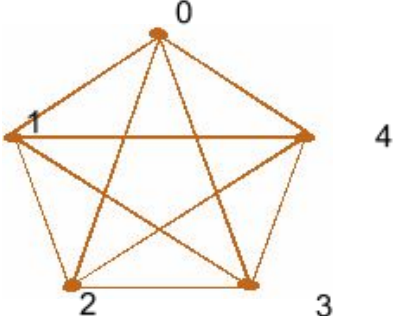
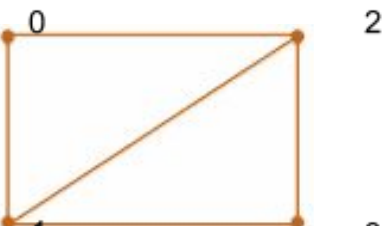


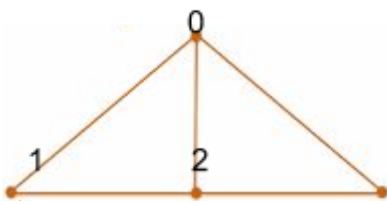
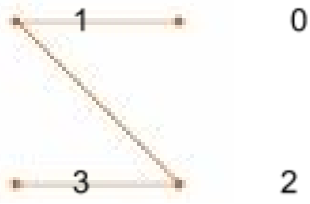
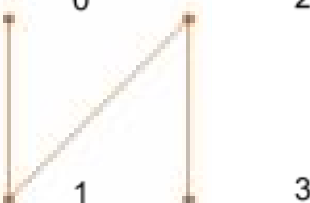
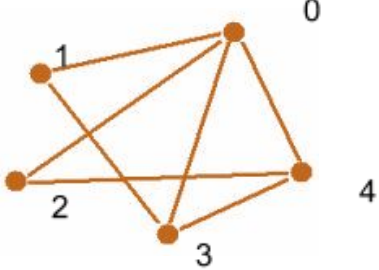
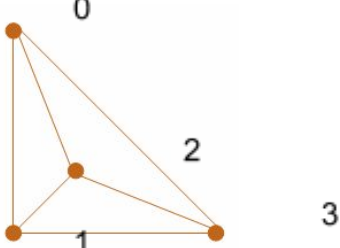
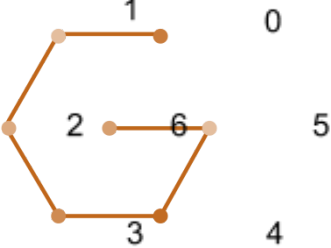
Para un diagrama de flujo más detallado, visitar el siguiente link:

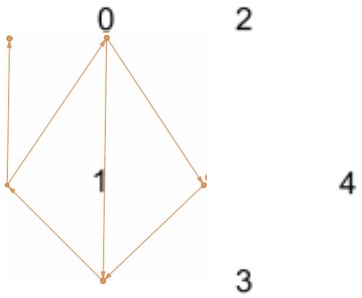
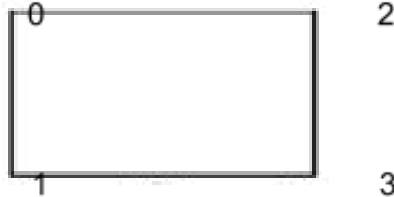
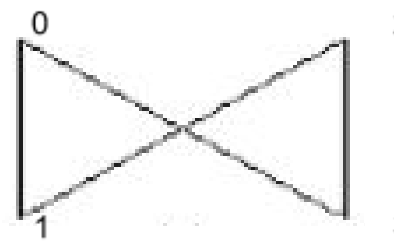
<https://drive.google.com/file/d/1DcDPh6NPESlyinWRfCWabU3M3-Q2tzd8/view?usp=sharing>

# Grafos a evaluar

#	Imagen	# de vértices	# de vértice inicial y final
1		6	0,1 0,2 1,2 2,3 2,4 3,5 4,5
		5	0,1 0,3 1,2 1,3 2,3 3,4
2		3	0,1 0,2 1,2
		10	0,1    3,4 0,5    4,9 0,4    5,7 1,6    5,8 1,2    6,9 2,7    6,8 2,3    7,9 3,8
3		6	0,5 0,1 1,2 2,3 3,4 4,5

		7	0,1    3,5 0,2    3,6 0,3    4,6 1,2    5,6 3,4
4		9	0,1    3,5 0,2    5,6 1,2    6,8 2,5    6,7 2,6    7,8 3,4    4,5
		9	0,1    4,5 0,4    4,6 1,4    5,6 2,4    4,7 2,3    4,8 3,4    7,8
5		6	0,1    2,3 0,3    2,5 0,5    4,1 2,1    4,3 4,5
		5	0,1    1,4 0,2    1,3 0,3    1,2 0,4    2,4 2,3    3,4
6		4	0,1 0,2 1,2 1,3 2,3

		4	0,1 0,2 0,3 1,2 2,3
7		4	0,1 1,2 2,3
		4	0,1 1,2 2,3
8		5	0,1    1,3 0,2    2,4 0,3    3,4 0,4
		4	0,1 0,2 0,3 1,2 1,3 2,3
9		7	0,1 1,2 2,3 3,4 4,5 5,6

		5	0,1 1,2 1,3 2,3 2,4 3,4
10		4	0,1 0,2 1,3 2,3
		4	0,1 0,3 1,2 2,3

## Batería de grafos

Pareja #	# de vértices	# de aristas	¿Es isomorfa?
1	6	7	No
	5	6	
2	3	3	No
	10	15	
3	6	6	No
	7	9	
4	9	12	Si
	9	12	
5	6	9	No
	5	10	
6	4	5	Si
	4	5	
7	4	3	Si
	4	3	
8	5	7	No
	4	6	
9	7	6	No
	5	6	
10	4	4	Si

	4	4	
--	---	---	--

## Pseudocódigo

```

class Program
{
    Main()
    {
        bool menu = true;
        while (menu)
        {
            //leemos ruta del primer grafo
            string ruta = LeerLinea();
            //Accedemos al archivo y lo leemos completo
            var File1 = new StreamReader(ruta);

            //leemos ruta del primer grafo
            ruta = LeerLinea();
            //Accedemos al archivo y lo leemos completo
            var File2 = new StreamReader(ruta);
            //separar por saltos de linea
            //Le enviamos un diccionario Vacio

            var Grafo1 = ClasSiicar(File1.LeerHastaElFinal.SepararPor("\n"), new Diccinario());
            var Grafo2 = ClasSiicar(File2.LeerHastaElFinal.SepararPor("\n"), new Diccinario());
            //si tiene los mismo vertices
            Si (MismosVertices(File1.ReadLine(), File2.ReadLine()) y MismasAristas(Grafo1,
Grafo2) y MismoGrado(Grafo1, Grafo2))
            {
                //si son isomorfos
            }
            else
            {
                //no lo son
            }

            string monitor = LeerLinea();
            Si ( monitor == "N")
            {
                //seguir validando
                menu = false;
            } else
            {

```



```
//terminar sesion
menu = true;
}
```

```
}
Console.ReadLine();
```

```
bool MismosVertices(string VerticesGRafo1, string VerticesGRafo2)
{
    Si (int.Parse(VerticesGRafo1) dSierente a int.Parse(VerticesGRafo2))
    {

        Devuelve false;
    }
    else
```

```
{
    Devuelve true;
}
}
```

```
bool MismasAristas(Diccionario Grafo1, Diccionario Grafo2)
{
    bool monitor = true;
    //comparamos Vactores
    ParaCada (var item in Grafo1)
    {
        Si (Grafo1[item.Key].SepararPor(',') dSierente a
Grafo1[item.Key].SepararPor(','))
        {
            monitor = false;
            break;
        }
    }
    Devuelve monitor;
}
```

```
bool MismoGrado(Diccionario Grafo1, Diccionario Grafo2)
{
    bool monitor = true;
    ParaCada (var item in Grafo1)
    {
        //comparamos longitud
```

```

        Si (Grafo1[item.Key].SepararPor(',').Length dSierente a
Grafo1[item.Key].SepararPor(',').Length)
        {
            monitor = false;
            break;
        }
    }
    Devuelve monitor;
}

Diccionario ClasSiicar(string[] array, Diccionario Diccionario)
{
    //para cada arreglo (valor en el array extraido del txt)
    ParaCada (var item in array)
    {
        //Separacion por ","
        var aux = item.SepararPor(',');
        Si (item.SepararPor(',').Length dSierente a 1)
        {
            //si el diccionario no contiene el vertice al cual agregar
            Si (!Diccionario.ContainsKey(aux[0]))
            {
                //agrega el nuevo indice con la aarista correspondiente
                Diccionario.Agregar(aux[0],aux[1]);
            }
            else
            {
                //agrega la arista al vertice correspondiente
                Diccionario[aux[0]]+="${aux[1]}";
            }
        }
    }
    Devuelve Diccionario;
}
}
}

```