UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

EDUCACIÓN SUPERIOR

Fundamentos de Estructuras Computacionales

Ingeniería en Computación Inteligente

PROYECTO FINAL

GRAFOS BIPARTITOS

Nombre de la maestra: Dra. Aurora Torres Soto.

Nombres de los alumnos:

José Luis Sandoval Pérez ID: 261731

Diego Emanuel Saucedo Ortega ID:261230

Carlos Daniel Torres Macías ID:244543

Ximena Rivera Delgadillo ID:261261

Fecha de entrega: Miércoles 8 de diciembre del 2021.

ÍNDICE:

INTRODUCCIÓN:

En matemáticas y en ciencias de la computación, la teoría de grafos (también llamada teoría de las gráficas) estudia las propiedades de los grafos. Donde se define a un grafo como un conjunto, no vacío, de objetos llamados vértices (nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no.

Un grafo se representa mediante una serie de puntos que nombramos vértices conectados por líneas llamadas aristas.

Todo grafo simple puede ser representado por una matriz, que llamamos matriz de adyacencia. Se trata de una matriz cuadrada de n filas x n columnas (siendo n el número de vértices del grafo). Para construir la matriz de adyacencia, cada elemento a_{ij} vale 1 cuando haya una arista que una los vértices i y j. En caso contrario el elemento a_{ij} vale 0. La matriz de adyacencia, por tanto, estará formada por ceros y unos.

En teoría de grafos, un grafo bipartito es un grafo G= (N, E) cuyos vértices se pueden separar en dos conjuntos disjuntos U y V. De manera que las aristas sólo pueden conectar vértices de un conjunto con vértices del otro. Los grafos bipartitos suelen representarse gráficamente con dos columnas (o filas) de vértices y las aristas uniendo vértices de columnas (o filas) diferentes.

Los dos conjuntos U y V pueden ser pensados como un coloreo del grafo con dos colores: si pintamos los vértices en U de azul y los vértices de V de verde obtenemos un grafo de dos colores donde cada arista tiene un vértice azul y el otro verde. Por otro lado, si un gráfico no tiene la propiedad de que se puede colorear con dos colores no es bipartito.

Dicho así en la siguiente narración académica nos gustaría establecer los puntos principales que tomamos en cuenta para el desarrollo de nuestro trabajo final que entorna como temas principales La Teoría de Grafos y más específicamente los Grafos Bipartitos.

A lo largo de este primer semestre de nuestra carrera Ingeniería en Computación Inteligente; la materia Fundamentos de las Estructuras Computacionales, nos llevó a plantear problemáticas lógicas desde una perspectiva más analítica permitiéndonos así experimentar con los conocimientos adquiridos y desarrollar estas mismas problemáticas desde el campo de la programación. Así como en el caso presentado a continuación.

Como proyecto final de la materia se nos presentó como problemática desarrollar un programa tal que nos permitiera identificar si un grafo ingresado es o no bipartito a través de su Matriz de Adyacencia. Problemática que radica dentro de la Lógica de Programación y la Teoría de Grafos, específicamente La Bipartición de un Grafo. Esto nos llevó a estudiar los conceptos anteriormente mencionados de manera exhaustiva para la total compresión de este proyecto.

Así llegamos a un punto principal para partir del desarrollo del programa que es que sabemos que para que exista una Bipartición en un Grafo tiene que haber dos conjuntos de vértices independientes dentro del mismo. Y si como objetivo principal del programa tenemos que facilitar el proceso de búsqueda de un grafo bipartito y determinar su bipartición sin necesidad de hacer un trazo mayor o más gráfico que la propia matriz de adyacencia para su búsqueda y entendimiento.

Entonces el problema principal es llegar a una deducción lógica de qué es lo que distingue la matriz de adyacencia de un grafo

bipartito a la de cualquier otro tipo de grafo, y llegando a la misma es posible posteriormente pensar en una solución que

se pueda implementar en un algoritmo para programar.

A través de las siguientes páginas se podrá conocer más a fondo de qué manera se llegó a la resolución de la problemática planteada y finalmente como fue plasmada para la formación de nuestro programa en ANSI C presentado como Proyecto Final de esta materia.

DESCRIPCIÓN DEL PROBLEMA:

Como desarrollo de proyecto debemos realizar un programa que a partir de una matriz de adyacencia ingresada manualmente por el usuario con un límite máximo de veinte nodos; logre identificar si esta es o no correspondiente a un grafo bipartito, en caso de serlo va a regresar al usuario, la distinción de los dos conjuntos correspondientes de forma v1 y v2.

Primeramente, el usuario debería de ingresar el número de nodos que contendría el grafo, que, a su vez, sería el tamaño de la misma matriz, el cual estará limitado a un máximo de veinte nodos. Una vez establecido el tamaño de la matriz (del grafo), el usuario tendría que llenar la misma con la guía del programa, que solicitaría cada número de vértices en cada nodo, sin repetir los nodos, es decir, si solicita el nodo (1,2), después no va a preguntar por el nodo (2,1), esto con el fin de hacer más optimo, sencillo y rápido el programa.

Una vez que la matriz esté llena, el programa empieza a delimitar si es bipartito o no. Esto se verá más a detalle en el apartado de *Descripción de la solución.*

El problema principalmente es llegar a una deducción lógica de qué es lo que distingue la matriz de adyacencia de un grafo bipartito a la de cualquier otro tipo de grafo, llegando a una deducción lógica del mismo es posible pensar en una solución que se pueda implementar en un algoritmo.

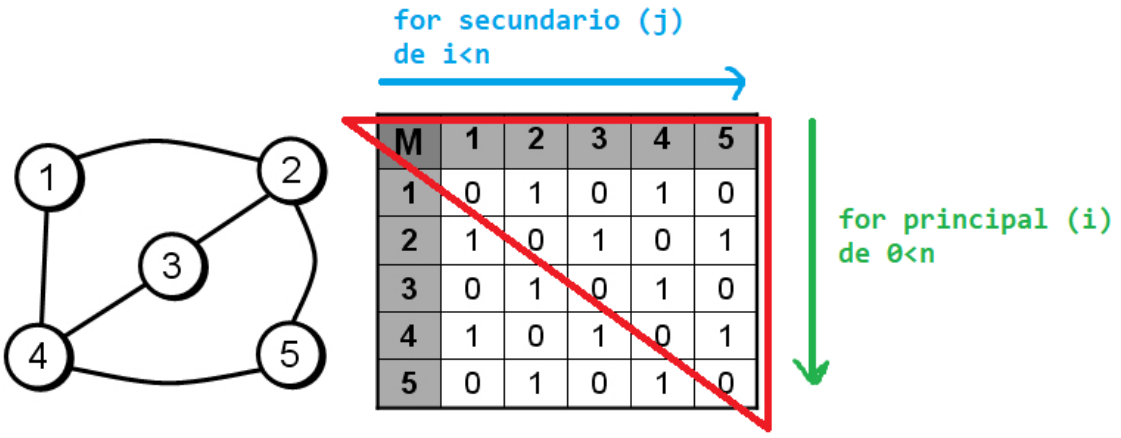
El programa, una vez teniendo la solución y la respuesta sobre si la matriz de adyacencia (el grafo) ingresado por el usuario debe de regresar la matriz en forma, así como una respuesta a la cuestión de si el grafo es bipartito o no y, por último, una distinción entre ambas partes del grafo, que serán representadas por los vértices que conforman cada una de estas partes.

Aunque el programa esté limitado a veinte nodos por matriz, con la misma lógica que implementamos en la resolución, se deberían de poder resolver si son bipartitos o no, sin distinción de cuantos nodos tiene el grafo, pues cómo mencione antes, la parte complicada es resolver de forma lógica la distinción de los grafos mediante sus matrices de adyacencia.

DESCRIPCION DE LA SOLUCIÓN DEL PROBLEMA:

El primer paso es construir la matriz de adyacencia por parte del usuario. La principal característica de las matrices de adyacencia es que son simétricas, por lo tanto, no era necesario permutar todos los nodos en busca de su adyacencia, sino combinar entre nodos y asignar el mismo valor en un sentido u otro. De forma que solo se necesita que el usuario ingrese la mitad de la matriz para posteriormente, completarla.

Incluimos un for principal que recorra desde el inicio de la matriz hasta el final, esta es la encargada de preguntar por "el nodo principal". Anidado, incluimos un segundo for que recorra desde el nodo actual hasta el final, ¿Por qué no se recorre desde el principio?, como pensamos preguntar una sola vez por la adyacencia de cada nodo, el primero tendrá la posibilidad de ser "combinado" con todos los nodos (Hasta el mismo por la posibilidad de que el grafo cuente con lazos), el siguiente tendrá una pequeña restricción, y es que ya fue "combinado" en una ocasión con el elemento anterior, y sucesivamente hasta terminar con los nodos, y el último solo pueda ser "combinado" consigo mismo. De forma que solo es necesario combinar con nodos "mayores iguales" al actual.



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Y así dentro del for anidado almacenamos en nuestra matriz el mismo valor para la adyacencia [i][j] y [j][i]. El fragmento del Código luce algo así:

```
for (int i=0; i<n; i++) {
        for (int j=i; j<n; j++) {
            int p;
            printf("Nodo[%d] y Nodo[%d], ¿Tienen relación?: ", 
i+1, j+1);
            scanf ("%d", &p);
            if(p==1) {
                ady[i][j] =1.
                ady[j][i] =1.
            }
        }//fin for i
}//fin for j
```

Lo más complicado fue idear una forma de ir descartando los casos en los que un grafo ya no era bipartito, y aún más complicado, que la solución también nos pudiese dividir el grafo en los dos conjuntos de nodos.

Después de exhaustivas sesiones de ideas, se nos ocurrió una forma "sistemática" de ordenar los nodos en dos grupos; al preguntar por la adyacencia de un nodo iríamos acomodando un nodo en cada grupo, sin embargo, había que considerar algunas situaciones:

- Hay que buscar si el nodo principal existía en el primer grupo, para acomodar el nodo secundario en el segundo.
- Hay que buscar si el nodo principal existía en el segundo grupo, para acomodar el nodo secundario en el primer grupo
- Hay que buscar si el nodo secundario existía en el segundo grupo, para acomodar el nodo principal en el primer grupo

- Hay que buscar si el nodo secundario existía en el primer grupo, para acomodar el nodo principal en el segundo grupo

Puede que estas situaciones sean reiterativas, pero dado a que pueda pasar que solo el nodo principal ya fue ubicado o solo el nodo secundario. En caso de que ninguna de estas situaciones ocurra, significa que ninguno de los nodos ha sido previamente ubicado.

Para ubicar los nodos, utilizamos dos arreglos A y B de hasta 20 elementos cada uno. Para identificar que un nodo se encuentra en un conjunto u otro marcamos como 1 la casilla correspondiente al nodo. El fragmento del código correspondiente a esta parte es:

```
    if(a[i]==1) {b[j]=1;}
    else if(a[j]==1) {b[i]=1;}
    else if(b[i]==1) {a[j]=1;}
    else if(b[j]==1) {a[i]=1;}
    else{a[i]=1; b[j]=1;}
```

Una vez que se han acomodado los nodos entre los dos grupos, para comprobar si es bipartito, habrá al menos un nodo que se encuentre en ambos grupos; es así como utilizamos un for de 0 a n que pregunte si el nodo en cuestión tiene como valor 1 en ambos grupos, en caso de que sea cierto, el nodo no es bipartito.

Además si existe algún nodo que no se encuentre en alguno de los conjuntos, significa que ese nodo no tiene adyacencia con otro nodo, y por tanto el grafo es disconexo. Un grafo disconexo cuenta con nodos aislados a los cuales no se podrá llegar por medio de una arista.

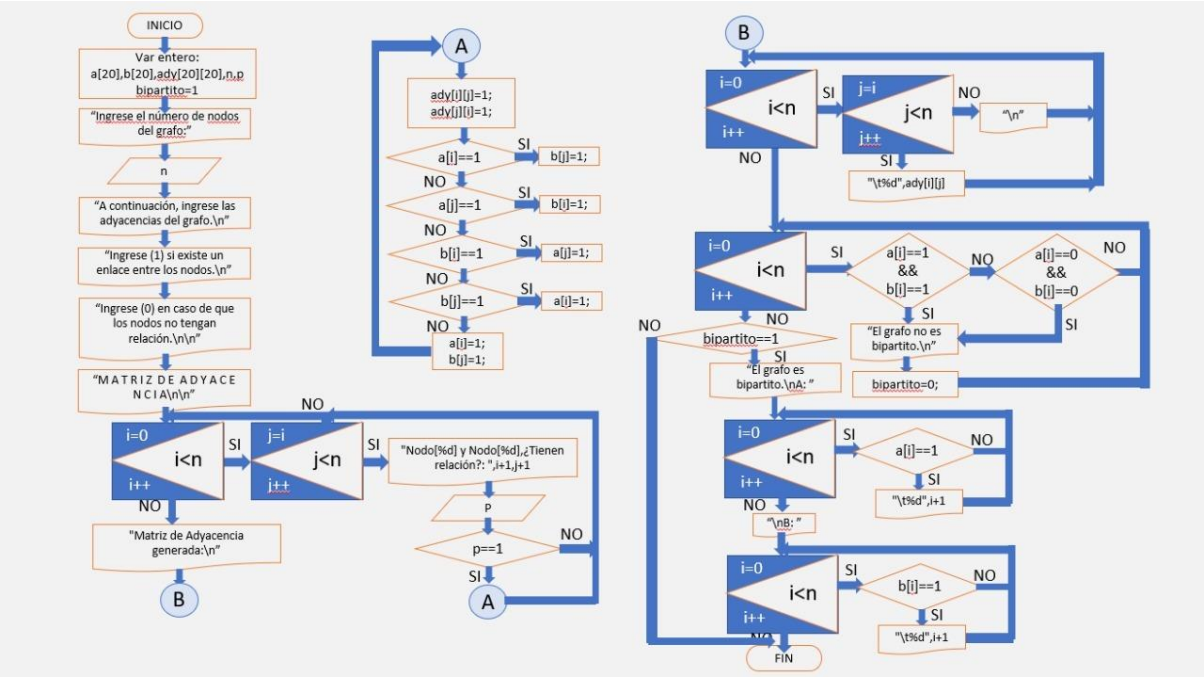
```
for (int i=0; i<n; i++) {
    if(a[i]==1 && b[i]==1) {bipartite=0;}
    if(a[i]==0 && b[i]==0) {bipartito=0;}
}
```

Entonces, si no se encuentra algún nodo con presencia en ambos grupos o algún nodo aislado. Utilizamos un for de 0 a n que imprima los nodos del primer conjunto, y después un for idéntico para imprimir los nodos del segundo conjunto.

```
    printf ("\nEl grafo es bipartito.\nA:");
    for (int i=0; i<n; i++) {
        if(a[i]==1) {printf ("\t%d”, i+1);}
    }

    printf("\nB:").
    for (int i=0; i<n; i++) {
        if(b[i]==1) {printf ("\t%d”, i+1);}
    }
```

DIAGRAMA DE FLUJO:



FUNCIONAMIENTO DEL PROBLEMA:

BiBi graph system

El Sistema Bifuncional para Bipartición de grafos o BiBi, es una herramienta desarrollada en lenguaje C para a partir de las adyacencias de un grafo conocer si se trata de un grafo bipartito.

> *GRAFO BIPARTITO*: Un grafo no dirigido es bipartito si sus vértices pueden repartirse en dos conjuntos disjuntos de tal forma que todas las aristas tengan un extremo en cada uno de esos conjuntos.

C.1. Una bienvenida llena de estrellas.

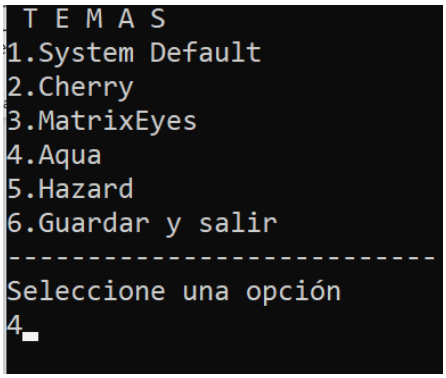
Las primeras interacciones son importantes, es por ello por lo que decidimos llenar su pantalla con estrellas aleatorias al principio de la ejecución de BiBi mediante la función "askyFullofStars". Su aparición es breve, y en un tiempo menor a 2 segundos, podrá comenzar a interactuar con BiBi.

ADVERTENCIA: La función "askyFullofStars" puede no ser compatible con todos los compiladores, lo que podría derivar en impresiones anormales.

C.2. Un tema para cada ocasión.

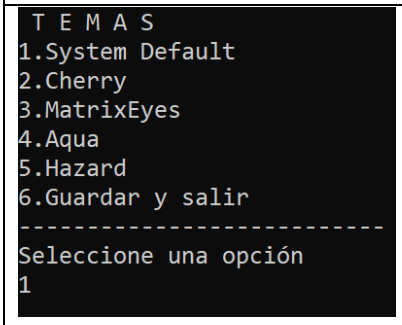
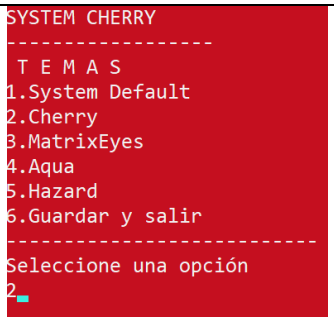
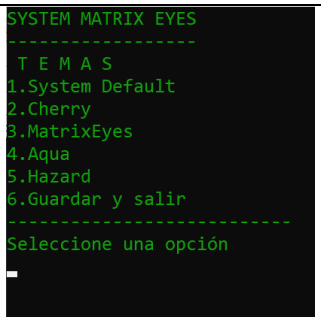
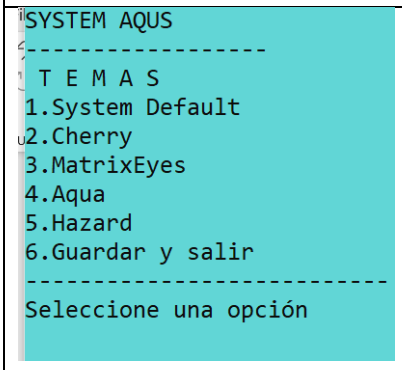
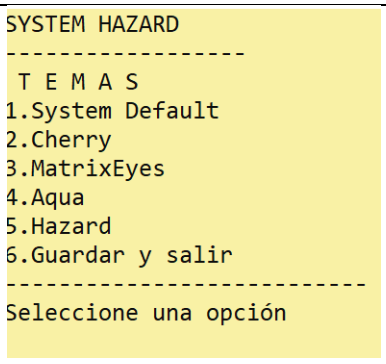
Después de una brillante bienvenida, usted podrá configurar un tema para la consola mediante la función "cambiarTemaConsola". Podrá seleccionar 5 diferentes temas:

1. **Default System:** El tema original de la consola, texto blanco brillante en un fondo oscuro.
2. **Cherry:** Un tema atrevido y renovado, un fondo rojo galante.
3. **MatrixEyes:** Inspirado en una de las sagas cinematográficas más importantes de la época moderna.
4. **Aqua:** Un tema calmo aguamarina, del gusto marino.
5. **Hazard:** "Una aventura es más divertida si huele a peligro".



Para seleccionar un tema, ingrese el índice del tema deseado, seguido de enter. Para continuar ingrese la opción "Guardar y salir" (6) y enter.

La consola desplegara las diferentes opciones de personalización. Ingrese mediante su teclado la opción deseada y de ENTER.

Default	Cherry	MatrixEyes
		
Aqua	**Hazard**	
		

C.3. El menú.

Una vez elegido el tema, pasará a un menú donde podrá realizar las siguientes acciones:

1. **Ingresar un grafo:** La función principal de BiBi, pide las adyacencias del grafo y determina si se trata de un grafo bipartito.
2. **Cambiar el tema:** Si no queda convencido con el tema elegido, puede cambiarlo.
3. **Salir:** Bibi termina su ejecución.

```
Cambios guardados
-----------------
PROYECTO FINAL: A partir de una matriz de adyacencia
determina si se trata de un grafo bipartito.

¿Qué desea hacer ahora?
1. Ingresar un grafo.
2. Cambiar el tema.
3. Salir.
Opción:
```

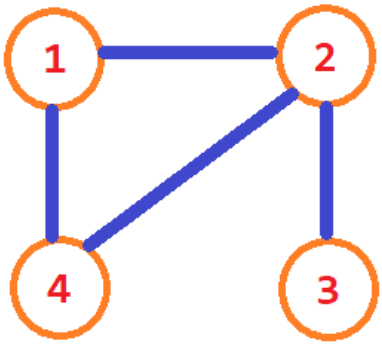
Ingrese el numero de la opción deseada mediante el teclado y presione ENTER.

C.4. Para ingresar un grafo.

Al elegir la opción 1 en el menú. BiBi pedirá el número de nodos del grafo a evaluar. A continuación, se mostrarán ejemplos de grafos y el resultado que BiBi determina.

CASO DE PRUEBA #1:

El grafo tiene las siguientes características:



- 4 nodos (Orden 4).
- 4 aristas.
- Conexo.
- Un ciclo de longitud impar (esto descarta automáticamente que sea bipartito).

Para ingresar las adyacencias, BiBi preguntara por la relación entre dos nodos, en caso de existir una arista entre ambos deberá ingresar (1), de lo contario debe ingresar (0); para avanzar presione ENTER.

```
M A T R I Z  D E  A D Y A C E N C I A

Nodo[1] y Nodo[1],¿Tienen relación?: 0
Nodo[1] y Nodo[2],¿Tienen relación?: 1
Nodo[1] y Nodo[3],¿Tienen relación?: 0
Nodo[1] y Nodo[4],¿Tienen relación?: 1
Nodo[2] y Nodo[2],¿Tienen relación?: 0
Nodo[2] y Nodo[3],¿Tienen relación?: 1
Nodo[2] y Nodo[4],¿Tienen relación?: 1
Nodo[3] y Nodo[3],¿Tienen relación?: 0
Nodo[3] y Nodo[4],¿Tienen relación?: 0
Nodo[4] y Nodo[4],¿Tienen relación?: 0
Press any key to continue . . .
```

- No hay lazos.
- El nodo 1 tiene relación con el nodo 2.
- El nodo 1 no tiene relación con el nodo 3.
- El nodo 1 tiene relación con el nodo 4.
- El nodo 2 tiene relación con el nodo 3.
- El nodo 2 tiene relación con el nodo 4.
- El nodo 3 no tiene relación con el nodo 4.

BiBi, imprime la matriz de adyacencia generada y su veredicto. Este grafo no es bipartito.

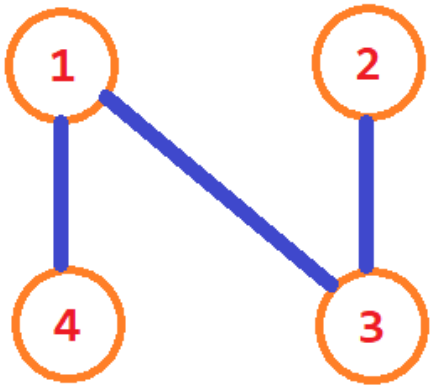
```
Matriz de Adyacencia generada:
        0       1       0       1
        1       0       1       1
        0       1       0       0
        1       1       0       0
El grafo no es bipartito.


Press any key to continue . . .
```

Presione ENTER para continuar. Se enviará de regreso al menú.

CASO DE PRUEBA #2:

El grafo tiene las siguientes características:



- 4 nodos (Orden 4).
- 3 aristas.
- Conexo.

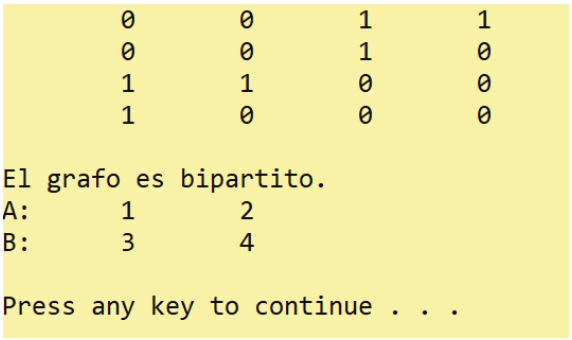
De igual forma, BiBi preguntara por la relación entre dos nodos, en caso de existir una arista entre ambos deberá ingresar (1), de lo contario debe ingresar (0); para avanzar presione ENTER.

```
M A T R I Z  D E  A D Y A C E N C I A

Nodo[1] y Nodo[1],¿Tienen relación?: 0
Nodo[1] y Nodo[2],¿Tienen relación?: 0
Nodo[1] y Nodo[3],¿Tienen relación?: 1
Nodo[1] y Nodo[4],¿Tienen relación?: 1
Nodo[2] y Nodo[2],¿Tienen relación?: 0
Nodo[2] y Nodo[3],¿Tienen relación?: 1
Nodo[2] y Nodo[4],¿Tienen relación?: 0
Nodo[3] y Nodo[3],¿Tienen relación?: 0
Nodo[3] y Nodo[4],¿Tienen relación?: 0
Nodo[4] y Nodo[4],¿Tienen relación?: 0
Press any key to continue . . .
```

- No hay lazos.
- El nodo 1 no tiene relación con el nodo 2.
- El nodo 1 tiene relación con el nodo 3.
- El nodo 1 tiene relación con el nodo 4.
- El nodo 2 tiene relación con el nodo 3.
- El nodo 2 no tiene relación con el nodo 4.
- El nodo 3 no tiene relación con el nodo 4.

BiBi, imprime la matriz de adyacencia generada y su veredicto. Este grafo es bipartito, BiBi imprime los dos grupos de nodos que se forman.



Presione ENTER para continuar. Y se enviará de regreso al menú.

C.5. Para salir

Si ingresa la opción "Salir" del menú, el programa mostrará una pantalla de créditos, y terminará la ejecución de BiBi.

```
Goodbye!
Hecho por:
Ximena Rivera Delgadillo ID:261261
José Luis Sandoval Perez ID: 261731
Diego Emanuel Saucedo Ortega ID:261230
Carlos Daniel Torres Macias ID:244543
Press any key to continue . . .

Process returned 0 (0x0)   execution time : 287.774 s
Press any key to continue.
```

CONCLUSIÓN:

A través de la investigación y proyecto realizado logramos concluir que definitivamente existe una solución para determinar si un grafo es bipartito, o existe una bipartición en dicho grafo, conociendo únicamente su matriz de adyacencia; el proceso lógico de la solución de nuestro problema no fue del todo fácil, sobre todo al tener que pasar nuestra lógica racional a un programa con lenguaje de programación como lo es ANSI C.

Aunque ANSI C puede resultar un lenguaje no tan complejo, al iniciar el desarrollo del programa logro acumular un grado de dificultad considerable incluyendo que la problemática inicial del programa ya representaba un reto. Definitivamente lo visto dentro de nuestro curso de Fundamentos en Estructuras Computacionales fue un parteaguas en la búsqueda de la solución definitiva a nuestro problema. Los grafos bipartitos son fáciles de determinar, solo debemos de ser muy cuidadosos al encontrar que vértices se relacionan entre si para poder determinar los 2 conjuntos de vértices del grafo y determinar su bipartición.

Una de las ventajas clave en el desarrollo del proyecto sin duda fue el contar con un equipo como apoyo ya que el hecho de que el trabajo haya sido en equipos nos dio posibilidad de explorar muchas más soluciones al problema, gracias a que la lógica de programación de cada persona varia, apoyo a una mejora considerable en el trabajo permitiendo innovar la manera en la que desarrollamos nuestra lógica a lo largo de la búsqueda de una solución para un problema en específico.

Logramos implementar de una manera conjunta los conocimientos generados a lo largo del curso con los propios también de nuestra materia de programación, así como trabajar de una manera conjunta para idear un programa aún más interesante y creativo que nos permitiera lograr el objetivo final de una manera eficiente y al mismo tiempo divertida e interesante para cualquiera que desee ejecutar el programa.

BIBLIOGRAFÍA:

1. Academia de Ingeniería México, Saracho Luna, A., & Castaño Meneses, V. M. (2017, noviembre). Teoría de Grafos Una Introducción Histórico-Técnica.

2. https://www.ai.org.mx/sites/default/files/teoria_de_grafos_.pdfCrea un grafo en línea y encuentra caminos más cortos entre vértices o usa otros algoritmos. (s. f.). Graph Online. Recuperado 30 de noviembre de 2021, de https://graphonline.ru/es/

3. Teoría de Grafos. (2012, julio). https://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf

4. UCAM Universidad Católica de Murcia. (2016, 21 abril). Matemática Discreta - Grafo bipartido - Jesús Soto [Vídeo]. YouTube. https://www.youtube.com/watch?v=oxslxaHiQEI&ab_channel=UCAMUniversidadCat%C3%B3licadeMurcia

5. GRAFOS. (s. f.). Departamento de Ciencias de la Computación e Inteligencia Artificial. Recuperado 30 de noviembre de 2021, de https://ccia.ugr.es/%7Ejfv/ed1/tedi/cdrom/docs/grafos.htm

6. López Avellaneda, D. (2019, 25 noviembre). Matriz de adyacencia de un grafo –. Matemáticas IES. https://matematicasies.com/Matriz-de-adyacencia-de-un-grafo