# Reporte Individual de Unidad II
# Identificando Grafos Bipartitos

Torres Colorado Juan Daniel*
*Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

***Resumen*— El presente proyecto resalta la importancia de comprender y visualizar el algoritmo de Grafos Bipartitos, una estructura fundamental en ciencias de la computación y matemáticas. El objetivo principal fue desarrollar una visualización interactiva del algoritmo de Grafos Bipartitos utilizando el lenguaje de programación C++. Se implementaron clases y funciones para la manipulación y representación gráfica de grafos, así como para la ejecución del algoritmo bipartito. La visualización del algoritmo no solo facilita su comprensión, sino que también ofrece una herramienta educativa valiosa para estudiantes y profesionales interesados en la teoría de grafos. Además, este proyecto demuestra la capacidad de implementar algoritmos complejos de manera efectiva y presentarlos de forma accesible y comprensible.**

## I. Introducción

El presente proyecto de nombre "Identificando Grafos Bipartitos" fue asignado por el docente y tuvo lugar a mediados de mayo del presente año y fue desarrollado en el lenguaje de programación C++ [1] utilizando la Programación Orientada a Objetos [2].

Un grafo [3] es un conjunto de nodos (también llamados vértices) que están conectados por medio de aristas, la cuales pueden tener una dirección o no tenerla. Esta estructura de datos es utilizada comúnmente (entre otros más) para representar una red de computadoras, relaciones en redes sociales o incluso para trazar rutas en un sistema de transporte.

Un grafo bipartito [4] es un tipo de grafo en la cual los nodos pueden ser divididos en dos conjuntos independientes *U* y *V*, tal que todas las aristas están conectadas de *U* a *V*. En otras palabras, para todas las aristas, cada nodo *u* pertenece a *U* y *v* a *V* o cada nodo *v* pertenece a *U* y *u* a *V*. Además, se puede decir que no hay aristas que conectan con nodos del mismo conjunto.

El propósito de este proyecto tiene como motivo principal comprender el algoritmo de un grafo bipartito de manera visual, en la cuál se puedan dibujar trazos de nodos y aristas de manera dinámica. Así también, como partir desde un punto especificado por el usuario.

## II. Desarrollo Experimental

El desarrollo experimental del código se basa principalmente de las fuentes atribuidas por el docente, en tales casos, se fundamenta en los principios y conceptos presentados en el libro "Data Structures & Algorithms in C++ [5]", así mismo como el uso de fuentes externas.

En primer lugar, en la investigación se hizo un análisis acerca de como trabaja trabajan los grafos para comprender en profundidad sus estructuras y funcionalidades, como lo es entender la funcionalidad de visitar nodos vecinos del grafo en cuestión. Esto incluyó estudiar diferentes tipos de algoritmos de recorrido y búsqueda (como lo es el Breadth-First Search), así como también técnicas para realizar la bipartición (a partir de dos conjuntos de listas *deque*).

Gracias a los conocimientos previstos en antiguos proyectos experimentales, se obtuvo cierto grado de facilidad para el desarrollo ante el desafío del diseño visual e implementación del algoritmo de grafos bipartitos. Así también, como para la gestión de memoria para el almacenamiento de los nodos, las aristas, el recorrido del algoritmo y la visualización de cada uno de estos elementos. Con ello mismo, se hizo uso del siguiente algoritmo como base para implementarse:

**Algorithm 1** Algorithm to check the Bipartiteness of a Graph

**Require:** $G(V,E)$, S
**Ensure:** Bipartite or Not Bipartite
1: $r$ = NULL;
2: $color.S$ = RED;
3: $r.push(S)$;
4: **while** $r$ *is not empty* **do**
5: $n1 = r.pop()$;
6: **for** *each* $n2$ *in* $n1.Adj()$ **do**
7: **if** $color.n2 == NULL$ **then**
8: $color.n2 = (color1.n2 == RED)\ ?\ BLUE : RED$;
9: $r.push(n2)$;
10: **else**
11: **if** $color.n2 == color.n1$ **then**
12: **return** "Graph is not a Bipartite";
13: **end if**
14: **end if**
15: **end for**
16: **return** "Graph is a Bipartite";
17: **end while**

El desarrollo de este proyecto se hizo necesario la creación de diversas clases (tales como: Node, Graph, Window) que fueran las capaces de lograr gestionar los nodos en los grafos así como de la visualización en pantalla del algoritmo.

## III. Resultados

Los resultados de este proyecto resultan en la visualización del algoritmo de Grafos Bipartitos, en la figura 1 demuestra menú principal del proyecto, las cuales conlleva acciones diponibles como: Crear y Eliminar Aristas, Restablecer colores de los nodos, Eliminar el Grafo y realizar la ejecución del algoritmo de Grafo Bipartito.

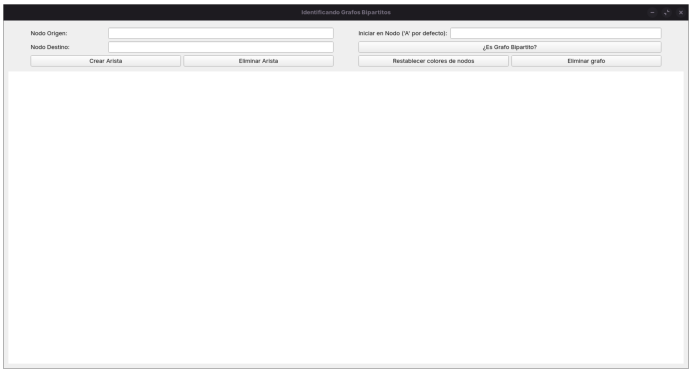



Figura 1: Menú principal

En la figura 2 se representa un ejemplo de cómo se agregan los nodos en el grafo, de manera sencilla como si de un pincel se tratara: con el puntero haciendo solamente click en algún lugar del recuadro blanco (tablero) hará que se agreguen nodos al grafo y se representará en el lugar en el que se hizo dicha acción.

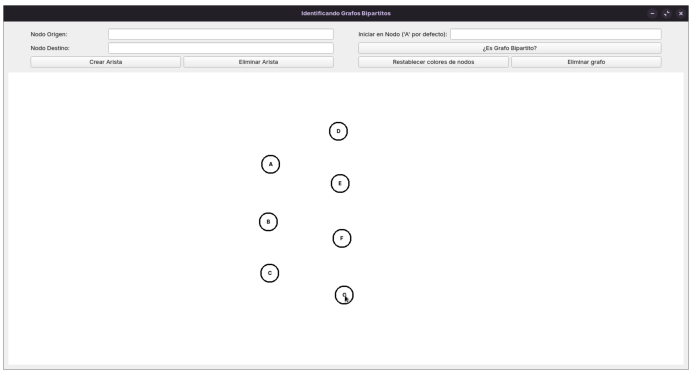



Figura 2: Creación de Grafo

Como se explicó anteriormente, los nodos tienen aristas de manera en que están tanto en el nodo A como en el nodo B, en la figura 3 demuestra esta misma funcionalidad, de manera en que para saber en que punto origen se tiene un punto destino para la creación de una arista, se deberá ingresar el valor de dicho nodo.

Con lo anterior realizado, ahora sí podemos ejecutar nuestro algoritmo de Grafo Bipartito presionando el botón "¿Es Grafo Bipartito?" de manera en que tiene como parámetro opcional recibir desde que nodo se quiere partir, en dado caso que no se ingrese algún valor del nodo inicio, entonces se realizará apartir del primer elemento que se haya agregado en primera instancia (es decir, el nodo 'A') de manera en que se guardará

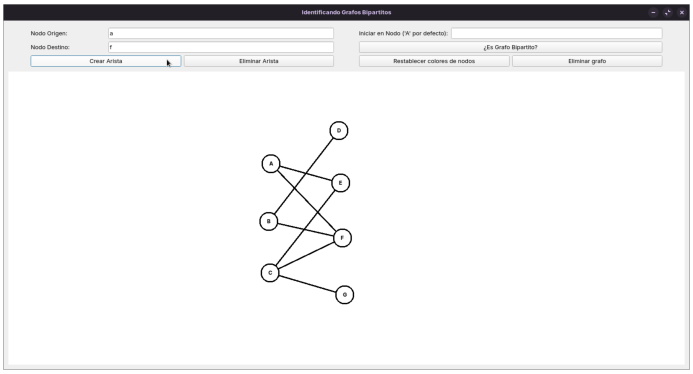



Figura 3: Creación de Aristas

este primero en un conjunto 'A', al visitar sus vecinos estos estarán en guardados en un conjunto 'B' y serán marcados como "visitados" generando que no se imparta un error de bucle y sepamos identificar los nodos que han sido visitados, en la cual (dependiendo del grafo ingresado) mostrará por mensaje en una venta mencionando si el gafo es o no un Grafo Bipartito. La figura 4 demuestra un ejemplo de la secuencia visual de esto mismo.

En el caso en el que el gafo ingresado no sea un grafo bipartito, entonces, se terminará la representación visual y se mostrará un mensaje en pantalla informando acerca de esto mismo, en la figura 5 se demuestra un ejemplo.

El resto de funcionalidades sirven de aprovechamiento para evitar tener que regenerar un grafo en caso de que haya ocurrido un error en la conexión de un nodo a otro, facilitando así la experiencia del uso impartido de este proyecto, en la figura 6 se demuestran los casos de usos de estos mismos.

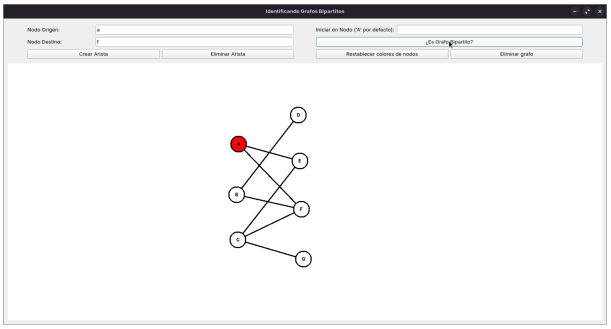
## IV. Conclusión

La conclusión de este proyecto destaca la importancia de comprender y visualizar el algoritmo de Grafos Bipartitos. Los grafos en general son una estructura de datos fundamental en ciencias de la computación y matemáticas, utilizados para modelar una variedad de problemas del mundo real, desde redes sociales hasta rutas de transporte, así como en ciertas áreas de la programación o incluso de redes de computadoras.

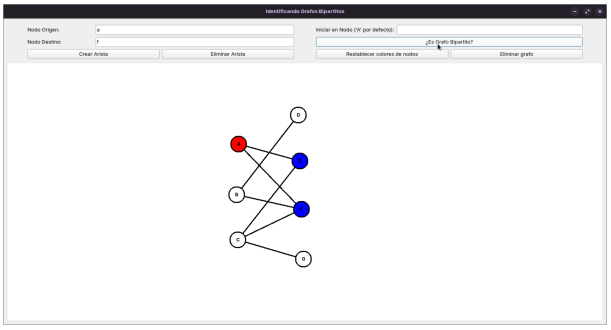
Específicamente, los Grafos Bipartitos tienen aplicaciones en áreas como la teoría de grafos, la optimización, la planificación de horarios, la asignación de recursos y más. Su propiedad de poder dividir los nodos en dos conjuntos independientes los hace especialmente útiles en problemas donde se necesita asignar recursos de manera eficiente sin conflictos.

La visualización del algoritmo de Grafos Bipartitos no solo facilita la comprensión del algoritmo en sí mismo, sino que también proporciona una herramienta útil para educación y análisis de problemas que involucran grafos bipartitos. Además, este proyecto demuestra la capacidad de implementar algoritmos complejos de manera efectiva y presentarlos de una manera accesible y comprensible.

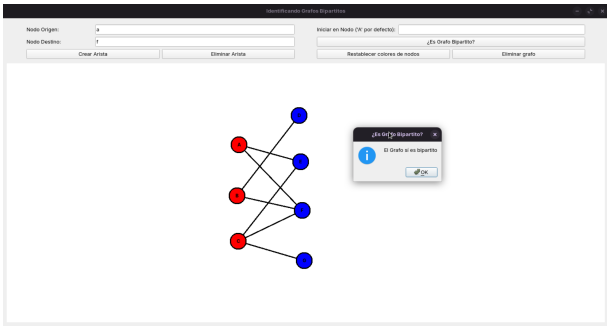
Para finalizar, este proyecto proporcionó conocimiento de manera exitosa al alumno, además de proporcionar resultados



(a) Iniciación en Nodo 'A'



(b) Visita en vecinos



(c) Resultados

Figura 4: Grafo Bipartito

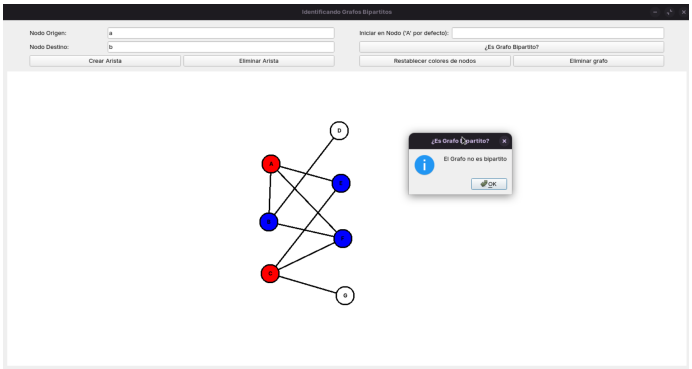
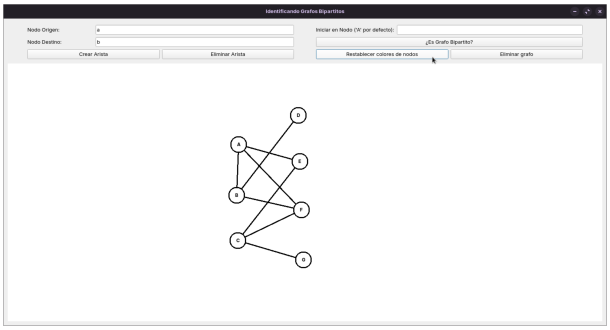
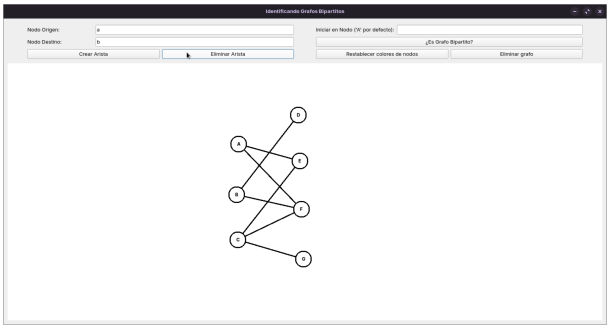


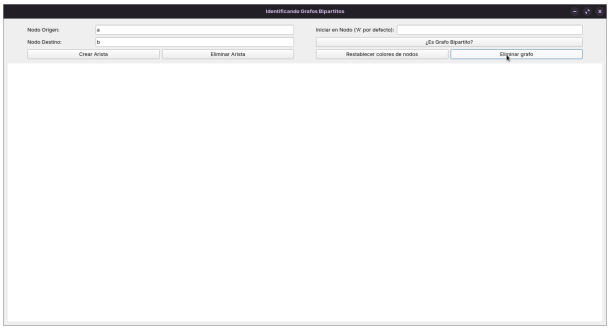
Figura 5: Grafo No Bipartito



(a) Restablecimiento de colores de los nodos



(b) Eliminación de arista



(c) Eliminación de grafo

Figura 6: Grafo Bipartito

satisfactorios para la visualización de la ejecución de este algoritmo.

## Referencias


[1] Nick Barney. *What is C++?* https://www.techtarget.com/searchdatamanagement/definition/C. Consultado el 16-03-2024.

[2] Alexander S. Gillis. *What is Object-Oriented Programming (OOP)?* https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP. Consultado el 16-03-2024.

[3] ThedShanbook. *Graph - Data Structure*. https://www.thedshandbook.com/graphs/. Consultado el 16-03-2024.

[4] *What is Bipartite Graph*. https://www.geeksforgeeks.org/what-is-bipartite-graph/. Consultado el 16-03-2024.

[5] Michael T. Goodrich & Roberto Tamassia & David Mount. *Data Structures & Algorithms in C++*. Second Edition. 2011.