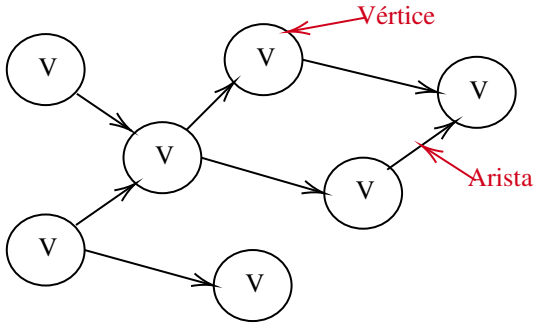
Reporte de Proyecto Grupal 1
Graphs

Moreno Ledesma Ximena Abigail*, Palmero Torres Javier Martín*,
Ruiz Marquez Adrián Alejandro* y Torres Colorado Juan Daniel *
Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

***Resumen*— En el presente reporte se dará a conocer el desarrollo para la creación de un sistema que busca ejemplificar el uso de grafos, implementando las operaciones típicas que se aplican a dicha estructura. La programación de este sistema se llevó a cabo en base al lenguaje de programación C++ y Qt5.**

I. Introducción

Los grafos son una composición interesante de conjuntos de objetos denominados ***nodos***, en los cuales es posible almacenar diferentes tipos de elementos o datos que pueden ser utilizados para procesar y analizar relaciones o procesos.
Este tipo de estructuras se puede ver como un par ordenado de V y A, donde ***V*** es el ***conjunto de vértices*** o nodos del grafo y ***A*** es una ***arista*** que une a un conjunto de pares de vértices, a estos también se les llama arcos o ejes del grafo. Un vértice puede tener 0 o más aristas, pero toda arista debe unir exactamente a 2 vértices.
De esta forma, los grafos representan conjuntos de objetos que no tienen restricción de relación entre ellos.[1][2][3]



El proyecto se centra en la implementación y exploración de estructuras de datos para grafos, por lo cual se hará uso particular del lenguaje de programación **C++** para la implementación y funcionamiento lógico, así como el uso de **Qt5**, un framework de desarrollo de software multiplataforma que permite crear aplicaciones gráficas para múltiples sistemas operativos. Con ayuda de estas herramientas se plantea la elaboración de una aplicación que permite a los usuarios explorar de manera gráfica las operaciones aplicables a la estructura de un grafo. Cabe destacar que la elaboración de este proyecto se fundamenta en el Capítulo 12 ¨Graphs¨ del libro ¨Open Data Structures¨.[4]
Este capítulo es un componente esencial para entender la aplicación de grafos en su forma de Matrices de Adyacencia y Listas de Adyacencia, así como sus técnicas de recorrido, como la búsqueda en Amplitud y la búsqueda en Profundidad.

Dentro de este reporte abordarán los conceptos teóricos presentados en el libro a través de la implementación y análisis de algoritmos relacionados con la Representación y el Recorrido de Grafos.

II. Desarrollo Experimental

C++ ha evolucionado a partir del lenguaje de programación C, y con el tiempo, ha experimentado una mayor evolución y desarrollo desde su definición original. Ha permitido la introducción de características que no formaban parte de C, como constantes simbólicas, sustitución de funciones en línea, tipos de referencia, polimorfismo paramétrico a través de plantillas y excepciones (que se discuten más adelante). Como resultado, C++ ha crecido hasta convertirse en un lenguaje de programación complejo y seguro.[5]
Los sistemas lógicos de grafos en C++ se beneficia de estas mejoras, permitiendo una manipulación eficiente de los datos y una mayor flexibilidad en la implementación de algoritmos. De este modo, hace posible la manipulación de punteros y referencias, así como la creación de estructuras de datos complejas entre los cuales se pueden destacar las listas enlazadas, matrices de adyacencia y listas de adyacencia; de esta forma facilitando la representación y manipulación de grafos.
La implementación de C++ se centró en la creación de una estructura de datos robusta y eficiente para grafos. Se consideraron varias estructuras, como listas enlazadas, matrices de adyacencia y listas de adyacencia, y se eligieron basándose en los requisitos específicos del proyecto y en su eficiencia en diferentes contextos.

Después de explorar los fundamentos de los grafos y su implementación en C++, es importante destacar la importancia de Qt como un framework que facilita el desarrollo de aplicaciones gráficas y lógicas de grafos de manera eficiente y multiplataforma. Qt5 ha evolucionado desde una sencilla biblioteca de clases hasta un amplio framework, el cual ha mejorado la calidad de su entorno de desarrollo, *Qt Creator*, y su compatibilidad con más lenguajes y plataformas gracias a su línea de comandos (CLI). Así, el equipo de Qt ha conferido siempre una gran importancia a abarcar también campos de

aplicación específicos de los sectores de la computación de escritorio, así como de los dispositivos móviles, ofreciendo una amplia gama de bibliotecas y módulos de desarrollo.[6]
Su integración se llevó a cabo para desarrollar una interfaz gráfica de usuario (GUI) que permita a los usuarios interactuar con las estructuras de datos de grafos de manera intuitiva. Además de su capacidad para desarrollar aplicaciones gráficas multiplataforma y por su amplia gama de herramientas y widgets, destacando los elementos de Creación de Ventanas, Entrada de Dato y Visualización de Resultados.

En el contexto de este proyecto, los grafos juegan un papel central.

Existen dos formas estándar de representar un grafo: como una colección de ***listas de adyacencia*** (Partiendo de los nodos de cada vértice, evoluciona una lista que informa los nodos que son adyacentes del inicial) o como una ***matriz de adyacencia*** (matrices cuadradas de tantas filas y columnas como vértices tenga el grafo a representar. En las celdas de la matriz se indica si existe un arco entre los vértices que determinan la celda). Cualquiera de las dos formas se aplica tanto a los grafos dirigidos como a los no dirigidos. [7]

Asimismo, algunas operaciones básicas realizadas en grafos son: addEdge, removeEdge, hasEdge, outEdges, inEdges que son esenciales para su manipulación. Estas operaciones permiten a los desarrolladores interactuar con la estructura de datos del grafo, modificándola según sea necesario para el proyecto.[8]

Además de algoritmos de búsqueda en grafos, como la búsqueda en amplitud (Breadth-First Search, BFS) llamada así porque expande la frontera entre vértices descubiertos y no descubiertos uniformemente a lo ancho de la frontera; y la búsqueda en profundidad (Depth-First Search, DFS) que como su nombre indica, busca *más profundo* en el grafo siempre que sea posible. La búsqueda en profundidad explora los bordes que salen del vértice más recién descubierto v que aún tiene bordes no explorados que salen de él.[9]
Estos algoritmos son fundamentales para explorar y analizar la estructura de los grafos.
Para validar procesos y asegurar la correcta funcionalidad de las Implementaciones en C++ y las Interfaces en Qt5, se llevaron a cabo pruebas de integración, pruebas de usabilidad, y pruebas de rendimiento para verificar la eficiencia de las operaciones realizadas en distintos sistemas.
Las pruebas revelaron que la aplicación funciona como se esperaba, permitiendo a los usuarios interactuar con grafos de manera efectiva y visualizar los resultados de las operaciones realizadas. **Los resultados obtenidos confirman la eficacia de la implementación en C++ y la eficaz integración con Qt5**

III. Resultados

El desarrollo del proyecto resultó en una aplicación gráfica interactiva que permite a los usuarios manipular y explorar estructuras de datos para grafos de manera visual. Esta aplicación, construida con C++ y Qt5, proporciona una plataforma robusta para la creación, visualización y análisis de grafos. A continuación, se detallan los resultados obtenidos en cada una de las funcionalidades implementadas así como si respectiva captura de pantalla.

III-1. ***Creación de Vértices****:* El componente principal del sistema es el área de dibujo, donde los usuarios pueden interactuar para crear grafos. La aplicación permite dibujar vértices al hacer clic dentro del área . Esta funcionalidad es esencial para la construcción de grafos, ya que cada clic representa la adición de un nuevo vértice al grafo. Los vértices se pueden agregar libremente, lo que permitirá a los usuarios diseñar grafos complejos y personalizados.

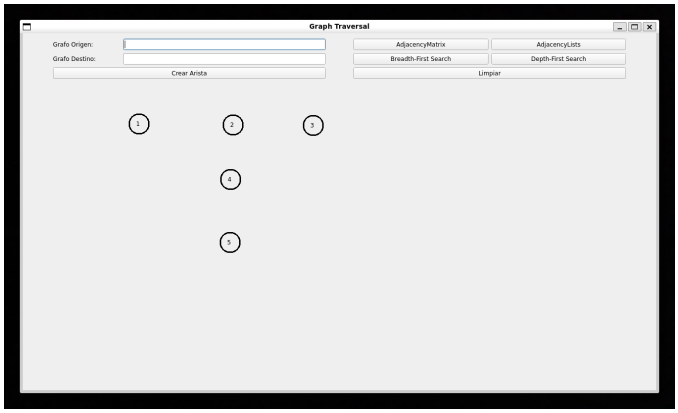


Figura 1: Adición de Vértices

III-2. ***Dibujo de Aristas****:* Para complementar el sistema de dibujo, se implementó una funcionalidad en la cual se le solicita al usuario ingresar los números de los vértice de origen y de destino para dibujar una arista que los una. Esta característica es crucial para definir las conexiones entre los vértices, permitiendo así la representación de relaciones o procesos entre objetos en el grafo.

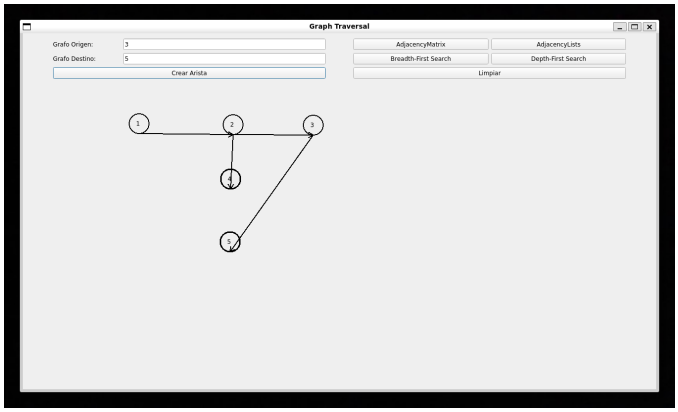


Figura 2: Trazado de Aristas

III-3. ***Generación de Matrices y Listas de Adyacencia****:* La aplicación cuenta con botones para generar tanto una matriz de adyacencia como una lista de adyacencia del grafo actualmente dibujado. Estas representaciones permiten una comprensión más profunda de la estructura del grafo, facilitando el análisis de sus propiedades y la aplicación de

algoritmos específicos.

- **Matriz de Adyacencia**: Representa las conexiones entre los vértices del grafo mediante una matriz cuadrada. Cada entrada en la matriz indica la presencia (1) o ausencia (0) de una arista entre dos vértices.

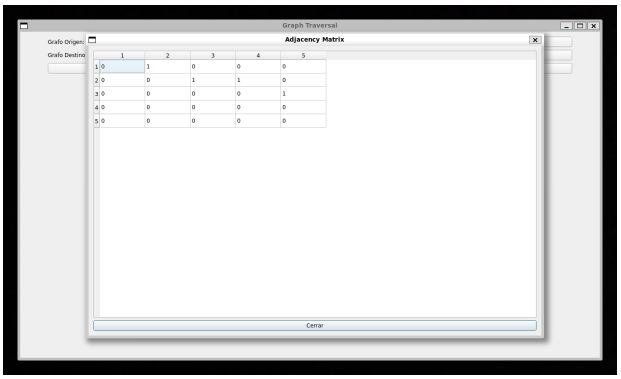


Figura 3: Matriz de Adyacencia

- **Lista de Adyacencia**: Ofrece una representación alternativa donde cada vértice tiene una lista de sus vértices adyacentes. Esta representación es más eficiente en términos de espacio para grafos dispersos.

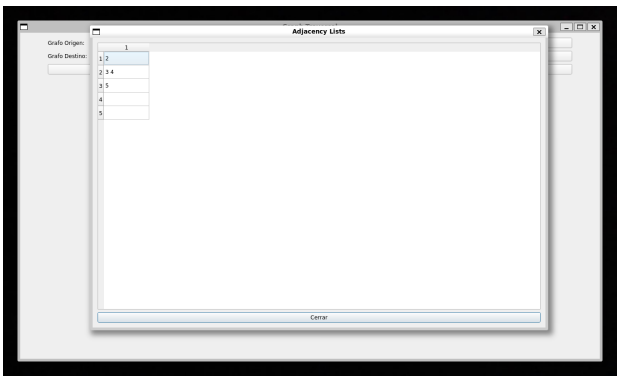


Figura 4: Lista de Adyacencia

*III-4. **Algoritmos de Búsqueda**:* La aplicación incluye botones para realizar búsquedas en profundidad y búsqueda en amplitud en el grafo. Estos algoritmos son fundamentales para explorar y analizar la estructura de los grafos, permitiendo encontrar caminos entre vértices, identificar componentes conectados, y más.

- **Búsqueda en Amplitud (BFS)**: En esta se comienza en un vértice raíz y se comienza a explorar los vértices vecinos a corta distancia antes de pasar a los vértices más lejanos. Es particularmente útil para encontrar el camino más corto entre dos vértices en un grafo no ponderado.
  Para su ejecución se activa el botón, el cual manda a llamar la función de búsqueda en amplitud para que esta realice el recorrido de los vértices, registre su alcance e imprima la información en una ventana nueva ventana.

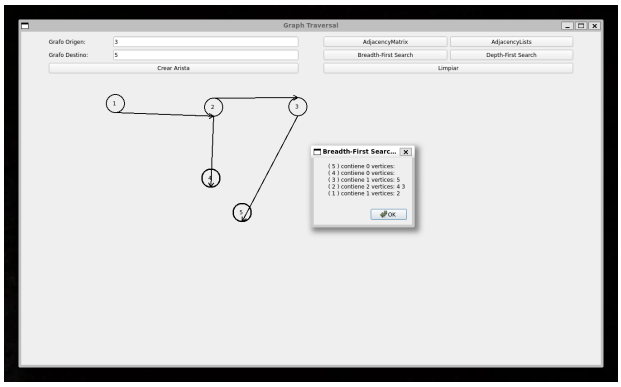


Figura 5: Breadth-First Search

- **Búsqueda en Profundidad (DFS)**: Se comienza en un vértice raíz y se busca explorar tan lejos como sea posible, a lo largo de cada rama antes de retroceder. Es útil para identificar todos los caminos posibles entre dos vértices o para encontrar componentes conectados.
  Su botón manda llamar a la función de búsqueda en profundidad, la cual explorara los vértices, registrara aquellos vértices contenidos en el recorrido e imprime los resultados en una nueva ventana.

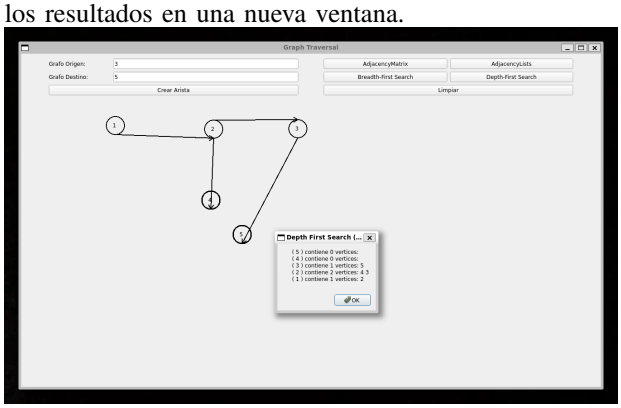


Figura 6: Deep-First Search

En resumen, los resultados obtenidos demuestran la eficacia de la aplicación en la creación, visualización y análisis de grafos. La implementación de estas funcionalidades en C++ y Qt5 ha permitido desarrollar una herramienta interactiva y visualmente intuitiva que facilita la comprensión de las estructuras de datos para grafos y sus aplicaciones. La cual se encuentra publicada en github. [10]

IV. Conclusión

En este proyecto, se abordó la implementación y exploración de estructuras de datos para grafos mediante el lenguaje de programación C++ y el framework Qt5, basándonos en el libro ¨Open Data Structures¨. Se buscó responder a los desafíos de representar y manipular grafos de manera eficiente y visualmente interactiva. A través de la implementación de listas de adyacencia y la integración de interfaces gráficas en Qt5, se logró una comprensión profunda de cómo las estructuras de datos pueden modelar y resolver problemas complejos, facilitando la visualización y análisis de grafos.

Los resultados obtenidos demostraron la eficacia de las estructuras de datos para grafos y la importancia de las inter-

faces gráficas para facilitar la interacción con estas estructuras. La aplicación de algoritmos de búsqueda en grafos, como la búsqueda en amplitud y la búsqueda en profundidad, permitió explorar y analizar la estructura de los grafos de manera intuitiva. La implementación exitosa de estas técnicas proporcionó una plataforma sólida para la exploración y comprensión de los conceptos de grafos y algoritmos.

Este proyecto no solo fortaleció nuestras habilidades en programación y desarrollo de software, sino que también enriqueció nuestro entendimiento de cómo las estructuras de datos y los algoritmos pueden aplicarse en situaciones prácticas. Los conocimientos adquiridos en este proyecto son fundamentales para futuros proyectos que involucren modelado de problemas complejos y desarrollo de aplicaciones interactivas. La integración de teoría y práctica, a través de la implementación en C++ y la interfaz gráfica en Qt5, demostró ser una estrategia efectiva para abordar los desafíos de la programación y el análisis de datos.

REFERENCIAS

[1] Graph Everywhere. *Grafos — Qué son, tipos, orden y herramientas de visualización*. https://www.graphevery where.com/grafos-que-son-tipos-orden-y-herramientas -de-visualizacion/. Consultado el 14-03-2024.

[2] Reinhard Diestel. *Graph Theory, 5th Edition*. Springer, 2017.

[3] Blogspot. *5.1 ELEMENTOS, CARACTERÍSTICAS Y COMPONENTES DE LOS GRAFOS*. https://conjuntos-y - relaciones . blogspot . com / 2017 / 11 / 51 - elementos - caracteristicas-y_20.html. Consultado el 14-03-2024.

[4] Pat Morin. *Open Data Structures*. AU Press, Athabasca University, 2013.

[5] Roberto Tamassia Michael T. Goodrich. *Data Structures and Algorithms in C++*. Pearson Education, 2011.

[6] ionos. *Qt: el framework en C++ para el desarrollo de software multiplataforma*. https://www.ionos.mx/digita lguide/servidores/know-how/qt/. Consultado el 14-03-2024.

[7] Fceia. *Estructura de Datos : Grafos - Fundamentos*. https://www.fceia.unr.edu.ar/estruc/2005/graffund.htm. Consultado el 14-03-2024.

[8] Pat Morin. *Open Data Structures*. AU Press, Athabasca University, 2013.

[9] Ronald L. Rivest Thomas H. Cormen Charles E. Leiserson y Clifford Stein. *Introduction to Algorithms - Fourth Edition*. Massachusetts Institute of Technology, 2022.

[10] Daniel T. *cpp - iti-271154_u1_eq_02_source*. Accedido en Marzo, 2024. 2024. URL: https://github.com/dani elt0312/cpp/tree/main/iti-271154/iti-271154_u1/iti-271154_u1_eq_02/iti-271154_u1_eq_02_source.