UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MATEMATICA PARA LA COMPUTACIÓN 2

CATEDRÁTICO: ING. JOSE ALFREDO GONZALEZ DÍAZ

TUTOR ACADÉMICO: BRAYAN MEJÍA

MANUAL DE USUARIO

JAVIER ANDRÉS VELÁSQUEZ BONILLA

CARNÉ: 202307775

AXEL ABRAHAM ROBLES SOLIZ

CARNÉ: 202307805

RODRIGO ANDRES GUAY MINERA

CARNÉ: 202308208

SECCIÓN: A

GUATEMALA, 29 DE OCTUBRE DEL 2,024

OBJETIVOS DEL SISTEMA

GENERAL

Poner en práctica los conceptos básicos de la teoría de grafos, aplicándolos en el ámbito de la programación.

ESPECÍFICOS

- Entender la importancia y aplicaciones de la teoría de grafos.
- Implementar principios de la teoría de grafos en diferentes lenguajes de programación.
- Ilustrar mediante un entorno gráfico las aplicaciones prácticas de la teoría de grafos.

INTRODUCCIÓN

El propósito de un manual de usuario es ofrecer a los usuarios la información necesaria para utilizar un producto o servicio de manera efectiva. Un manual de calidad no solo detalla el uso de las funcionalidades del producto, sino que también puede incluir consejos prácticos, soluciones a problemas comunes y advertencias de seguridad.

La aplicación que se ha desarrollado es un programa informático con una interfaz gráfica que permite a los usuarios ingresar un grafo y visualizar cómo funciona un algoritmo específico sobre él. Los usuarios pueden añadir los vértices y aristas del grafo, y luego seleccionar el algoritmo que desean aplicar, como el algoritmo de búsqueda en anchura, por ejemplo. Tras introducir el grafo y elegir el algoritmo, la aplicación presenta tanto el grafo original como el resultado obtenido después de aplicar el algoritmo, facilitando así una comprensión visual de su funcionamiento en ese contexto específico.

REQUISITOS DEL SISTEMA

Python: Es necesario tener Python instalado en el sistema. El código está diseñado para funcionar con Python 3.x, por lo que se recomienda contar con una versión compatible.

Bibliotecas de Python: Se requieren las siguientes bibliotecas y sus respectivas dependencias:

- **tkinter:** Utilizada para la creación de la interfaz gráfica de usuario.
- **networkx:** Permite trabajar con grafos en Python.
- **matplotlib:** Se encarga de la visualización de los grafos dentro de la interfaz gráfica.
- **matplotlib.backends.backend_tkagg:** Facilita la integración de matplotlib con tkinter, permitiendo mostrar gráficos en la interfaz.

Estas bibliotecas pueden instalarse de manera sencilla usando un administrador de paquetes como pip. Por ejemplo: pip install tkinter, pip install networkx, y pip install matplotlib.

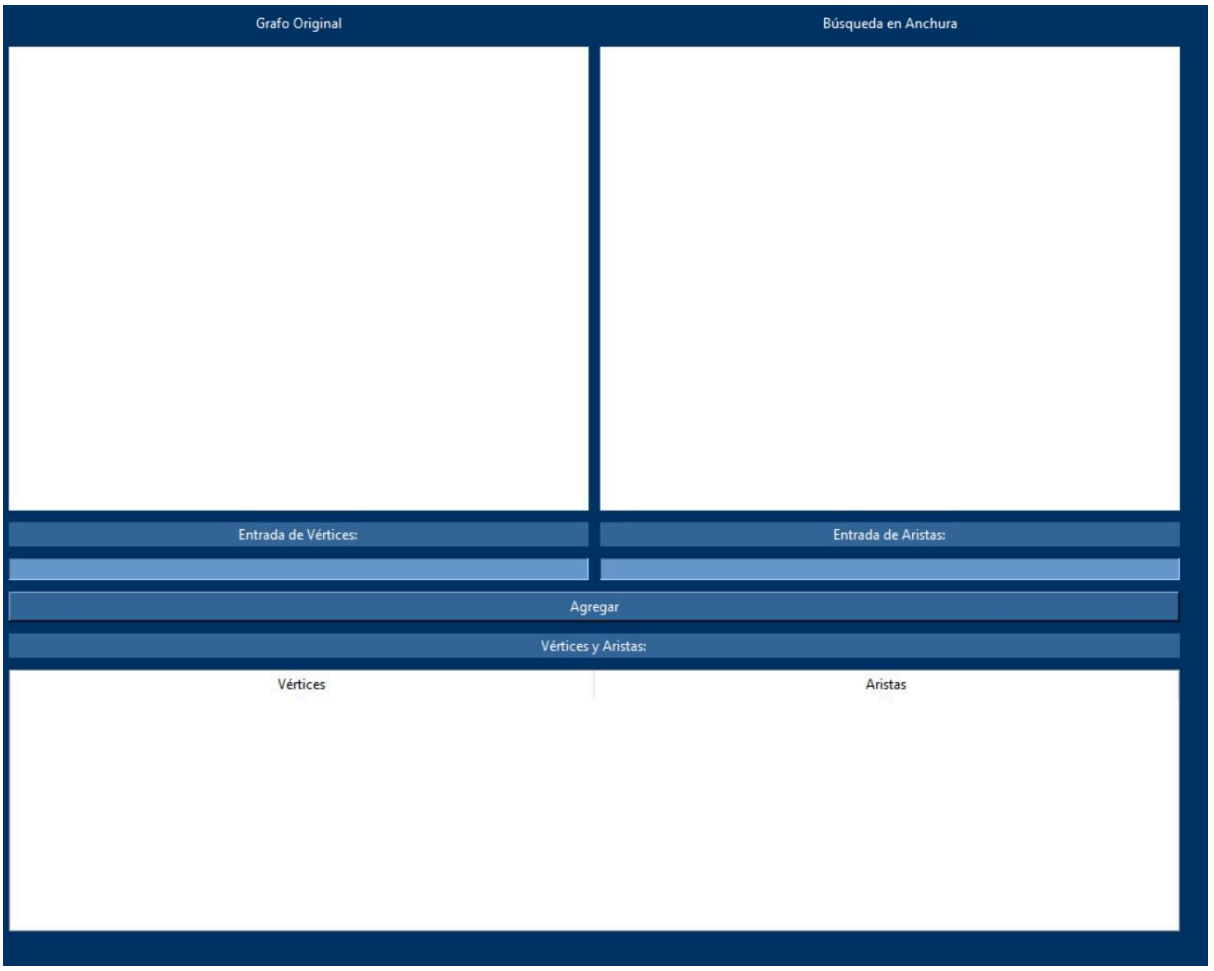
Entorno de Ejecución: Es fundamental contar con un entorno de ejecución que sea compatible con las bibliotecas y el código proporcionado. Se sugiere el uso de un entorno de desarrollo como Anaconda, que ofrece una amplia variedad de bibliotecas y herramientas para el desarrollo en Python.

INFORMACIÓN DEL SISTEMA

- **Interfaz Gráfica de Usuario (GUI):** Al inicio, el programa presenta una interfaz gráfica que permite al usuario interactuar de manera sencilla. Esta interfaz cuenta con áreas específicas para ingresar los vértices y aristas del grafo, así como controles para elegir el algoritmo deseado y otras opciones adicionales.
- **Ingreso del Grafo:** El usuario tiene la capacidad de introducir los vértices y aristas del grafo a través de los campos de entrada disponibles en la interfaz. Esto se puede realizar manualmente, escribiendo los nombres de los vértices y las conexiones entre ellos en un formato específico, como "A--B" para representar una arista entre los vértices A y B.
- **Visualización del Grafo Original:** Una vez que se ingresan los datos del grafo, el programa genera y muestra una representación visual del grafo original en la interfaz.
- **Selección del Algoritmo:** El usuario elige el algoritmo que desea aplicar al grafo que ha ingresado. Por ejemplo, puede optar por el algoritmo de búsqueda en anchura para explorar el grafo desde un vértice inicial y determinar la distancia más corta a todos los demás vértices.
- **Aplicación del Algoritmo:** Tras seleccionar el algoritmo, el programa lo ejecuta sobre el grafo ingresado. Esto implica trabajar con la estructura de datos del grafo para calcular el resultado deseado.
- **Visualización del Grafo Resultante:** Luego de aplicar el algoritmo, el programa presenta una representación visual del grafo resultante en la interfaz. Esta visualización puede incluir el resaltado de ciertos vértices o aristas, según el resultado del algoritmo utilizado, como los vértices visitados en el caso de la búsqueda en anchura.

FLUJO DE LAS FUNCIONALIDADES DEL SISTEMA

1. **Inicio:** El proceso del algoritmo comienza con la selección de un vértice inicial desde el cual se llevará a cabo la exploración del grafo. Este vértice puede ser especificado por el usuario o elegido automáticamente, dependiendo de las necesidades del problema.



2. **Cola de Vértices:** Se establece una cola de vértices para almacenar aquellos que se visitarán en el orden en que son descubiertos. El vértice inicial se coloca en esta cola.
3. **Exploración del Grafo:** Se inicia un bucle que se ejecutará mientras la cola no esté vacía. En cada iteración, se extrae un vértice de la cola y se visitan todos los vértices adyacentes que aún no han sido explorados.

4. **Marcar como Visitado:** Cada vértice que se visita se marca para evitar que se vuelva a visitar, garantizando así que el algoritmo no se quede atrapado en bucles infinitos.
5. **Agregar a la Cola:** Los vértices adyacentes que se visitan se añaden a la cola para ser explorados más adelante. Es fundamental que los vértices se añadan a la cola en el orden en que se descubren, asegurando que el algoritmo opere "en anchura" y explore todos los vértices a la misma profundidad antes de avanzar a los de la siguiente profundidad.
6. **Finalización:** El algoritmo continúa su exploración hasta que la cola queda vacía, indicando que todos los vértices alcanzables desde el vértice inicial han sido visitados.
7. **Resultado:** Al finalizar, el algoritmo genera un árbol de búsqueda en anchura que ilustra la estructura de alcanzabilidad desde el vértice inicial. Este árbol tiene la característica de que la distancia entre el vértice inicial y cualquier otro vértice en él es la mínima posible.