UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
MATEMÁTICA PARA COMPUTACIÓN 2
CATEDRÁTICO: ING. JOSÉ ALFEDO DÍAZ GONZÁLEZ
TUTOR ACADÉMICO: ROBERTO GÓMEZ

MANUAL TÉCNICO

MARCOS GEOVANNI BARRIOS 202300396
JOSUE DAVID FIGUEROA ACOSTA 202307378
VALERY PAMELA ALARCÓN RAMOS 202300794
THANYA LISSET MAZARIEGOS MORALES 202307691
SECCIÓN: A

GUATEMALA, 28 DE ABRIL DEL 2,024

ÍNDICE

INTRODUCCIÓN

El presente manual técnico acompaña al desarrollo del proyecto de implementación de algoritmos de búsqueda en grafos con interfaz gráfica, el cual proporciona una herramienta visual y funcional para comprender y aplicar los conceptos fundamentales de la teoría de grafos en el contexto de las Ciencias de Computación.

En este documento se detallan los aspectos técnicos y funcionales del programa desarrollado, incluyendo la arquitectura del software, las herramientas utilizadas, los algoritmos implementados y las técnicas de visualización empleadas. Adicional, se brindan instrucciones sobre cómo utilizar la interfaz gráfica para ingresar grafos, aplicar algoritmos de búsqueda y visualizar los resultados obtenidos, así como sugerencias para su extensión y personalización.

OBJETIVOS

1. GENERAL

1.1. Brindar una referencia completa y detallada para desarrolladores interesados en comprender y utilizar el programa de implementación de algoritmos de búsqueda en grafos con interfaz gráfica.

2. ESPECÍFICOS

2.1. Describir la arquitectura del software al detallar la estructura del programa, incluyendo sus componentes principales, módulos y relaciones.

2.2. Explicar las tecnologías utilizadas, las herramientas, las bibliotecas y lenguajes de programación empleados en el desarrollo del programa, así como las razones detrás de su elección.

ALCANCES DEL SISTEMA

Proporcionar una guía detallada y accesible que permita a los desarrolladores entender completamente el funcionamiento del programa desarrollado, así como su implementación y uso práctico. Además, busca servir como referencia exhaustiva para los desarrolladores interesados en realizar modificaciones, mejoras o extensiones en el software.

El documento busca garantizar que los lectores puedan aprovechar al máximo las capacidades del programa, comprendiendo su estructura interna, su funcionalidad y su potencial para abordar problemas en el ámbito de la teoría de grafos.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**
  - Computadora Personal: se requiere una computadora que cumpla con los requisitos mínimos para ejecutar Python y las bibliotecas utilizadas en el programa (NetworkX y Tkinter).
  - Sistema Operativo Compatible: sistemas como Windows, macOS o Linux son compatibles con el lenguaje de programación Python.
  - Memoria RAM adecuada: tener al menos 4 GB de RAM es recomendable para un desarrollo fluido y sin problemas.
  - Procesador Moderno: con múltiples núcleos y una velocidad de reloj decente será beneficioso para la ejecución del código.
  - Espacio de Almacenamiento Suficiente: tener el espacio suficiente en el disco es necesario para el desarrollo del programa se recomienda para poder almacenar el código fuente, las bibliotecas y todos los archivos relacionados con el proyecto.
  - Conexión a Internet: para poder descargar o instalar las bibliotecas necesarias y adicionales, buscar la información relacionada con el desarrollo o el acceso a recursos en línea.
- **REQUISITOS DE SOFTWARE**
  - Python: se necesita tener el lenguaje de programación instalado en el sistema. Idealmente se recomienda utilizar una versión de Python que sea compatible con las bibliotecas utilizadas en el programa. En este caso, además, se recomienda tener Python actualizado sus últimas y más recientes versiones estables para aprovechar las últimas características y correcciones de errores.
  - Bibliotecas de Python: se necesitará instalar las bibliotecas en el entorno de programación, utilizando un administrador de paquetes.

- Entorno de desarrollo Integrado (IDE): para escribir, depurar, y ejecutar el código de Python de manera eficiente.
- Herramientas de Control de Versiones: para un desarrollo colaborativo y un control de versiones como Git. Que permite que el programador realice un seguimiento de los cambios en el código fuente.

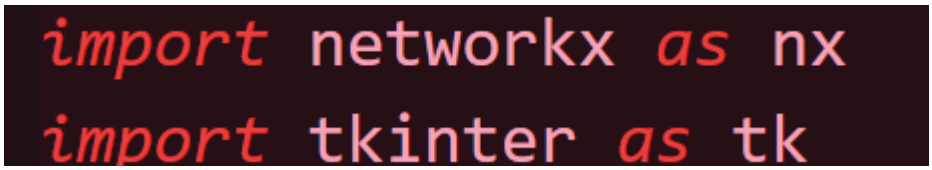
DESCRIPCIÓN DE LA SOLUCIÓN

- Se comprendió el problema presentado, identificando los temas y teoría necesaria para el desarrollo del programa. Se analizaron las funcionalidades solicitadas como lo son la capacidad de ingresar nodos y aristas, aplicar el algoritmo de búsqueda de preferencia y la visualización de los grafos resultantes de manera gráfica. Se evaluaron las tecnologías y herramientas disponibles para implementar las funciones descritas, se utilizó el lenguaje de programación Python debido a su versatilidad y amplio soporte en el campo de la ciencia de datos y la programación. Además se seleccionaron las bibliotecas NetworkX y Tkinter para trabajar con grafos y crear una interfaz gráfica. Al desarrollar un diseño inicial de la arquitectura del programa, se definió cómo se estructurían los diferentes componentes y cómo interactuarían entre sí. Esto incluye la creación de funciones para ingresar nodos y aristas, aplicar los algoritmos de búsqueda y la actualización de la interfaz gráfica. Se implementó el programa de manera iterativa, desde las funciones más básicas hasta las adicionales, siempre realizando numerosas pruebas para asegurar el correcto funcionamiento y lógica.

LÓGICA DEL PROGRAMA

❖ **NOMBRE DE LA CLASE**

Captura de las librerías usadas



➢ **Librerías**

NetworkX: se utiliza para la creación, manipulación y análisis de estructuras de grafos y redes complejas. Se utiliza para representar y trabajar con grafos en el programa.

Tkinter: es la biblioteca estándar de Python para crear interfaces gráficas de usuario GUI. Se utiliza para crear la interfaz gráfica del programa y manejar eventos de usuario.

➢ **Variables Globales**

Captura de sus variables globales

network: representa el grafo utilizado en el programa.

window: representa la ventana principal de la interfaz gráfica.

node_input: representa un campo de entrada de texto para ingresar nodos.

connection_input1/connection_input2: representan campos de entrada de texto para ingresar conexiones entre nodos.

graph_figure: representa la figura que contiene los gráficos de NetworkX.

original_axis/search_axis: representan los ejes de los grafos de NetworkX.

➢ **Función Main**

El programa comienza su ejecución directamente con el llamado a "window.mainloop()" que inicia el bucle principal de la interfaz gráfica.

Captura del código de su función main

```
80    window.mainloop()
```

➢ **Métodos y Funciones utilizadas**

A continuación se dará una explicación general de lo que hace cada función:

- update_graph(dfs_paths): actualiza los grafos de NetworkX con los datos proporcionados. Si se proporciona una lista de caminos de búsqueda en amplitud, dibuja el grafo resultante de la búsqueda en el eje derecho. De lo contrario, dibuja el grafo original en el lado izquierdo.

```
75    update_graph(dfs_paths)
```

- ancho(): se activa cuando se presiona un botón "Búsqueda a lo ancho" en la interfaz gráfica. Calcula los caminos de búsqueda en amplitud desde el nodo de origen proporcionado por el usuario y llama a "update_graph" para actualizar los gráficos.

```
65    def ancho():
```

- execute_dfs(): se activa al presionar el botón "Búsqueda en profundidad" en la interfaz gráfica. Calcula los caminos de búsqueda en profundidad desde

el nodo de origen proporcionado por el usuario y llama a “update_graph()” para actualizar los gráficos.