UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MATEMATICA PARA COMPUTACION 2

CATEDRÁTICO: ING. JOSE ALFREDO GONZALEZ DÍAZ

TUTOR ACADÉMICO: ROBERTO GOMEZ

MANUAL DE USUARIO
PROYECTO

ANGEL JOSE CARDENAS POCON

CARNÉ: 202304063

SECCIÓN: A

GUATEMALA, 29 DE ABRIL DEL 2,024

ÍNDICE

OBJETIVOS DEL SISTEMA

GENERAL

Aplicar los conceptos generales sobre la teoría de grafos enfocados a programación.

ESPECÍFICOS

- Comprender la utilidad de la teoría de grafos.
- Aplicar la teoría de grafos empleando distintos lenguajes de programación.
- Demostrar por medio de un entorno gráfico las aplicaciones de la teoría de grafos.

INTRODUCCIÓN

El fin de un manual de usuario es proporcionar a los usuarios la información necesaria para utilizar eficazmente un producto o servicio. Un buen manual de usuario no solo describe cómo utilizar las funciones del producto, sino que también puede incluir consejos útiles, solución de problemas comunes y precauciones de seguridad

La aplicación desarrollada es un programa informático con una interfaz gráfica que permite ingresar un grafo y visualizar el funcionamiento de un algoritmo específico sobre ese grafo. El usuario puede ingresar los vértices y aristas del grafo y luego seleccionar el algoritmo que desea aplicar, como el algoritmo de búsqueda en anchura, por ejemplo. Una vez que se ingresa el grafo y se elige el algoritmo, la aplicación muestra el grafo original ingresado y el grafo resultante después de aplicar el algoritmo, lo que permite al usuario comprender visualmente cómo funciona el algoritmo en ese contexto particular.

INFORMACIÓN DEL SISTEMA

1. Interfaz Gráfica de Usuario (GUI): El programa comienza mostrando una interfaz gráfica donde el usuario puede interactuar fácilmente con él. Esta interfaz puede incluir áreas designadas para ingresar vértices y aristas del grafo, así como controles para seleccionar el algoritmo deseado y otras opciones adicionales.
2. Ingreso de Grafo: El usuario ingresa los vértices y las aristas del grafo utilizando los campos de entrada proporcionados en la interfaz. Esto puede hacerse manualmente, escribiendo los nombres de los vértices y las conexiones entre ellos en un formato específico (por ejemplo, "A--B" para una arista entre los vértices A y B).
3. Visualización del Grafo Original: Después de ingresar los datos del grafo, el programa muestra una representación visual del grafo original en la interfaz.
4. Selección del Algoritmo: El usuario selecciona el algoritmo que desea aplicar al grafo ingresado. Por ejemplo, puede elegir el algoritmo de búsqueda en anchura para explorar el grafo desde un vértice inicial y encontrar la distancia más corta a todos los demás vértices.
5. Aplicación del Algoritmo: Una vez seleccionado el algoritmo, el programa aplica el algoritmo al grafo ingresado. Esto implica ejecutar el algoritmo sobre la estructura de datos del grafo para calcular el resultado deseado.
6. Visualización del Grafo Resultante: Después de aplicar el algoritmo, el programa muestra una representación visual del grafo resultante en la interfaz. Esto puede incluir resaltar ciertos vértices o aristas según el resultado del algoritmo aplicado, como los vértices visitados en el caso de la búsqueda en anchura.

REQUISITOS DEL SISTEMA

1. Python: El sistema debe tener Python instalado. El código está escrito en Python 3.x, por lo que se recomienda tener instalada una versión compatible de Python.
2. Bibliotecas Python: Se requieren las siguientes bibliotecas Python y sus dependencias:
    - tkinter: Para crear la interfaz gráfica de usuario.
    - networkx: Para trabajar con grafos en Python.
    - matplotlib: Para la visualización de los grafos en la interfaz gráfica.
    - matplotlib.backends.backend_tkagg: Para integrar matplotlib con tkinter y mostrar los gráficos en la interfaz.

Estas bibliotecas se pueden instalar fácilmente utilizando un administrador de paquetes como pip. Osea: pip install tkinter, pip install networkx y pip install matplotlib

3. Entorno de Ejecución: El sistema debe tener un entorno de ejecución compatible con las bibliotecas y el código proporcionado. Se recomienda un entorno de desarrollo como Anaconda, que proporciona una amplia gama de bibliotecas y herramientas para el desarrollo en Python.

FLUJO DE LAS FUNCIONALIDADES DEL SISTEMA

1. Inicio: El algoritmo comienza seleccionando un vértice inicial desde el cual comenzará la exploración del grafo. Este vértice puede ser proporcionado por el usuario o puede ser elegido automáticamente según las necesidades del problema.
2. Cola de Vértices: Se inicializa una cola de vértices, donde se almacenarán los vértices que se visitarán en el orden en que se descubren. Se coloca el vértice inicial en esta cola.
3. Exploración del Grafo: Se ejecuta un bucle mientras la cola no esté vacía. En cada iteración del bucle, se extrae un vértice de la cola y se visitan todos los vértices adyacentes a este vértice que no hayan sido visitados.
4. Marcar como Visitado: Cada vértice visitado se marca como visitado para evitar visitarlos nuevamente y para garantizar que el algoritmo no se quede atrapado en bucles infinitos.
5. Agregar a la Cola: Los vértices adyacentes que se visitan se agregan a la cola de vértices para su posterior exploración. Es importante tener en cuenta que los vértices se agregan a la cola en el orden en que se descubren, lo que garantiza que el algoritmo se comporte de manera "en anchura" y explore todos los vértices a la misma profundidad antes de pasar a los vértices de la siguiente profundidad.
6. Finalización: El algoritmo continúa explorando el grafo hasta que la cola esté vacía, lo que indica que se han visitado todos los vértices alcanzables desde el vértice inicial.
7. Resultado: Como resultado, el algoritmo produce un árbol de búsqueda en anchura que representa la estructura de alcanzabilidad desde el vértice inicial. Este árbol tiene la propiedad de que la distancia entre el vértice inicial y cualquier otro vértice del árbol es la mínima posible.