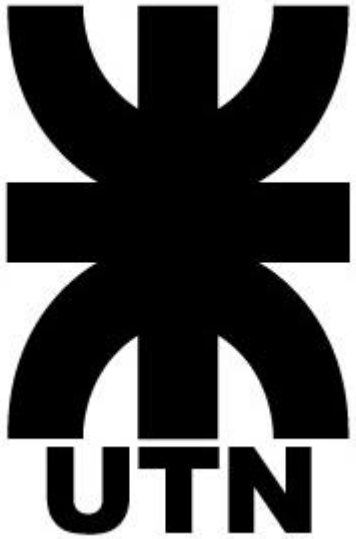

Miembros:
Alan Suarez
Barrios Alejandra
Dylan Grosseti

Informe de Trabajo Práctico: Implementación de Algoritmos de Grafos

Introducción

Este trabajo práctico tiene como objetivo desarrollar un algoritmo que resuelva actividades relacionadas con conjuntos, matrices o grafos. En particular, se ha implementado un grafo

utilizando HTML y JavaScript, donde se pueden añadir aristas entre nodos y realizar búsquedas de caminos mínimos, así como generar tablas para los algoritmos de Prim y Kruskal, y realizar una búsqueda por anchura.

Descripción del Código

El código se compone de una interfaz HTML que permite al usuario interactuar con un grafo representado visualmente. Los nodos están organizados en una estructura en forma de pirámide, y se pueden añadir aristas entre ellos especificando el nodo de inicio, el nodo final y el peso de la arista.

Estructura del Grafo

- **Nodos**: Se definen cinco nodos etiquetados como A, B, C, D y E.
- **Aristas**: Se pueden crear aristas entre nodos, que tienen un peso asociado, lo que permite calcular caminos mínimos y realizar análisis de conectividad.

Algoritmos Implementados

1. **Búsqueda por Camino Mínimo (Dijkstra)**:
    - **Descripción**: Este algoritmo busca el camino más corto desde un nodo de inicio a todos los demás nodos en un grafo ponderado. Utiliza una cola de prioridad para explorar los nodos.
    - **Funcionamiento**:
        - Se inicializan las distancias de todos los nodos a infinito, excepto el nodo de inicio, que se establece en cero.
        - Se utiliza una cola de prioridad para gestionar los nodos a explorar, asegurando que siempre se explore el nodo con la menor distancia acumulada.
        - Se actualizan las distancias de los nodos vecinos y se rastrean los nodos visitados para reconstruir el camino mínimo al final.
2. **Algoritmo de Prim**:
    - **Descripción**: Este algoritmo encuentra el árbol de expansión mínima en un grafo conectado y ponderado.
    - **Funcionamiento**:
        - Se inicia desde un nodo y se utiliza un conjunto para rastrear los nodos visitados.
        - Se añaden aristas de costo mínimo a un resultado hasta que todos los nodos estén conectados.
        - El resultado se presenta en una tabla que muestra las conexiones y sus pesos.
3. **Algoritmo de Kruskal**:
    - **Descripción**: Este algoritmo también busca el árbol de expansión mínima, pero funciona de manera diferente al de Prim, seleccionando las aristas de menor peso.
    - **Funcionamiento**:
        - Se ordenan todas las aristas en función de su peso.

        - Se utilizan estructuras de conjuntos disjuntos para evitar ciclos al añadir aristas al árbol.
        - Al igual que Prim, el resultado se muestra en una tabla con las conexiones y sus pesos.
4. **Búsqueda por Anchura (BFS)**:
    - **Descripción**: Este algoritmo explora todos los nodos a un nivel antes de pasar al siguiente, garantizando que todos los nodos alcanzables se visiten.
    - **Funcionamiento**:
        - Se inicia desde un nodo y se utilizan una cola y un conjunto para rastrear los nodos visitados.
        - A medida que se exploran los nodos, se añaden a un resultado que se muestra al usuario.

Conclusiones

El código implementado proporciona una base sólida para explorar y visualizar la teoría de grafos utilizando algoritmos básicos de búsqueda y análisis. Los algoritmos de Dijkstra, Prim y Kruskal permiten al usuario comprender mejor cómo funcionan las estructuras de grafos y cómo se pueden utilizar para resolver problemas prácticos.

Esta implementación no solo mejora el entendimiento teórico de los grafos, sino que también ofrece una interfaz visual que facilita la comprensión de conceptos como caminos mínimos y árboles de expansión mínima. El siguiente paso podría incluir la optimización del rendimiento y la adición de características más avanzadas, como la visualización de los procesos de búsqueda en tiempo real.