ALGORITMOS Y ESTRUCTURAS DE DATOS

Práctica de Laboratorio | Grafos

Integrantes:

Enrique Miguel Solís Sandoval

Joshua Salvador Vílchez Gutiérrez

Camilo Darío Galeano Vargas

Elián Gabriel Olivares Traña

Docente:

Silvia Gigdalia Ticay López

Proceso realizado para solucionar la práctica de grafos:

Se comenzó desarrollando la estructura base del grafo mediante el constructor __init__. En esta etapa se optó por utilizar un diccionario para representar el grafo, donde cada clave corresponde a un vértice y su valor asociado es un conjunto de vecinos (los vértices conectados a él). Esta decisión facilitó el acceso rápido a la información de las conexiones y evitó duplicados.

Posteriormente, se implementaron métodos para agregar vértices y aristas. En el caso de las aristas, se tuvo especial cuidado en contemplar si el grafo debía ser dirigido o no. Si no lo era, se agregó la arista en ambos sentidos, asegurando que la relación entre los vértices fuera bidireccional.

A continuación, se programaron funciones para obtener la lista de vecinos de un vértice dado y para verificar si existe una arista entre dos vértices específicos. Estas funciones resultaron útiles para probar las conexiones agregadas y confirmar que el grafo se estaba construyendo de forma correcta.

Con la estructura básica del grafo ya establecida, se procedió a implementar los algoritmos de recorrido: BFS (Breadth-First Search) y DFS (Depth-First Search). Para el recorrido en anchura (BFS), se utilizó una cola (deque) que permite manejar los elementos de forma eficiente. En cambio, para el recorrido en profundidad (DFS), se aplicó una estrategia recursiva. Al principio fue un poco complicado evitar visitar un mismo vértice más de una vez, pero se resolvió mediante el uso de un conjunto (set) que almacenaba los vértices visitados durante el recorrido.

Después de tener listos los algoritmos de recorrido, se agregaron métodos adicionales: es_conexo y encontrar_camino. El primero permite verificar si el grafo es conexo, es decir, si se puede llegar desde cualquier vértice a cualquier otro. Para ello, se empleó el recorrido BFS desde un vértice arbitrario y luego se comprobó si se habían visitado todos los vértices. El segundo método, encontrar_camino, también utiliza BFS para buscar un camino entre dos nodos y reconstruir la secuencia de vértices que conectan el origen con el destino.

Finalmente, se realizaron pruebas con distintos ejemplos para asegurarse de que todos los métodos funcionaran correctamente. Para visualizar los resultados y depurar más fácilmente, se incluyeron varios print que mostraban el estado interno del grafo y los resultados de los recorridos y verificaciones. Esto permitió confirmar que la implementación era sólida y que los recorridos se comportaban como se esperaba.