**Informe de Desarrollo de la Práctica: Grafos y Navegación GPS**

**Nombres:** - José María Moncada Maya

-Oscar Iván Alvarado Jirón

-Oscar Enrique Arnuero Ramos

**Tema de investigación:** Aplicación de grafos en programación para navegación GPS

**Actividad:** Resolución modular de problemas con estructuras de datos no lineales

**Objetivo de la práctica**

Desarrollar un conjunto de programas en Python utilizando la estructura de **grafos** para modelar un sistema de navegación GPS. Cada ejercicio fue planteado para practicar diferentes operaciones, recorridos y análisis sobre grafos.

**Estructura de la solución**

La solución se dividió en **tres módulos** o ejercicios, cada uno en su propio archivo:

1. **grafo.py** – Define la clase Grafo con operaciones básicas.
2. **recorridos.py** – Implementa los recorridos BFS y DFS.
3. **analisis.py** – Implementa conectividad y búsqueda de caminos.
4. **main.py** – Archivo principal de pruebas.

**Desarrollo del Ejercicio 1: Construcción del grafo**

* Se creó la clase Grafo con soporte para grafos dirigidos o no dirigidos.
* Se utilizaron **diccionarios** como estructura para representar listas de adyacencia.
* Se implementaron métodos para agregar vértices, agregar aristas, obtener vecinos y verificar existencia de aristas.
* En las pruebas se simularon ubicaciones como nodos (‘A’, ‘B’, ‘C’, etc.) y rutas como aristas.

**Desarrollo del Ejercicio 2: Recorridos (BFS y DFS)**

* Se desarrolló el recorrido **BFS** utilizando collections.deque como estructura de cola eficiente.
* Para **DFS**, se implementó una función recursiva (dfs\_rec) y se diseñó una función auxiliar llamada insertar\_final(lista, elemento) para insertar elementos al final de listas.
* Se realizaron pruebas sobre el grafo no dirigido para verificar el orden de recorrido desde el vértice ‘A’, incluyendo un caso con un vértice desconectado (‘F’).

**Desarrollo del Ejercicio 3: Conectividad y rutas**

* Se implementó el método es\_conexo() usando DFS para verificar si todos los vértices pueden alcanzarse desde uno cualquiera.
* Se creó encontrar\_camino(inicio, fin) adaptando BFS para almacenar el padre de cada vértice y así reconstruir la ruta una vez alcanzado el destino.
* Se manejaron casos especiales como grafos vacíos, vértices inexistentes y caminos no posibles.

**Resultados obtenidos**

* Se validaron los recorridos BFS y DFS.
* Se comprobó la conectividad del grafo.
* Se encontró la ruta más corta entre dos nodos.