UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

ESTRUCTURAS DE DATOS

PROYECTO FINAL GRAFOS

JUAN JOSÉ GONZÁLEZ RÍOS

19/05/2025

INTRODUCCIÓN

A lo largo del último ciclo del primer semestre del año 2025, se estuvieron trabajando varias formas de estructuras de datos enlazadas, entre las cuales se incluían los grafos. Los grafos son estructuras de datos que representan objetos llamados “vértices” o “nodos”, conectados entre sí a través de “aristas” o “arcos”. Permiten modelar una variedad de situaciones.

En clase fue proporcionada la librería “*TADgrafo.h*”. Esta librería incluía estructuras definidas para los nodos y los arcos, y funciones que permiten las tareas fundamentales, como enlazar, agregar, eliminar y demás. Con las clases siguientes, se fueron incorporando más funciones que hacen algunos algoritmos ya conocidos, como el recorrido BFS (explora nodos nivel por nivel), el recorrido DFS (explora nodos rama por rama), el algoritmo de Dijkstra (la ruta más corta entre dos nodos), el algoritmo de Prim (árbol de expansión mínima), el algoritmo de Bellman-Ford (parecido a Dijkstra, pero puede tomar aristas con negativos) y el algoritmo de Kruskal (parecido a Prim, pero ordenado y no iterativo).

Teniendo una librería completa de estructuras y herramientas para el trabajo con grafos, se presenta el trabajo final compuesto de 3 escenarios, los cuales presentan aplicaciones en múltiples áreas reales.

A lo largo de este documento se explicarán estos escenarios y se hablará de su funcionamiento, sus entradas, sus resultados y sus procesos.

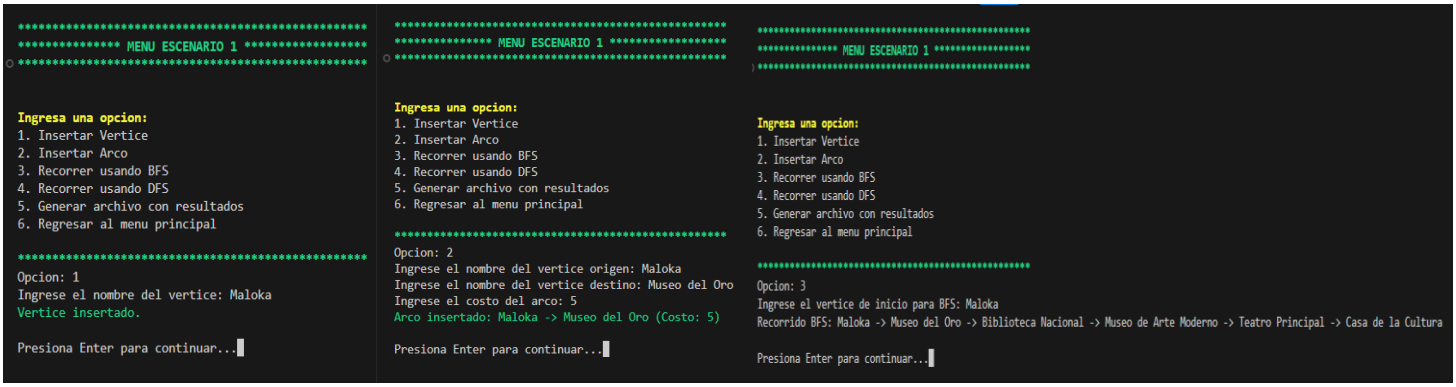
ESCENARIO 1 – RECORRIDOS EN GRAFOS

Propósito: Explorar un grafo para obtener el orden en que se visitan sus nodos, bien nivel a nivel (BFS) o rama a rama (DFS), lo cual es útil para aplicaciones como búsquedas por proximidad (mapas de ciudades, redes sociales) y detección de componentes.

Funcionamiento:

1. **Inserción:** Al usuario se le da la opción de insertar un vértice o un arco. En el caso del vértice, se le pide el nombre del mismo, en el caso del arco, se le pide el nombre de origen y de destino, y el peso del arco. En caso de que haya un error en la entrada (que no exista uno o ambos vértices, o que el peso no sea válido), se le notificará y se abortará el proceso.
2. **Recorridos:** Al usuario se le da la opción de ver el recorrido desde un vértice, sea mediante BFS o mediante DFS. Se le pedirá al usuario el nombre del vértice de inicio, y al igual que en el caso anterior, se abortará en caso de errores en la entrada. Cada uno de los dos recorridos disponibles hará uso de su propio algoritmo definido previamente en la librería.
3. **Generar archivo con los resultados:** En un archivo, que tiene como nombre “resultados_escenario1.txt” se guardan todos los datos: una lista con todas las ubicaciones, una lista con todos los caminos y sus distancias, una lista con los recorridos BFS desde todos los nodos, y una lista con los recorridos DFS con todos los nodos.

Ejemplos de salidas:



ESCENARIO 2 – CAMINOS MÍNIMOS

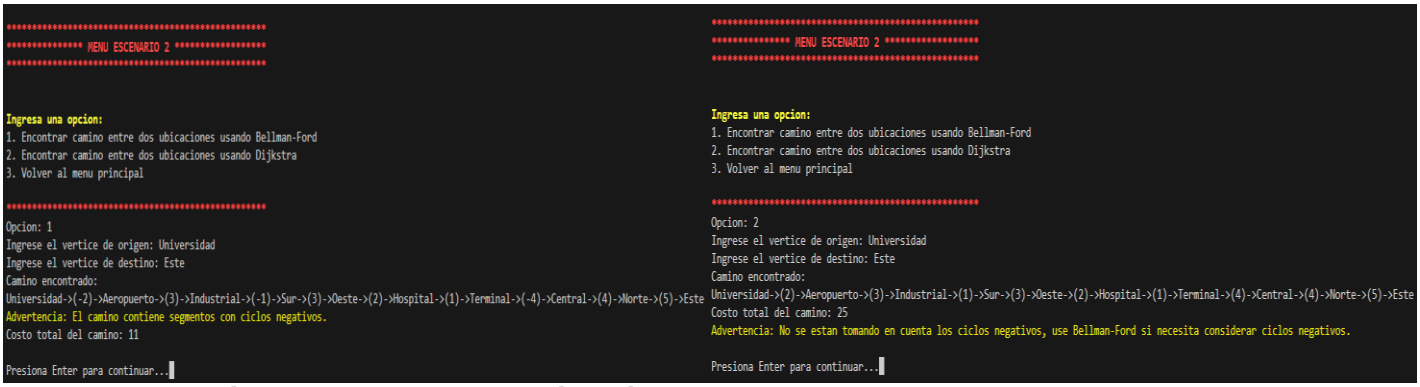
Propósito: Obtener los caminos mínimos entre dos nodos, usando los algoritmos de Dijkstra y Bellman-Ford.

Funcionamiento:

1. **Recorrido con Bellman-Ford:** Se le da al usuario un grafo entero con conexiones entre vértices, y se le pide que escoja dos vértices específicos para encontrar el camino mínimo entre ellos. En el caso de que uno de los caminos tenga segmentos (espacios entre vértices) negativos, se le avisará al usuario.

2. **Recorrido con Dijkstra:** Trabaja de la misma forma que el anterior, pero con una copia del grafo, el cual únicamente tiene segmentos positivos. Siempre se le recuerda al usuario que este algoritmo no toma en cuenta segmentos negativos.

Ejemplos de salidas:



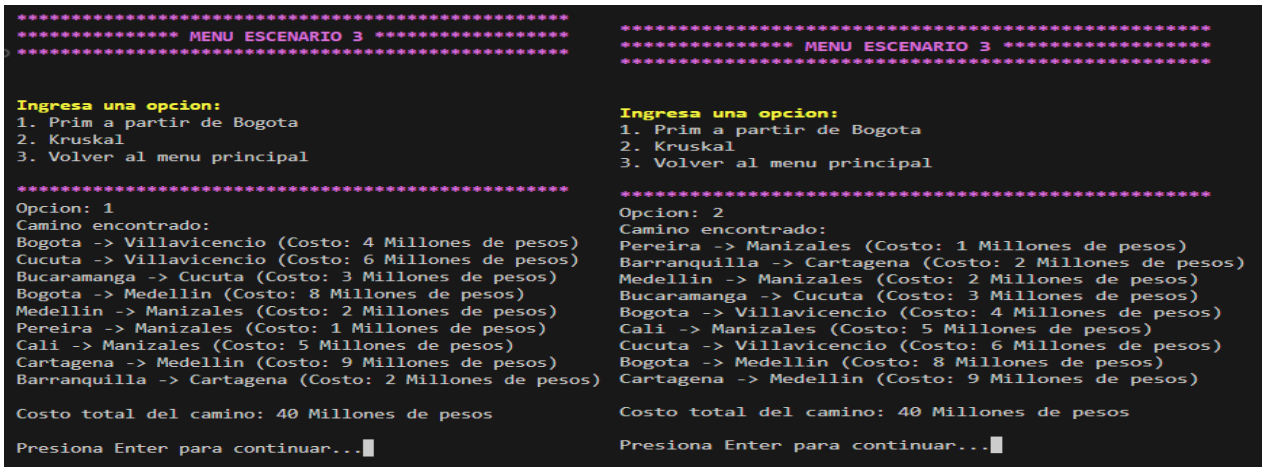
ESCENARIO 3 – ÁRBOLES DE EXPANSIÓN MÍNIMA

Propósito: Construir un subconjunto de arcos que conecte todos los vértices con el coste global mínimo.

Funcionamiento:

1. **Prim a partir de Bogotá:** Se ejecuta el algoritmo de Prim desde un nodo previamente definido, mostrando todos los arcos y el costo total.
2. **Kruskal:** No se pide nodo inicial, ya que no lo requiere. Muestra todos los arcos y el costo total, igual al anterior.

Ejemplos de salidas:



USO DE I.A

Dada la naturaleza del proyecto, y el hecho de que fue realizado en múltiples sesiones y en múltiples equipos, no fue posible recopilar todos los prompts usados. Sin embargo, la mayoría de prompts fueron orientados a los comentarios, a la explicación de algoritmos, a la solución de problemas y a la construcción de ciertas funciones.

COMENTARIOS

Dentro del archivo de cabecera y el archivo principal, cada función, estructura y sección está comentada en formato Doxygen, con ejemplos de uso, explicaciones breves, parámetros y salidas, con la intención de que todo sea entendido de forma ideal y el programador pueda usarlos en otros proyectos.