Alumno: Santiago Obregón
Materia: Programación II

Planteamiento del problema real

En este trabajo se analiza un problema real que puede resolverse utilizando un algoritmo de grafos pesados.
El caso planteado es el siguiente:

Una empresa de telecomunicaciones necesita instalar una red de fibra óptica para conectar distintas ciudades de una región. Cada posible conexión entre dos ciudades tiene un costo asociado, que puede representar distancia, mano de obra, materiales o dificultad de instalación. El objetivo es conectar **todas las ciudades con el menor costo posible**, evitando construir caminos redundantes y garantizando que la red sea eficiente.

Este tipo de problema aparece de manera frecuente en:

- Redes eléctricas
- Redes de transporte
- Redes ferroviarias
- Redes de agua y gas
- Redes de telecomunicaciones

Algoritmo utilizado: Kruskal

Para resolver este problema se utilizó el algoritmo de **Kruskal**, uno de los 3 algoritmos de grafos pesados vistos en clase.

Implementación bajo Tipo de Dato Abstracto (TDA)

Se implementó una solución completamente modular, siguiendo la consigna del trabajo práctico.
Los componentes son:

1) IGrafoPesado

Interfaz que define las operaciones del grafo pesado:

- agregar vértices
- agregar aristas con peso
- mostrar matriz de pesos
- ejecutar Kruskal

2) GrafoPesado

Implementación del grafo pesado usando:

- lista dinámica de vértices
- matriz de pesos
- lista de aristas
- método `aplicarKruskal()`

3) Arista

Representa una conexión entre dos vértices con su peso.
Es comparable para poder ordenarse por costo.

4) UnionFind

Estructura encargada de:

- encontrar el representante de un nodo
- unir conjuntos
- evitar ciclos

Es esencial para el funcionamiento del algoritmo.

5) Main (caso práctico)

Se simula una red de ciudades (A, B, C, D, E, F) con sus respectivos costos de conexión.
El programa:

- carga los vértices
- agrega aristas con sus pesos
- muestra la matriz de pesos
- ejecuta Kruskal
- imprime las aristas seleccionadas
- muestra el costo total del MST

Resultado visual del programa

La salida del programa incluye:

- Matriz de pesos del grafo
- Aristas elegidas por Kruskal
- Costo total mínimo de conexión

Esto permite visualizar claramente cómo Kruskal resuelve el problema real planteado.

Las diferencias principales son:

✓ En clase no se trabajó con un **TDA formal**.
✓ En este trabajo se encapsula todo el funcionamiento en clases separadas.
✓ Se utiliza una **matriz de pesos real**, no solo listas.
✓ Se implementa un **Union-Find completo**, tal como requiere el algoritmo.
✓ La estructura es reutilizable para cualquier grafo pesado.
✓ Se realiza testeo con un programa ejecutable.

Esto hace que la implementación sea más profesional y cercana a un proyecto real.

Algoritmos relacionados observados en la investigación

Durante la investigación aparecieron otros algoritmos importantes de grafos pesados:

• Prim

También calcula un MST.

Diferencias con Kruskal:

- Kruskal trabaja seleccionando aristas por peso globalmente.
- Prim comienza desde un nodo inicial y expande el árbol.
- Prim funciona mejor en grafos densos.
- Kruskal es más eficiente cuando el grafo tiene pocas aristas.

Ambos llegan al mismo resultado, pero por caminos diferentes.

• Dijkstra

Trabaja con grafos pesados, pero **no calcula MST**.

Sirve para determinar el **camino mínimo** desde un nodo origen a todos los demás.
Es útil para:

- GPS
- redes de rutas
- costos de transporte
- logística

Aunque no resuelve el mismo problema, se relaciona porque trabaja con grafos ponderados.

Testeo bajo TDA

El grafo implementado se probó con las 6 ciudades mencionadas.
El testeo permite verificar:

✓ Correcto funcionamiento de la matriz de pesos
✓ Ordenamiento correcto de aristas
✓ Funcionamiento de Union-Find sin ciclos
✓ Selección precisa del MST
✓ Costo total coherente con la estructura del grafo

El TDA se comportó de forma consistente y resolvió el problema de manera correcta.