Trabajo Práctico Obligatorio: Programación II

Integrantes: Maraulo Valentín Franco, Marolda Joaquin, Naboulet Pablo Leon, Piccin Rivera Salvador Guillermo & Puente Valentín Ignacio
Profesor: Perez, Nicolás Ignacio
Asignatura: Programación II
Turno: Viernes - Mañana
Año lectivo: 2025

Índice:

1. Enunciado 2
2. Introducción 2
3. Objetivo 2
4. Solución 3
5. Simulación 3
6. Diferencias con el algoritmo mostrado en clase 4

Enunciado:

- Buscar un problema real que se resuelva con alguno de los 3 algoritmos de Grafos Pesados que vimos en clase.
- Crear o conseguir un programa que muestre este problema resuelto con alguno de estos algoritmos, bajo TDA.
- En el repositorio del Proyecto crear un PDF breve (3/4 páginas) donde se explique el problema, la solución y las diferencias si las hay, con el algoritmo mostrado en clase. Además si en la investigación apareció algún otro algoritmo comentarlo y buscar un testeo bajo TDA.

Introducción:

Hoy en día, la empresa de energía Genneia es la principal encargada de gestionar y brindar electricidad y energía a varias de las ciudades de la provincia de Buenos Aires a través de una central eléctrica que conecta dichas ciudades mediante un sistema de cableado complejo y extenso. Sin embargo, los directivos se han dado cuenta que muchos de los cables del sistema son extensos e innecesarios, y por consecuencia la empresa consume más recursos de los que debe todos los meses.

Es por este motivo que, para mejorar el sistema, se busca eliminar todos los cables innecesarios y de mayor longitud, tal que se logre conectar a todos los pueblos con el cableado justo, necesario y de menor longitud posible.

Objetivo:

Plasmar el problema a un proyecto en Java y aplicar uno de los algoritmos principales vistos en clase para solucionarlo, mediante la implementación de TDA.

Solución:

Para poder plasmar el problema de la empresa Genneia a un proyecto en Java utilizaremos los grafos pesados. De esta manera, un nodo del grafo tendrá como valor la Central Eléctrica, y los demás nodos tendrán como valor las ciudades a las cuales se brinda electricidad mediante la central. Por otro lado, representaremos el sistema de cableado ya existente en la empresa mediante aristas, donde el peso de las mismas representa la longitud de los cables.

El Algoritmo que usaremos para eliminar los cables extensos e innecesarios es Prim, el cual encuentra un árbol de expansión mínima (MST), al igual que Kruskal´s. Sin embargo, llegamos a la conclusión de que Prim es la elección correcta ya que nos permite definir, de manera previa, cuál será el nodo raíz del árbol de expansión mínima, el cual queremos que sea el nodo, que tiene como valor, a la central eléctrica.

Deberemos adaptar el grafo para que sea genérico, y de esta manera pueda funcionar con cualquier tipo de objeto. Por otro lado, también en el proyecto haremos uso de las herencias, es decir, crearemos una clase llamada Ubicación de las cuales se extenderá la clase Ciudad y la clase Central Eléctrica. Gracias a esto, dentro del grafo podremos tener nodos con cualquier instancia de clase que se extienda de la clase base, permitiendo tener un nodo con valor de una instancia de clase Central Eléctrica, y el resto nodos con valores de instancias de clase de Ciudad.

¿Qué ventajas obtendremos al realizar un grafo genérico?

- Mayor escalabilidad de proyecto
- Adaptabilidad
- Vuelve el código más reutilizable

Diseño del software:

El proyecto fue organizado en base al paradigma de TDA, el cual divide el proyecto en los paquetes de:

- Interfaz: es el contrato de nuestro proyecto. Aquí se definen todas las operaciones (métodos) que podrá realizar cada una de las clases que crearemos. Las interfaces no contienen implementación, solo las cabeceras de los métodos (funciones o procedimientos).
- Modelo: aquí implementamos de forma pública los métodos definidos en las interfaces de cada clase, es decir, se escribe el cuerpo de cada uno de los métodos. También declaramos los atributos de las clases como private, respetando así el principio de encapsulamiento, pero controlando su acceso y modificación mediante los getters y setters.
- Test: en este paquete probamos el funcionamiento de la clase grafo.

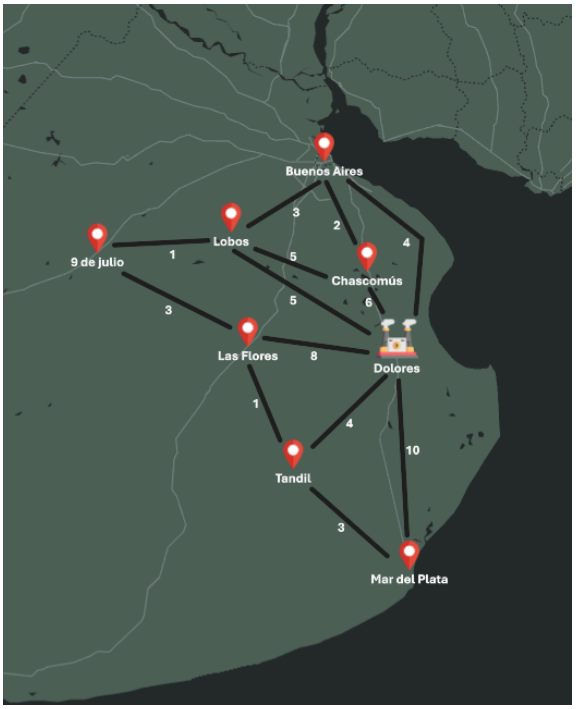
Además, incluimos el paquete de servicios, en el cual se implementa el algoritmo de Prim que permite calcular el árbol de expansión mínima (MST) a partir de un nodo raíz determinado.

Simulación:

1. Entradas ->

Las ciudades a las que la empresa Genneia brinda electricidad son:

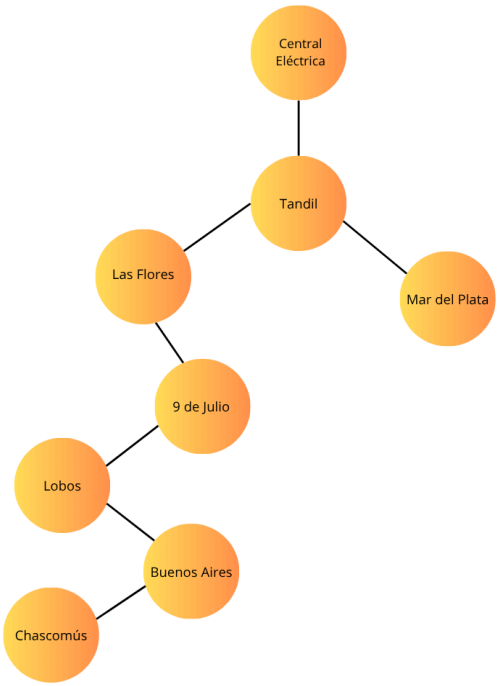
Y el sistema de cableado actualmente es:



2. Salidas ->

Luego de aplicar el algoritmo de Prim, el resultado de salida es:

Y el árbol de expansión mínima es:



Diferencias con el algoritmo mostrado en clase:

Algoritmo	Tipo de algoritmo	Aplicacion tipica	Complejidad algoritmos BigO	Características	Función principal
Prim	Greedy	Redes eléctricas, telecomunicaciones	O(E log V)	Parte de un nodo, expande agregando la arista más barata sin ciclos	Construye un árbol de expansión mínima (MST)
Kruskal	Greedy	Redes eléctricas, telecomunicaciones	O(E log E)	Ordena todas las aristas según su peso, usa Union-Find para evitar ciclos	Construye un árbol de expansión mínima (MST)
Dijkstra	Greedy	GPS, mapas, planificación de rutas	O(E log V)	Encuentra el camino más corto desde un nodo, sin heurística	Obtiene caminos más cortos desde un nodo origen a todos los demás
A*	Búsqueda informada con heurística	GPS, mapas, planificación de rutas	Depende de la heurística, en mejor caso O(E)	Usa heurística (estimación), generalmente más rápido que Dijkstra	Encuentra el camino más corto desde un nodo origen a un destino
Floyd-Warshall	Programación dinámica	Redes de telecomunicaciones	$O(V^3)$	Programación dinámica para calcular distancias entre todos los pares de nodos	Encuentra los caminos más cortos entre todos los pares de nodos

Conclusión:

El proyecto permitió aplicar el algoritmo de Prim en un problema real basado en infraestructura eléctrica, modelado a través de grafos pesados y estructuras genéricas en Java.

Posibles mejoras futuras:

- Incorporar una visualización gráfica del grafo utilizando la librería GraphStream.
- Agregar interfaz gráfica para ingresar pueblos y cables de forma dinámica.