TRABAJO PRÁCTICO OBLIGATORIO:

PROGRAMACIÓN II

Integrantes:

- Giordano, Nicolás
- Lan, Franco
- Mannino, Valentina
- Nava, Vito
- Umansky, Nicolás

Facultad: Ingeniería y Ciencias Exactas

Materia: Programación II

Profesor: Pérez, Nicolás Ignacio

Año: Primer Cuatrimestre 2025

Nro. de grupo: 7

Universidad: Universidad Argentina de la Empresa (UADE)

Índice:

Consigna:

- 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.

Sistema de Asignación de Ambulancias hacia el Hospital más cercano

Aplicación del Algoritmo de Dijkstra para Reducir Tiempos Críticos

Introducción al problema:

En situaciones de emergencia médica, trasladar a una persona al hospital adecuado en el menor tiempo posible es fundamental para aumentar las posibilidades de supervivencia y disminuir la gravedad de las secuelas. Sin embargo, en muchas ciudades de Argentina, el sistema de salud presenta no es eficiente. Las ambulancias suelen tardar demasiado tiempo en llegar al lugar del incidente, poniendo en riesgo la vida de las personas. Según la Radio LU9 de Mar del Plata, las demoras en el traslado de pacientes suelen superar las dos horas, generando una situación crítica para la salud pública.

Si bien lograr que la ambulancia llegue rápidamente al lugar del incidente es un gran desafío, una vez que el paciente está a bordo, es imprescindible optimizar el tiempo de traslado al hospital. En situaciones de vida o muerte, la elección del hospital más cercano y el cálculo de la ruta más eficiente pueden salvar vidas.

El objetivo del trabajo es implementar una solución tecnológica utilizando el algoritmo de Dijkstra que permita identificar automáticamente el hospital más cercano desde el BaseAmbulancia y determinar el camino óptimo en términos de tiempo.

Solución propuesta:

El algoritmo de Dijkstra es un método clásico para encontrar el camino más corto en un grafo con pesos positivos. Propuesto por Edsger W. Dijkstra en 1956, funciona explorando sistemáticamente los nodos con menor coste acumulado desde el origen y analiza los costes de sus vecinos hasta determinar el camino más eficiente.

Modelado del problema:

Primero desarrollamos una base que implementa el algoritmo de Dijkstra sobre un grafo pesado. A partir de esta base, implementamos la idea elegida: un sistema de asignación de rutas para ambulancias que permita determinar el hospital más cercano.

El mapa de la ciudad se modela como un grafo dirigido y ponderado:

- **Vértices (nodos):** representan intersecciones, hospitales, plazas, escuelas y otros puntos clave de la ciudad. En nuestro sistema, hay cinco hospitales disponibles, cada uno definido con su nombre, ID y dirección.

- **Aristas:** representan las calles que conectan los distintos vértices. Cada arista tiene un peso que representa el tiempo estimado de recorrido, generado aleatoriamente en la simulación para modelar variaciones en las condiciones del tránsito.

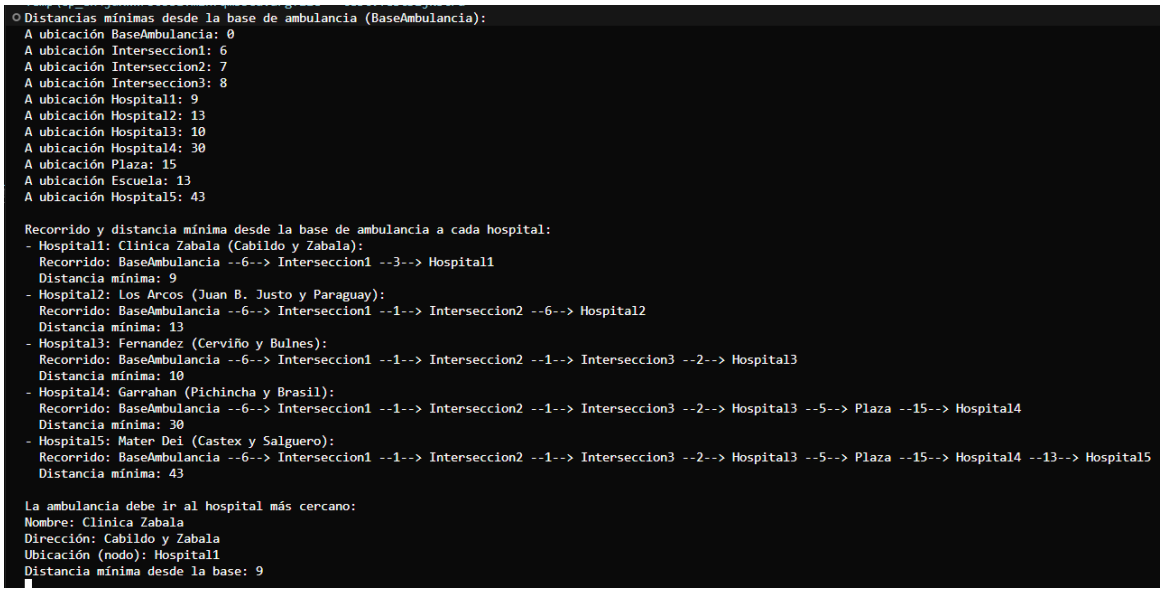
- **Pesos:** tiempos de viaje estimados bajo condiciones normales, que pueden ser adaptados dinámicamente si se dispone de datos de tráfico. La generación de los pesos de las aristas se realiza de manera aleatoria en cada simulación, para que existan distintas condiciones del tránsito.

- **Entradas:**

  - El grafo representando el mapa de la ciudad, con sus nodos y aristas.
  - El nodo de origen (BaseAmbulancia).
  - El conjunto de nodos destino (hospitales disponibles).

- **Salidas:**

  - El hospital más cercano al BaseAmbulancia.
  - La secuencia de nodos que representa el camino óptimo.
  - El tiempo total estimado del traslado.



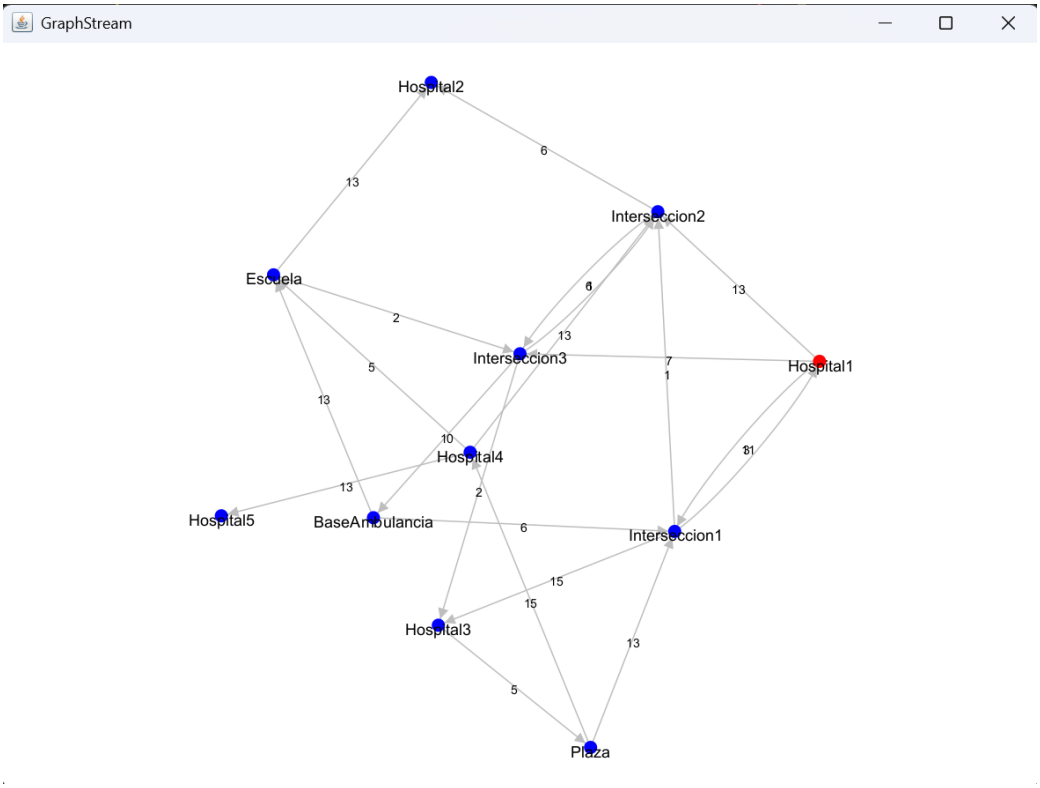
Virtudes del diseño e implementación:

Desarrollamos el programa aplicando buenas prácticas de programación.

- **Uso de TDA (Tipo de Dato Abstracto):** el grafo, los nodos y las aristas se implementaron como tipos abstractos que encapsulan los datos y definen operaciones específicas, asegurando modularidad y flexibilidad. Además, incluimos precondiciones, postcondiciones y axiomas.
- **Getters y Setters:** se emplearon métodos de acceso para preservar la privacidad de los datos y mantener el control sobre los valores de los atributos de las clases.
- **Interfaces:** se definieron interfaces para garantizar que las clases cumplan con contratos específicos, permitiendo mayor claridad y potencial extensión del sistema.
- **Generalización mediante <T>:** utilizamos variables genéricas para que el programa sea adaptable y extensible para diferentes tipos de datos, aumentando su reutilización y robustez.

Utilización de GraphStream:

Para poder comprender y analizar correctamente el sistema de ambulancias, agregamos la biblioteca GraphStream en la clase VisualizadorGrafo.java. Esta es una herramienta orientada a la visualización de grados en entornos Java. Haberlo incorporado nos permitió representar gráficamente la red de calles, intersecciones y hospitales.



Teníamos que asegurarnos de que el comando de compilación y ejecución incluya correctamente todos los JAR de la carpeta *lib*, es por esto que creamos dentro de la carpeta *.vscode* (Usamos Visual Studio Code) un archivo llamado *t*asks.json, que ejecutara comandos en la terminal para que esto ocurra. También eliminara el contenido de la carpeta bin antes de ejecutar, ya que esto nos trajo problemas. Esto no permite automatizar y simplificar más el código.

Diferencias con lo visto en clase:

En clase trabajamos con distintos algoritmos los cuales podemos comparar con el realizado por nosotros

- Algoritmo A*: es similar a Dijkstra, pero el proceso de búsqueda es más rápido ya que se realiza con heurística. A pesar de que este método sea más eficiente, decidimos no implementarlo porque la heurística se debe recalcular en cada paso.
- Floyd-Warshall: es menos eficiente para nuestro caso, sería útil si necesitáramos conocer todos los caminos mínimos entre todos los pares de nodos.
- Kruskal y Prim: no sirven para buscar caminos entre dos puntos, sino para conectar todo el grafo con el menor costo, es por eso que no es útil para este caso.

Conclusión

Haber realizado este trabajo nos permitió poner en práctica todos los conocimientos aprendidos en clase gracias a plasmarlos en un caso real. Nuestra decisión se basó en una problemática que, a pesar de que todo avance, sigue estancada sin lograr encontrar una solución óptima. El algoritmo de Dijkstra nos resultó efectivo para buscar el hospital más cercano en momentos de emergencia.

Bibliografía

Radio LU9 Mar del Plata. *Grave crisis en la salud municipal: las ambulancias tardan más de dos horas y no hay respuestas.*

https://www.lu9mardelplata.com.ar/nota-grave-crisis-en-la-salud-municipal-las-ambulancias-tardan-mas-de-dos-horas-y-no-hay-respuestas-15112

Free Code Camp. *Información sobre Dijkstra.*

https://www.freecodecamp.org/espanol/news/algoritmo-de-la-ruta-mas-corta-de-dijkstra-introduccion-grafica/