Trabajo Práctico Obligatorio

Integrantes: Matias Sanmiguel(1191950), Lisandro Ramos(1191701), Augusto Miguez(1190581), Matias Miranda(1194124), Juan Cegielski(1190183)

Docente: Esp. Lic. Nicolas Ignacio Perez

Universidad: Universidad Argentina De la Empresa

Facultad: Ingeniería y Ciencias Exactas

Materia: Programación II

Cursada: Día viernes, turno mañana

Año: 2025

Fecha de entrega: 27/06/2025

Problemática y solución:

La problemática la cual hemos buscado aliviar a la hora de realizar este proyecto es la demora la cual se genera a la hora de enviar un mensaje hacia un receptor, este retraso del mensaje era generado por la ineficiencia con la cual el mensaje viajaba desde el emisor hasta el receptor mismo.

La solución que propusimos consiste en encontrar el camino de mayor eficiencia desde el emisor hasta el receptor, de forma tal que se minimice el tiempo de demora entre ambos actores. De esta manera, el mensaje evitará trayectos redundantes o ineficientes, utilizando solo el recorrido mínimo e indispensable. Esta solución fue implementada en forma de una red Onion, una arquitectura que además de priorizar la eficiencia, mejora la privacidad del mensaje durante la transmisión del mismo .

Una red Onion es un sistema de comunicación que emplea varias capas de cifrado y nodos intermedios conocidos como “relays” para transmitir información de manera segura y anónima. Cada nodo en el trayecto conoce solo al nodo que le precede y el que lo precede, pero no el origen ni el destino mensaje, lo cual protege la identidad de tanto los emisores como los receptores.

Algoritmo utilizado:

La manera por la cual hemos podido llegar a el recorrido de menor tiempo es mediante el algoritmo de Dijkstra, una algoritmo el cual busca la manera de llegar de

uno de sus nodos a cualquier otro por el camino más corto, este algoritmo suele ser elegido especialmente para tareas como estas, donde la búsqueda es desde un nodo inicial hacia cualquier otro nodo, este algoritmo tiene una complejidad describida como O((V + E) log V) donde V es la cantidad de nodos y E son la cantidad de aristas. Las diferencias que posee en relación a el algoritmo Floyd-Warshall son principalmente:

. Los caminos que calcula: ya que mientras que Dijkstra solo calcula desde un nodo hacia todos, Floyd calcula el camino de cualquier nodo hasta otro.

. Eficiencia: Dijkstra tiende a ser de una eficiencia mayor debido a que calcula todos los caminos hacia un solo módulo, Floyd hace ese cálculo para cada nodo en el grafo, lo que suele conllevar una menor eficiencia

. Negativos: una desventaja clave dentro de Dijkstra es la inhabilidad de soportar pesos negativos, cosa la cual Floyd es capaz de soportar

En conclusión: Los algoritmos Dijkstra y Floyd tienen funcionalidades y objetivos distintos, pero debido a las necesidades presentadas en nuestro caso nos vemos altamente inclinados a elegir Dijkstra como nuestro algoritmo a utilizar.

Paradigma de TDA utilizado:

Para que se cumpla el paradigma de TDA se necesita que se cumplan ciertos requisitos para garantizar la organización del código y los datos mediante estructuras que ocultan su implementación interna y muestran solo las opciones que son necesarias para su implementación. Este paradigma implica que se cumplan cuatro requisitos clave, el encapsulamiento, que define qué es lo que el usuario puede y no puede ver para no comprometer el código y su funcionalidad, la abstracción, que hace que el usuario no necesite saber cómo está compuesto el dato internamente y, finalmente ayuda a mejorar la modularidad del programa, haciendo que cada módulo pueda ser reutilizado o cambiado sin romper el resto del programa.

Para cumplir con estos requisitos, comentamos las funciones para que, el que lee el código, pueda entender cada función y su uso, también, se realizaron clases dentro de un paquete de interfaces para que el desarrollador del código sepa qué funciones son las que tiene que realizar dentro de las implementaciones en los paquetes “modelo”, además se encapsuló la lógica interna de los datos para que nadie los pueda editar sin modificar el código fuente.

Aplicaciones posibles:

Las cualidades las cuales el proyecto posee es la eficiencia y seguridad a la hora de enviar mensajes, estas son cosas las cuales se pueden implementar en servicios de mensajería como Telegram o whatsapp, el cual su foco es la seguridad y animosidad, también se puede utilizar para conversaciones de alta sensibilidad como las comunicaciones internas de organizaciones y gobiernos

Ejemplificación:

A continuación daremos una ejemplificación del uso del proyecto:

el programa nos pedirá una cantidad de nodos a generar

Cantidad de nodos: <10>

se generan los nodos y se les asigna un nombre

Nodos generados: [A,B,C,D,E,F,G,H,I,J]

El programa nos pedirá un nodo de origen

Nodo de origen: <E>

Ahora se nos pedirá el destino

Nodo de destino: <A>

Ingresamos el mensaje

Mensaje: <¡Hola mundo!>

Se nos mostrará el camino recorrido

E - - > B - - > H - - > A

Y se nos presentará la distancia tomó su recorrido

Distancia: 8

Conclusiones:

Luego de este proyecto hemos podido diferenciar los diferentes algoritmos y las ocasiones en las cuales usarlos, la generación y aplicación de grafos en la computación y la comunicación, el uso de Git en Java dentro de Eclipse e Intellij, el aprender a trabajar utilizando una cantidad no fija de nodos y el utilizar el paradigma de TDA a la hora de organizar nuestro código, gracias al enfoque que le hemos dado al proyecto también usamos librerías que están en Java como lo es java.util, java.security y javax.crypto.Cipher para poder encriptar y desencriptar los mensajes con los que estamos trabajando. Además, incluimos prioridades dentro de los grafos con los que trabajamos para darle más prioridad al camino elegido para emitir el mensaje.