****

**Universidad Peruana de Ciencias Aplicadas**

**COMPLEJIDAD ALGORÍTMICA CC-184**

**Profesora: Reyes Silva, Patricia Daniela**

**Trabajo Final**

**Integrantes:**

● Espejo Macuri, Paolo Andre

● Gonzales Carrión, Jorge Enrique

● Laguerre Challco, Fabrizzio Hernan

**Sección: CC42**

**Link de repositorio github:** [**https://github.com/Loleao/CuritohMap**](https://github.com/Loleao/CuritohMap)

**Link de video:**

**Link de video exposición:** [**https://www.youtube.com/watch?v=9fguC\_A42qk**](https://www.youtube.com/watch?v=9fguC_A42qk)

**CICLO 2023-1**

**Contenido:**

➔ Descripción del problema

➔ Descripción del conjunto de Datos (dataset)

➔ Propuesta

➔ Diseño aplicativo

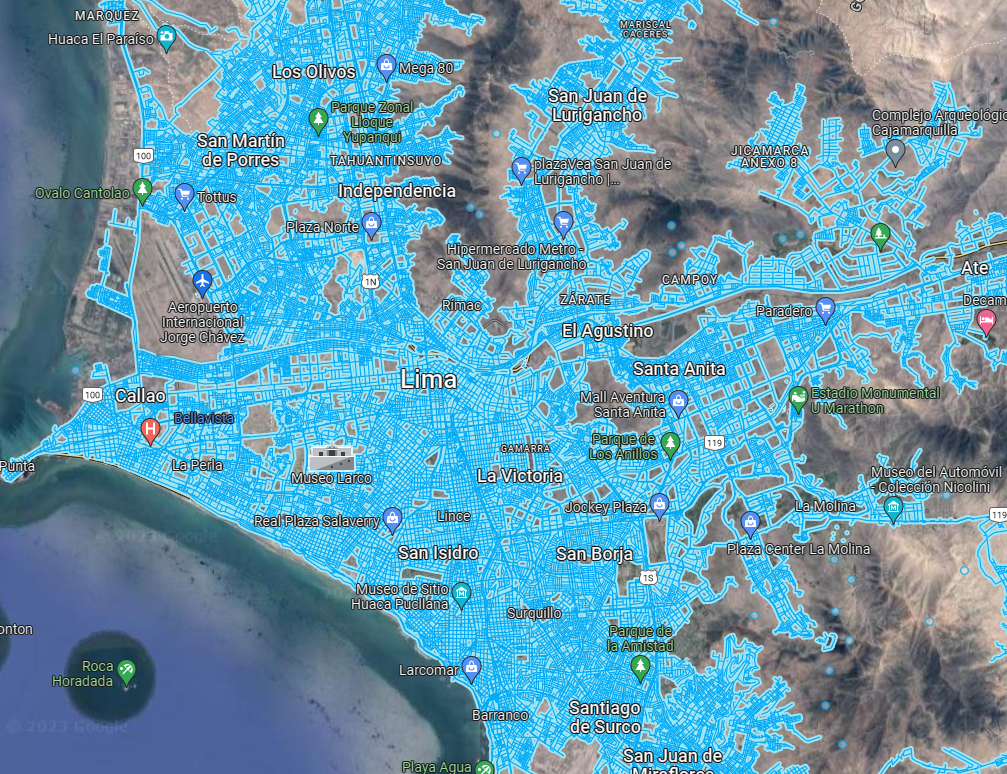
➔ Validación de Re

➔ Conclusiones

**Descripción del problema**

Nuestro Trabajo Parcial se enfocará en encontrar la ruta más corta entre distintas intersecciones de calles o direcciones, añadiendo la posibilidad de ignorar las calles que el usuario introduzca, debido a razones de tráfico u otras, utilizando información obtenida de Google Maps. Cada punto será representado como un nodo en un grafo, mientras que la distancia entre ellos será representada por una arista cuyo valor se determinará por la distancia entre dos nodos. Elegimos utilizar el modelo de Google Maps para este proyecto debido a su similitud con un sistema de grafos, lo que nos permitirá representar las direcciones de manera efectiva utilizando la información recopilada por Google Maps. Esta herramienta ofrece una gran utilidad al usuario final, ya que no solo proporciona información sobre rutas, sino también sobre calles congestionadas de tráfico y cualquier tipo de información georeferenciable. (Pérez, Gielen, Péres, Pérez, 2019, pág. 3)

**Imagen N° 1: Mapa de las carreteras de Lima Metropolitana**



Fuente: Google Maps

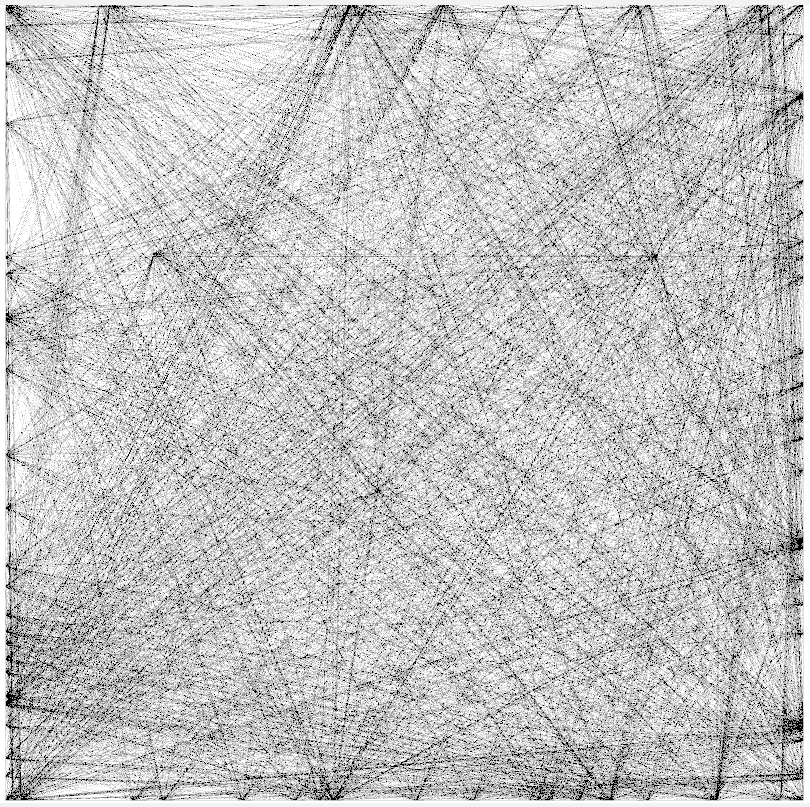
**Descripción del conjunto de datos (dataset)**

Nuestro dataset contiene la variable de direcciones de las intersecciones entre calles de varios distritos de Lima Metropolitana. Además, también trabajaremos con las variables de latitud y longitud, para poder ubicarlas dentro de la implementación BackEnd del producto solución, aproximadamente hay 1500 nodos.

Debido a que el objetivo del trabajo es encontrar la ruta más corta entre dos puntos, los cuales serán recopilados por medio de Google Maps, estos puntos tienen que poseer como principal característica ser identificables dentro del sistema de navegación Google Maps

* Nodos: Conformados por las direcciones como intersecciones con avenidas, calle, etc.
* Aristas: Distancia en kilómetros entre un punto a otro. Nos basaremos en datos como la longitud y la latitud.

**Imagen N° 2: Representación de los 1500 nodos en forma de imagen**



Fuente: Elaboración propia

**Link del dataset :**

<https://docs.google.com/spreadsheets/d/1bzdtLtELyB9NEIul0R5mbFvfdzoE7M0v/edit?usp=sharing&ouid=108106892070307931768&rtpof=true&sd=true>

**Imagen N° 3: Dataset a utilizar del proyecto**



**Propuesta**

Nuestra propuesta se llama “CuritohMaps” consiste en utilizar la teoría de grafos para organizar los datos recolectados de Google Maps, y para el recorrido de grafos, hemos optado por el algoritmo de Dijkstra. Este algoritmo fue creado por Edsger Dijsktra en 1959. Tiene una complejidad Big O de O((|A|+|V|) log|V|). Según Restrepo y Sanchez (2004), el algoritmo de Dijkstra sirve para encontrar la distancia más corta de un nodo origen a un nodo cualquiera, donde se tiene que tener en cuenta la existencia de la distancia asociada en el par de nodos. Debido a ello, este algoritmo encuentra la ruta más corta en un grafo ponderado, donde el peso de las aristas representa la distancia entre un punto a otro y los nodos las direcciones. De esta manera, el usuario puede hallar el camino más corto entre los puntos, en este caso entre dos direcciones, añadiendo la funcionalidad de ignorar algunos nodos introducidos predeterminadamente, de esta manera contamos con las preferencias del usuario para no pasar por la lista de direcciones o calles predeterminadas que la aplicación brindará, estos nodos pueden ser ignorados debido a tráfico o accidentes que el usuario previamente tiene conocimiento (la propuesta no brinda información acerca de estos factores).

**Diseño de aplicativo**

Para el diseño del producto solución tenemos que analizar los requisitos o objetivos a cumplir, entre ellos tenemos:

* El ingreso de un punto de partida y el destino
* Encontrar la ruta mínima entre los puntos ingresados por el usuario
* Graficar las conexiones mediante una ruta detallada en un grafo
* Permitir la opción de ignorar nodos
* Implementar un dataset de las calles por un solo distrito a destinos entre el mismo distrito o otros de Lima Metropolitana
* Desarrollar una interfaz simple pero que cumpla con el funcionamiento de ingresar las búsquedas
* Usar archivos .csv para el dataset del programa
* Implementar un algoritmo de recorrido de grafos
* Mostrar en pantalla junto con un mapa donde se visualiza el trayecto

Por otro lado, en la implementación Front-End se utilizó Tkinter y TkInterviemap. ComboChecks, Layers, y TextLayers fueron parte del menú en ventana donde el usuario ingresa el inicio y el fin de la trayectoria deseada, asimismo para confirmar los datos ingresados se usaron botones en el caso de la casilla donde se ponen los nodos que se ignorar. Posteriormente, con el ingreso del inicio y la llegada del recorrido manualmente por el usuario, se mostrará mediante un grafo la ruta más corta hacia el destino deseado. Finalmente el usuario puede visualizar en el mapa el recorrido y también si desea ignorar un nodo o lugar.

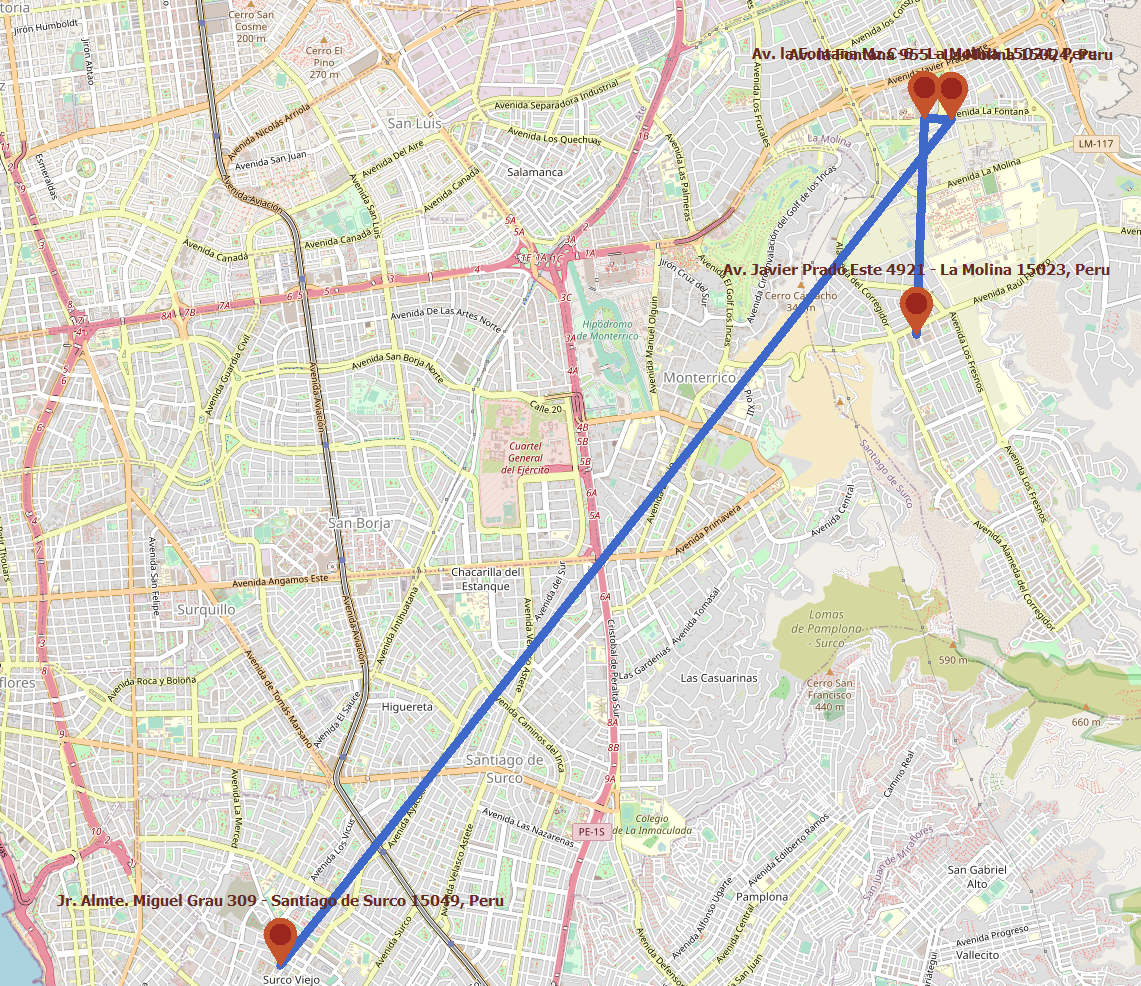
Finalmente, en el Back-End se usó el algoritmo de recorrido de grafos Dijsktra para determinar el camino más corto usando networkx junto con el lenguaje de programación Python. La ruta deseada será producto de una lista obtenida mediante el grafo del distrito y habiendo aplicado la función del algoritmo de Dijsktra perteneciente a la librería networkx. Para la parte visual de los resultados se utilizó TkInter y Tkinterviewmap debido a la amplia selección de widgets y boxes, además sus propiedades para definir pines o marcadores haciendo uso de la latitud y longitud procedentes del dataset, se adecuan para trazar el recorrido dentro de la interfaz.

**Validación de resultados y pruebas**

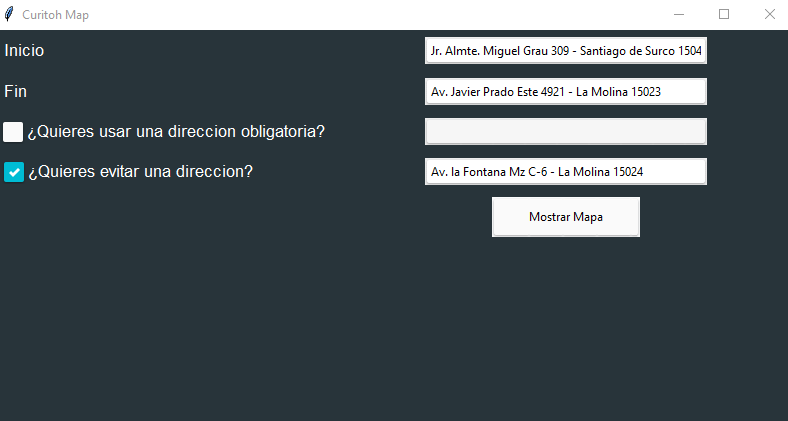
A continuación, vamos a ingresar como lugar de partida “Jr. Almte. Miguel Grau 309 - Santiago de Surco 15049” y como destino “Av. Javier Prado Este 4921 - La Molina 15023”.



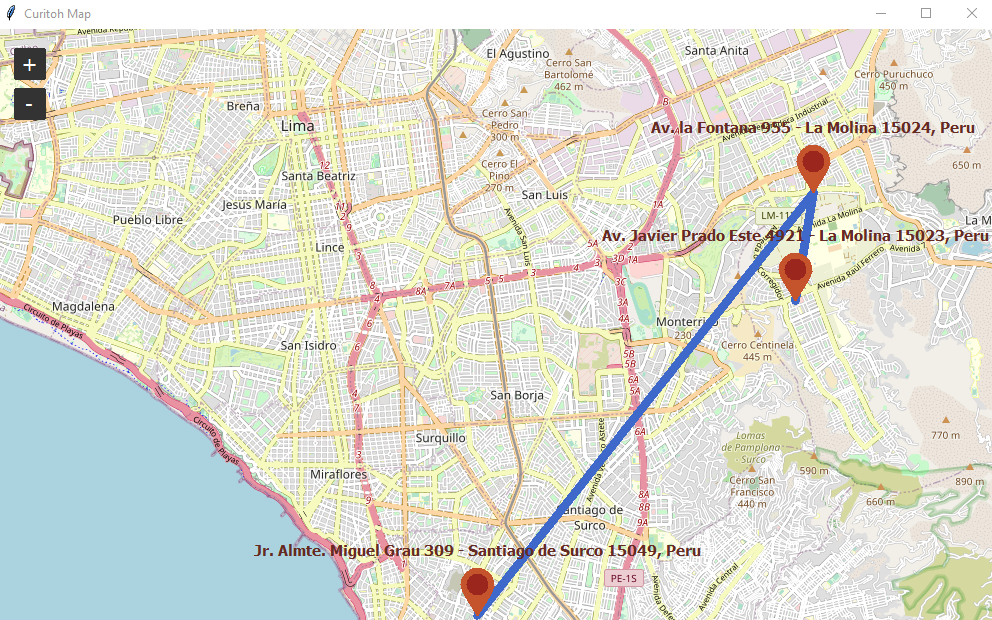
Después de ingresar los datos se mostrará el recorrido y se podrá visualizar la ruta asimismo con el grafo recorrido luego de pulsar el botón “Mostrar Mapa”



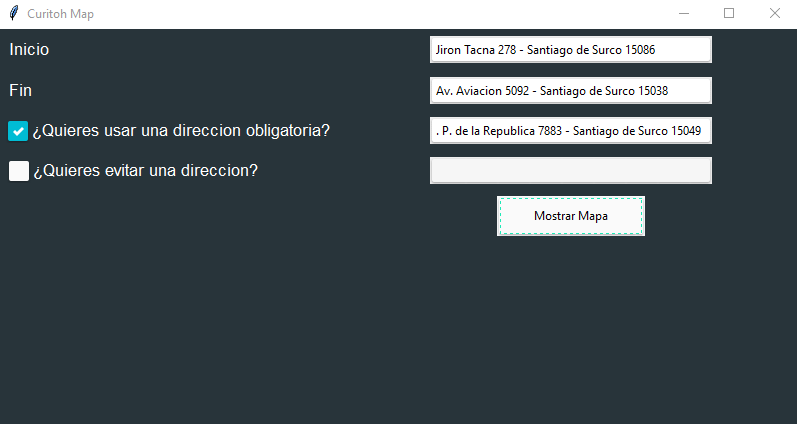
Luego, si el usuario desea evitar una calle, tendrá que marcar la casilla “¿Quieres evitar una dirección?” para habilitar la entrada de texto, luego se ingresará la dirección “Av. la Fontana Mz C-6 - La Molina 15024” esta dirección es parte del recorrido de la búsqueda anterior, debido a ello se comprobará si se eliminará.

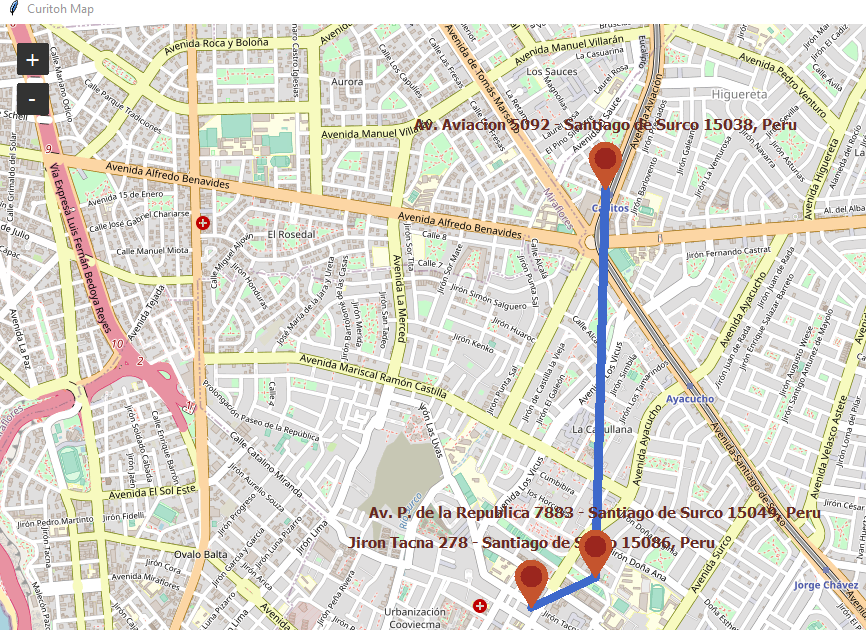


Luego de ingresar la dirección que se quiere evitar, se hace click al botón “Mostrar Mapa”



Finalmente, para usar la opción de “¿Quieres usar una dirección obligatoria?” se tiene que marcar la casilla primero y luego introducir una dirección, por ejemplo “Av. P. de la Republica 7883 - Santiago de Surco 15049” y el punto inicial será “Jiron Tacna 278 - Santiago de Surco 15086” y el final “Av. Aviacion 5092 - Santiago de Surco 15038”, luego le damos al botón “Mostrar Mapa”





Como se observa en el grafo y en la ruta del mapa, se pasa por el nodo “Av. P. de la Republica 7883 - Santiago de Surco 15049” obligatoriamente.

**Conclusiones**

Finalmente, gracias al algoritmo de Dijsktra implementando en lenguaje de programación “Python” se pudo encontrar la ruta más corta entre 2 puntos de interés introducidos por el usuario manualmente añadiendo funciones de ignorar nodos o obligar el recorrido por nodos para solucionar la problemática de encontrar una ruta totalmente personalizada acorde a las necesidades del usuario. Asimismo, los resultados se pueden visualizar de manera gráfica en el mapa donde se detalla el camino mínimo y el grafo usado en una ventana diferente.

Toda la información de las calles ha sido recopilada previamente en un archivo csv. De esta manera se permite la conexión entre distintos distritos o entre calles del mismo distrito, gracias a las variables de latitud y longitud.

Como grupo se puede concluir que esta aplicación ha cumplido con nuestras expectativas y ayudó a alcanzar el objetivo que se planteó, el cual fue de resolver la problemática planteada.

**Referencias Bibliográficas**

Pérez, Y., Gielen, E., Pérez, R., & Pérez, J (2019). *Potencialidades de Google Maps en la investigación social aplicada*. Recuperado de<https://dialnet.unirioja.es/servlet/articulo?codigo=7038144> (Consulta: 25 de septiembre de 2022)

Restrepo, J. H., & Sánchez, J. J. (2004). Aplicación de la teoría de grafos y el algoritmo de Dijkstra para determinar las distancias y las rutas más cortas en una ciudad. Recuperado de <https://www.redalyc.org/pdf/849/84911640021.pdf>