**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media**

**TRABAJO PARCIAL**

CURSO DE COMPLEJIDAD ALGORÍTMICA – CC184

**SECCIÓN: WS6B**

**GRUPO 3**

Pilco Chiuyare, André Dario

Valencia Rivera, Fiorella Viviana

Paiva Quispe, Josue Gonzalo

Romero Polli, Erik

Septiembre 2022

**SuperRoute**

**Descripción del problema**

En nuestra vida diaria solemos trasladarnos de un lugar a otro para lograr algunos de nuestros objetivos cotidianos, por lo que es natural utilizar diferentes medios de transporte para ello, ya sea por la distancia, el clima u otros factores que no nos permitan dirigirnos a los lugares a pie. Según Yarac Abarc, Y. C. y Neyr Yarang, K. L. (2016), el uso del transporte público está aumentando paulatinamente, destacando que este crecimiento, tanto como otros tipos de transporte, como por ejemplo el transporte privado, se ha incrementado en un 40 por ciento desde la década de 1990, lo cual genera de cierto modo congestión vehicular.

Asimismo, es muy común que el tema de encontrar un transporte se complique más en hora punta, si nos referimos a Lima. Esto tiene un impacto importante, ya que muchas personas tienen horarios estrictos, en los cuales es necesario llegar a tiempo, por lo que requieren salir con mucha antelación.

Frente a esta problemática, la solución podría relacionarse con el inminente cambio tecnológico, obteniendo una plataforma con el objetivo de reducir el tiempo que suele tomar transportarse al proponer el recorrido más corto que el conductor podría tomar. Esta idea ayudaría enormemente a la reducción de las dificultades que actualmente Lima sufre con respecto a la congestión vehicular.

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

El dataset utilizado (**Anexo 1**) para el presente proyecto proviene de OpenStreetMap. Este es un proyecto colaborativo de *software* libre que presenta una inmensa base de datos geográfica y un mapa digital mundial. En este caso, hemos decidido tomar la información del distrito de Comas, Lima, Perú. Este dataset presenta siete columnas que representan las variables:

|  |  |
| --- | --- |
| **WKT** | Se refiere a los POINT o LINESTRING, donde los primeros presentan como parámetros a la longitud y la latitud; y los segundos, la latitud y longitud de los puntos inicio y final. Además, cabe mencionar que los POINT equivalen a los 5808 nodos de nuestro grafo; y los LINESTRING, a los 8981 bordes. |
| **osm\_id** | Se refiere a la identificación única del POINT o LINESTRING dentro de la base |
| **highway** | Toma sus valores dependiendo si el POINT es un cruce (crossing), un cruce de autopista (motorway\_junction), un radio de giro (turning\_circle) o una señal de tránsito (traffic\_signals). Por otro lado, si el WKT es un LINESTRING, sus valores dependen si es un boulevard (living\_street), una carretera principal (primary), un enlace principal (primary\_link), una calle residencial (residential), una carretera secundaria (secondary), un enlace secundario (secondary\_link), una carretera terciaria (tertiary), un enlace terciario (tertiary\_link), una carretera troncal (trunk), un enlace troncal (trunk\_link) o desclasificado (unclassified). |
| **lanes** | El número de carriles presentes en la LINESTRING. |
| **maxspeed** | Límite de velocidad máxima en los LINESTRING. |
| **name** | Nomenclatura de los LINESTRING. |
| **oneway** | Parámetro que nos indica si el LINESTRING presenta un flujo motorizado de un solo sentido. |

**Propuesta**

Nuestra propuesta consiste en utilizar los grafos construidos a partir del dataset obtenido de OpenStreetMap para la elaboración de un sistema de viaje similar al de aplicaciones como Waze o Google Maps, para de esta manera permitirle al usuario encontrar una ruta óptima desde su punto de ubicación hasta su destino.

Iniciaremos trabajando con el grafo inicial presentado en la descripción del dataset, este grafo nos permite reconstruir con precisión el distrito de Comas y con él podremos empezar a trabajar la implementación de rutas de viaje en un espacio limitado y reducido, el cual nos servirá para realizar pruebas de manera más eficiente, teniendo como prioridad que la duración del viaje sea lo más corta posible.

***Funcionalidades***

* Presentar la ruta más corta haciendo uso de marcadores de Inicio, Final y Destino Secundario.
* Bloqueo de rutas más congestionadas en función de la hora y de la decisión del usuario.
* Permitir que el usuario escoja entre los algoritmos Dijkstra y A\* para encontrar la ruta más corta a su destino.
* Presentar el ETA (*Estimated Time of Arrival*) y la cantidad de metros de la ruta más corta.

Con respecto a la metodología, hemos utilizado la ágil, la cual se adapta bastante bien al desarrollo de este tipo de proyectos. Seccionamos el trabajo en Sprints para así mantener un control de cada uno de los avances y percibir los mismos como productos funcionales. (**Anexo 2**)

**Diseño del aplicativo**

***Análisis de los requisitos***

Después de analizar la problemática, definimos los objetivos de nuestro proyecto:

* Facilitar la búsqueda del usuario por la ruta más corta tomando en cuenta la congestión de las vías
* Facilitar al usuario la capacidad de presentar un destino secundario por el cual se debe recorrer en la búsqueda de la ruta óptima
* Facilitar el bloqueo de rutas por las cuales el usuario no desea transitar (incluyendo las advertencias de vías congestionadas)
* Presentar una interfaz gráfica intuitiva, minimalista y funcional
* Comparar los algoritmos de búsqueda Dijkstra y A\*

***Diseño del sistema***

**Front-end.** Actualmente utilizamos Ipyleaflet para el mapa interactivo y Panel para los widgets (botones y texto en pantalla). En cuanto a los datos ingresados, es necesario que el usuario modifique la ubicación de los marcadores de Inicio, Final y Destino Secundario arrastrándolos dentro del mapa. Además, es capaz de excluir rutas a través de botones y recomienda eliminar las más congestionadas en función de la hora. En adición, el usuario es capaz de escoger el tipo de algoritmo de búsqueda para encontrar la ruta más corta mediante los botones “Dijkstra” y “A\*”. En cuanto a la salida esperada, el usuario es capaz de visualizar la ruta más corta de color azul o negro (A\* o Dijkstra, respectivamente) en el mapa interactivo, el tiempo de recorrido y la distancia total.

**Back-end.** Para dar solución a esta problemática hemos utilizado el lenguaje de programación Python con sus librerías: networkx, osmnx, ipyleaflet, pandas, warnings, panel y shapely. Además, hemos implementado los algoritmos Dijkstra (O(vértices2)) y A\* (O(aristas + vértices)) para la búsqueda de la ruta óptima.

**Solución de los objetivos.**

* Generamos la ruta más corta haciendo uso de Dijkstra y A\* implementados en NetworkX, además de alertar al usuario sobre las rutas más congestionadas en función de la hora del día.
* El usuario es capaz de agregar un destino secundario con un marcador adicional que modifica la ruta óptima a su destino final.
* Exclusión de vías del mapa modificando las aristas generadas por OSMNX. Por otro lado, las advertencias de las vías congestionadas para emular el tráfico, se basa en las horas punta (7 – 10 horas y 17 – 21 horas).
* El usuario es capaz de manejar nuestro producto en una interfaz gráfica en su totalidad gracias a las librerías Panel y Ipyleaflet, haciendo uso de Widgets y Map, respectivamente.
* El usuario es capaz de visualizar la diferencia entre la distancia y tiempo de recorrido de cada uno de los algoritmos de búsqueda al generar la ruta más corta.

***Implementación***

Para este proyecto hemos utilizado la distribución de Python de Anaconda mediante JupyterLab y Voilá. Además, realizamos pruebas de los algoritmos mediante Google Colaboratory.

Enlace del repositorio de Github: <https://github.com/AndrePilcoChiuyare/SuperRoute>

***Verificación***

|  |  |
| --- | --- |
| **Dijkstra** | **A\*** |
| **Diagrama, Mapa  Descripción generada automáticamente** |  |

|  |  |
| --- | --- |
| **Sin exclusión** | **Con Exclusión** |
|  |  |

|  |  |
| --- | --- |
| **Hora punta** | **Hora valle** |
| Texto  Descripción generada automáticamente |  |

***Mantenimiento o puesta en producción***

Mapa

Descripción generada automáticamente con confianza mediaCódigo desplegado en web mediante Voilá.

**Conclusiones**

Nuestro proyecto verifica y aprovecha la capacidad de los algoritmos de búsqueda, tomando en cuenta a Dijkstra y A\*. Los cuales presentan la misma ruta óptima como resultado, sin embargo, A\* es 60 veces más rápido. Además, SuperRoute es capaz de generar la ruta más corta tomando en cuenta vías excluidas debido a su congestión en función de la hora o por decisión del usuario. En adición, gracias a la metodología ágil, hemos logrado consolidar nuestros conocimientos en Python y optimizar el proceso de trabajo. Como objetivos futuros, nuestro proyecto tiene la capacidad de ser implementado en entorno móvil, lo cual potenciaría su valor agregado.

**Anexos**

***Anexo 1***

**Tabla

Descripción generada automáticamente**

**Anexo 1.1.**

Texto

Descripción generada automáticamenteGracias a la documentación de OSMnx 1.2.2 (librería de Python utilizada para el presente proyecto) presente en la página Read the Docs, obtuvimos el dataset antes mencionado haciendo uso del código:

**Anexo 1.2.**

Mapa

Descripción generada automáticamente

Mapa

Descripción generada automáticamente **Anexo 1.3.** Vista interactiva.

Dibujo de ingeniería, Mapa

Descripción generada automáticamente **Anexo 1.4.** Visualizar nombre de las calles mediante *click.*

**Anexo 1.5.** Visualizar grafo mediante QGIS mediante un archivo GeoPackage.

Mapa

Descripción generada automáticamente con confianza baja

Mapa de colores

Descripción generada automáticamente con confianza media

Imagen que contiene Gráfico de superficie

Descripción generada automáticamente

***Anexo 2***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

***Anexo 3***

**Referencias bibliográficas.**

OpenStreetMap (s.f.). OpenStreetMap. Recuperado de <https://www.openstreetmap.org/#map=14/-12.0783/-77.0808> [Consulta: 23 de septiembre de 2022]

OSMnx (s.f.). OSMnx 1.2.2. Read the Docs. Recuperado de <https://osmnx.readthedocs.io/en/stable/> [Consulta: 23 de septiembre de 2022]

Universidad Don Bosco (s.f.). Tema: Algoritmos para la ruta más corta de un grafo. Recuperado de <https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-iv/2019/ii/guia-10.pdf> [Consulta: 25 de septiembre de 2022]

Vallejo, B. (2021). Shortest path algorithm used in OSM street network. TowardsDataScience. Recuperado de <https://towardsdatascience.com/shortest-path-algorithm-used-in-osm-street-network-33e0f864f34b> [Consulta: 23 de septiembre de 2022]

Yarac, Y., & Neyra, K. (2016). Pertinencia en la adquisición de la flota vehicular para el transporte urbano y nivel de rentabilidad de la empresa Starlet en Lima y Callao. Repositorio de la Universidad Inca Garcilaso de la Vega. Recuperado de <http://repositorio.uigv.edu.pe/bitstream/handle/20.500.11818/1605/TESIS_YESICA%20CLAUDIA%2C%20KATHERINE%20LIZZET.pdf?sequence=3&isAllowed=y> [Consulta: 24 de septiembre de 2022]