Logotipo

Descripción generada automáticamenteImagen que contiene Texto

Descripción generada automáticamente**Problemática:**

Canaval Sanchez, Luis Martin

TRABAJO PARCIAL (HITO 1)

CICLO 2022-2

* Gonzales Astoray, Andrea Abigail / U20211C561/Ciencias de la computación
* Chumán Yallico, Aldair Armando / U202017200 / Ingeniería de Software
* Cardozo Acuña, Jhon Mathyu / U202011302 / Ingeniería de Software

INTEGRANTES

**[Complejidad Algorítmica  CC184-220](https://aulavirtual.upc.edu.pe/webapps/blackboard/execute/courseMain?course_id=_340399_1" \o "Complejidad Algorítmica CC184-2202-WS6A)**

**[2-WS6A](https://aulavirtual.upc.edu.pe/webapps/blackboard/execute/courseMain?course_id=_340399_1" \o "Complejidad Algorítmica CC184-2202-WS6A)**

El Metropolitano es un sistema de buses de tránsito rápido - BRT que opera en la ciudad de Lima y está diseñado para soportar 750 mil usuarios por día. Este sistema está compuesto por corredores exclusivos y estaciones que están ubicadas de acuerdo a su longitud y latitud (*Metropolitano*, s. f.). Pero según Campuzano (2018) menciona que durante el transcurso de los años los usuarios han aumentado demasiado, haciendo que llegue al tope de su capacidad. A razón de esto León (s. f.) sugiere que para atender la demanda de viajes y evitar congestionar el servicio es necesario reprogramar las rutas ya establecidas por unas que sean más cortas, haciendo que el usuario pueda llegar más rápido a su destino.

Por eso nuestro grupo de trabajo buscará una solución al problema de rutas donde lo representaremos a través de un grafo debido a que podemos encontrar caminos óptimos para un mejor trayecto aplicando distintos algoritmos de búsqueda (Villalobos,2006).

Nuestra motivación se dio, dado que como transeúntes se nos ha presentado esta situación de demora para llegar a alguna reunión importante, a la universidad, el trabajo o a un destino a cierta hora establecida aun saliendo con tiempo de anticipación. Por lo mismo, como estudiantes de ingeniería presentamos la solución que vemos pertinente al caso y haremos uso de los conocimientos presentados en el curso.

**Descripción y visualización del conjunto de datos**

En esta oportunidad nuestro grupo diseñará un mapa de grafos conformado por nodos. Estos nodos se representarán en una posición no aleatoria previamente identificada. Para determinar los puntos cardinales de los nodos, se escogerá su latitud en el eje de las abscisas y la longitud en el eje de las ordenadas. Estos datos serán dados por la intersección de dos avenidas en el mapa.

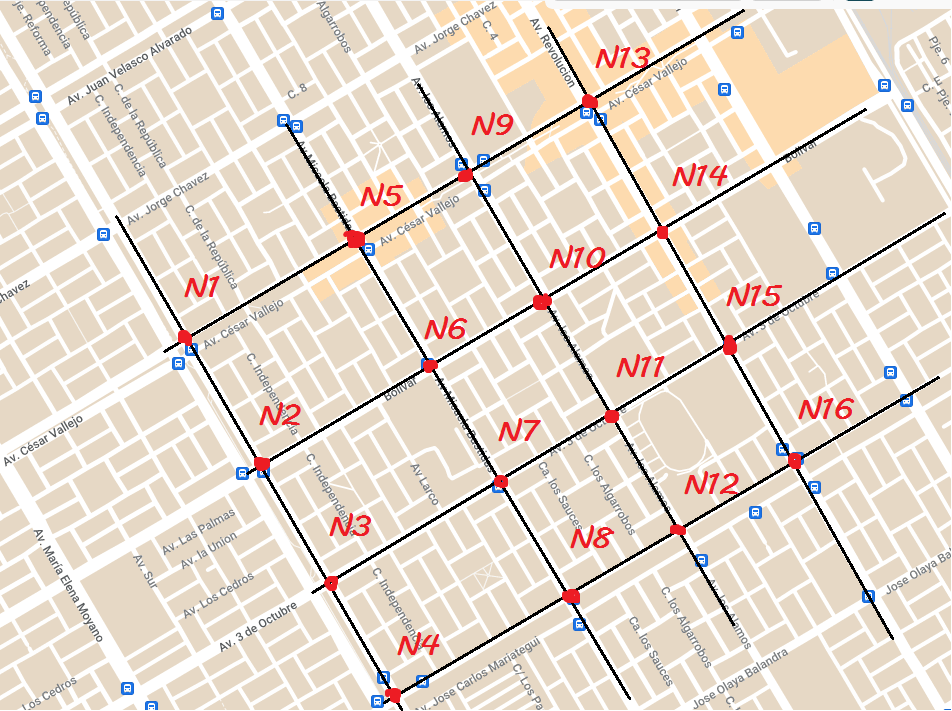


Para ejemplificar nuestra recolección, se puede apreciar la avenida A, de color rojo, y la avenida B, de color azul. La intersección de estas avenidas será nuestro primer nodo, al cual llamaremos N1.

Texto

Descripción generada automáticamente

Paso siguiente, determinaremos su latitud y longitud, los cuales serán asignados a las variables “x” e “y” respectivamente para poder representarlas en el graficador.  
Además, para poder desarrollar una solución lo más cercana a la realidad, se desarrollará la recolección de 1500 nodos, los cuales serán distribuidos por los integrantes del grupo. Por último, se adjunta una imagen de lo que se desea desarrollar.



En la imagen se pueden apreciar 16 nodos, cada uno con una distinta intersección, con un valor de longitud y latitud únicos. Con estos valores, se hallará posteriormente la distancia entre los nodos mediante el teorema de Pitágoras. Como primera instancia se desea obtener 250 nodos por distrito, seis distritos, para poder realizar viajes a lo largo de Lima Metropolitana sin ningún inconveniente.

Estos puntos, a los cuales, al momento de programar, los identificaremos como nodos; de igual manera, los caminos para llegar de un nodo a otro serán los vértices del grafo. En este grafo mencionado se realizará el método de búsqueda que hemos analizado.

**Propuesta**

Como se presentó previamente en el informe, en Lima Metropolitana, según estudios, se toma en ir de un punto a otro un tiempo mayor al estimado por los usuarios. Esa demora se debe a distintos factores, tales como tráfico, cierre de avenidas, marchas políticas, desviaciones, entre otros.

La solución que nuestro grupo ha determinado es la implementación de un algoritmo de búsqueda, para de esta manera asegurarnos que el recorrido que se tomará será el de menor tiempo posible, agilizando la movilización en la ciudad.

Una vez analizados los algoritmos de búsqueda se implementará el algoritmo A\* o también conocido como A star, el cual se explicará brevemente.

Diagrama, Esquemático

Descripción generada automáticamenteA star :; en el cual g(n) equivale a la distancia entre los nodos y h(n) es el peso en el nodo.

En el presente ejemplo se desea ir de A hasta F, su resolución mediante el algoritmo A star sería:

* **Primer tramo**

AB = 3 +8 =11    ;      AC = 2 +7 =9

Se escoge el menor: **AC** (color rojo)

* **Segundo tramo**, se lleva acumulado 9

CE = 9 (acumulado) +4=13

Se escoge el menor **CE** (color verde)

* **Tercer tramo**, se lleva acumulado 13

EB =13 (acumulado) +5 +8 =26    ;    EF =13 (acumulado) +4 =17

Se escoge el menor: **EF** (color rosado)

**Diagrama, Esquemático

Descripción generada automáticamente**

Por último, se hace mención que el desarrollo del código a implementar será en Python y en caso se busquen repositorios externos, en fin, de guía, serán mencionados con sus autores y en el anexo de fuentes.

**App Diseño**

* **Explicación**

Para este proyecto se usará una interfaz gráfica en Visual Studio 2022 donde se usará el lenguaje de programación Python. En este apartado podrá ingresar los vértices de origen y destino mediante la elección de nombres de estaciones de buses.

Para el apartado del algoritmo se usará un dataset en el cual se encuentra las posiciones de las estaciones de buses previamente ingresadas por los integrantes del grupo. Estos vértices se unirán mediante aristas, cuya distancia se hallará mediante el uso de su longitud, latitud y el algoritmo de Pitágoras, que tendrán un peso aleatorio cuyos valores se determinarán según la hora, de un reloj programado.

Para determinar el recorrido de menor costo, usaremos el algoritmo A\*; finalmente, se mostrará el recorrido de menor peso (horas de recorrido) y la distancia entre los dos vértices del grafo no dirigido.

* **Boceto de la interfaz – durante proceso – resultado**

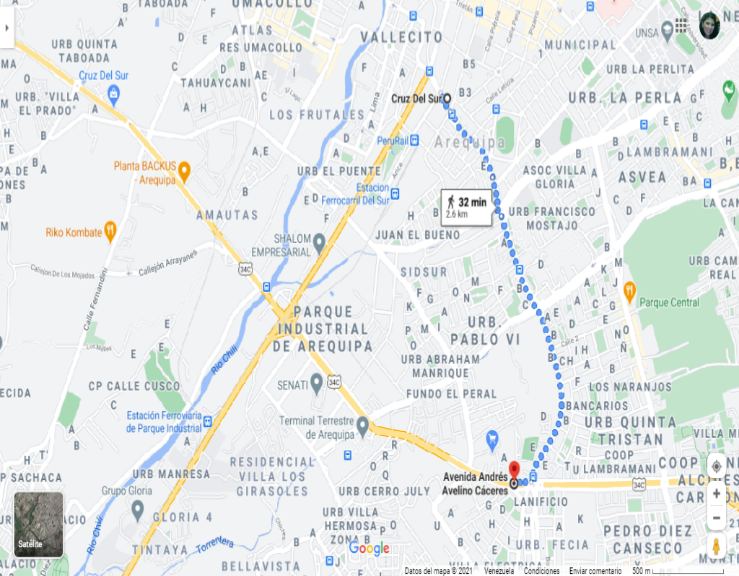
SELECCIONE EL LUGAR DE INICIO

Encuentra la mejor ruta para llegar a tu destino

BUSCAR RUTA

SELECCIONE EL LUGAR DE DESTINO

Resultados de la búsqueda

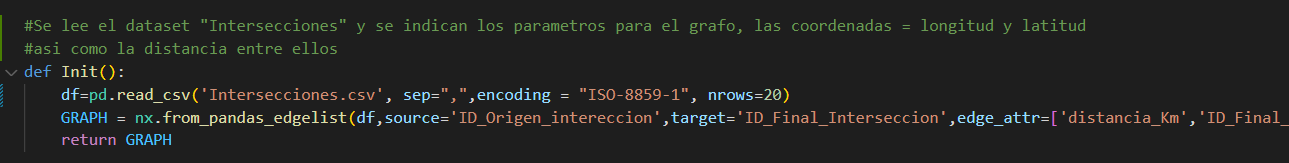


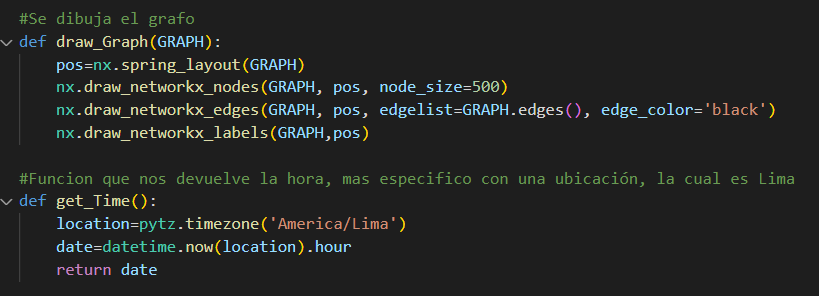
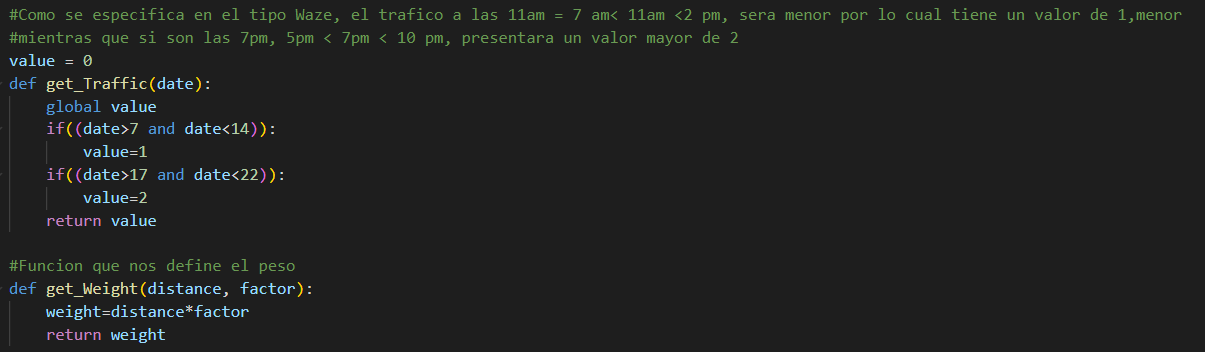
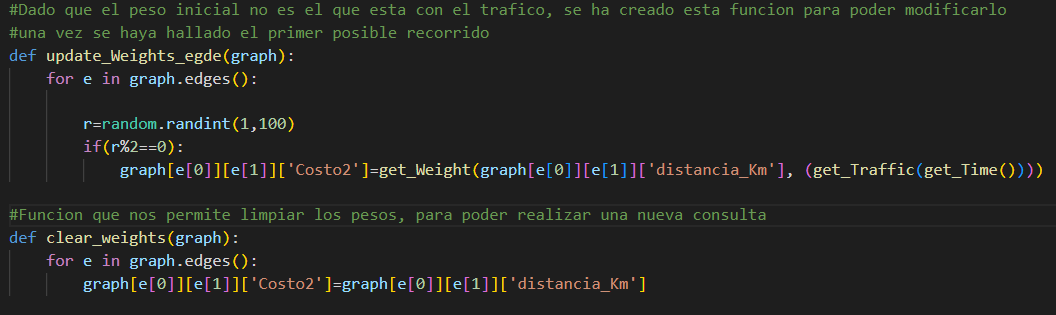




**Validaciones**

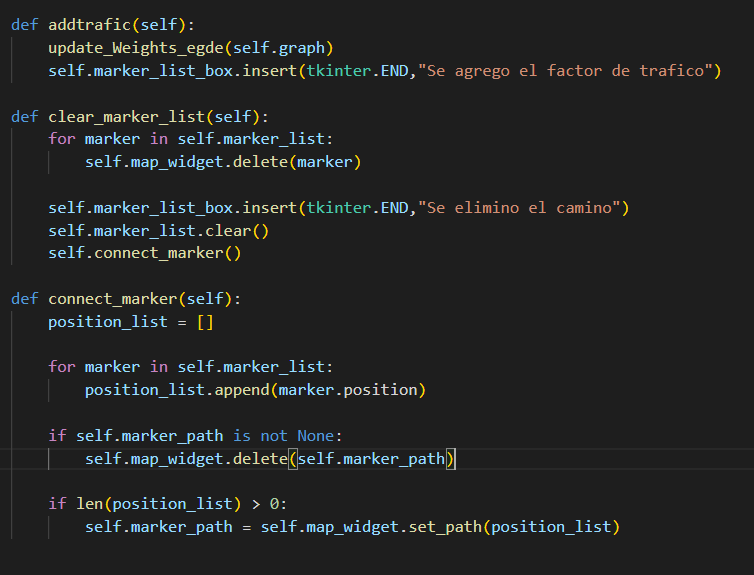
Una vez se definió, lo que se deseaba realizar para la entrega del trabajo final. Se empezó a codificar, en primer lugar, se hizo uso de interfaz gráfica mediante Visual Studio Code y se usó el lenguaje de Python. Para la recolección de datos, en primera instancia se pretendió el de la búsqueda manual; sin embargo, se encontró un dataset, el cual proporcionaba la información requerida e incluso información más detallada. Por esto, se usó el dataset. A continuación, se presentará las funciones creadas para la solución:



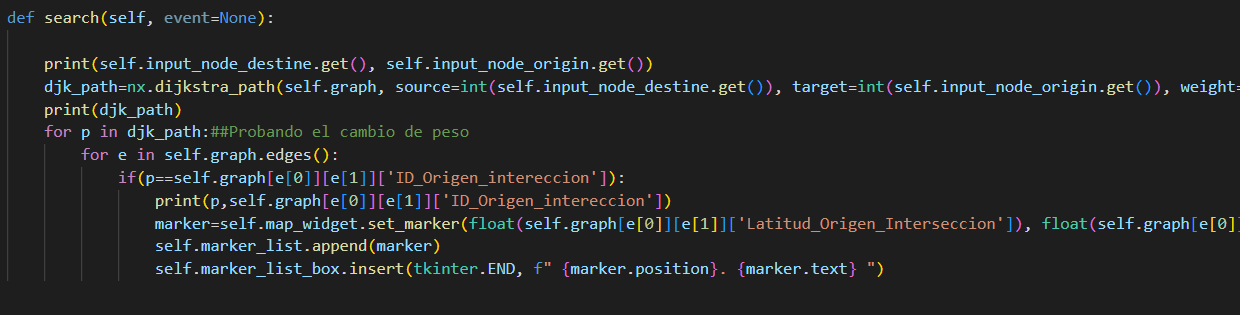
  

Una vez realizada las funciones, se empezó a desarrollar el aspecto visual, por lo cual se consideró la opción de incorporar la opción de poder insertar el punto de inicio como el punto final, un botón para buscar el camino entre estos.

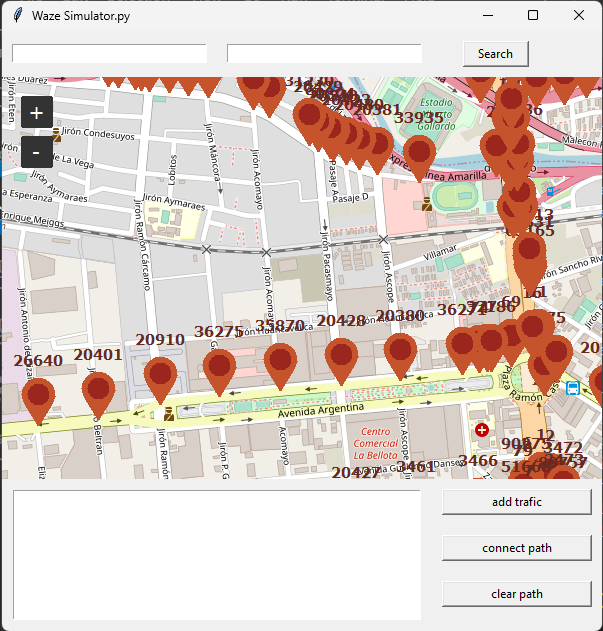
Como en primer lugar, no se toma en cuenta el tráfico se incorporó un botón el cual actualiza los pesos de los nodos, uno que conecta mostrando el recorrido y uno para limpiar el peso, dado que este varia con las horas.

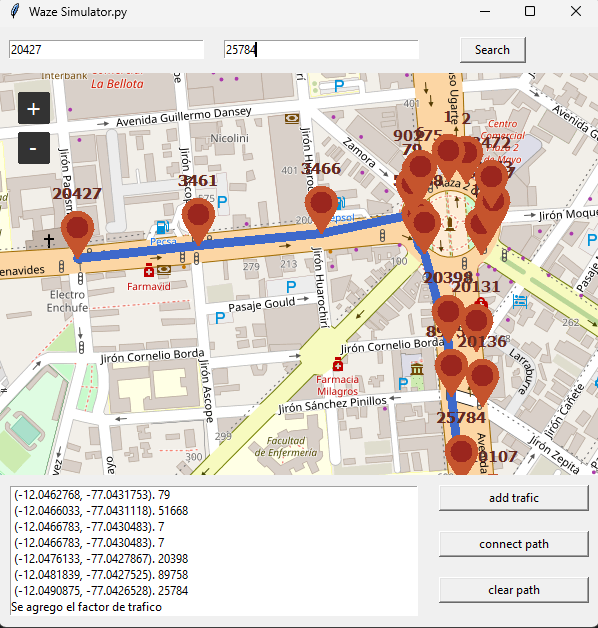


A su vez, se pensó hacer uso del algoritmo de búsqueda A\*; sin embargo, este código nos presentó múltiples problemas al momento de la instancia. Por lo tanto, al momento de investigar, se encontró que existía una librería, la cual permitía hacer uso del algoritmo Dijkstra con parámetros establecido. Al final, se usó este modo para poder hacer la búsqueda.



Como resultado final se obtuvo:





**Conclusiones**

A la primera conclusión que se puede llegar al momento del desarrollo del código fuente, la primera complicación que se nos presento fue la de mostrar un mapa interactivo, no solo puntos. Lo cual se pudo resolver con múltiples pruebas y errores al usar como guía la creación de mapas interactivos para aplicaciones como Uber, Viajes, entre otros, tanto en videos como en código. Al resolver esta complicación, la codificación de las funciones se realizó con sencillez, debido al dataset encontrado, este nos brindó información útil y optimizada. Por último, el problema principal se nos presentó al momento de implementar el código A\* visto en clase, dado que se crasheaba y mandaba múltiples errores. Por conveniencia en tiempo se decidió hacer uso de una librería. Sin embargo, como grupo, consideramos que este sería la primera nueva implementación que se le daría al código; así como la instancia de múltiples algoritmos de búsqueda codeando la lógica sin hacer uso de librerías. Además, nuestro código presenta el problema de que en ciertas ocasiones la búsqueda del recorrido nos manda error, se plantea que esto se debe a la dirección de tránsito, pero no se ha llegado a solucionar en su totalidad siendo el segundo punto a mejorar.

**Anexos**

Github: <https://github.com/GrupoComplejidad>

**Bibliografía**

Campuzano, Ó. P. (2018). Metropolitano al tope: quejas y problemas por alta demanda. El Comercio Perú. Recuperado de:

<https://elcomercio.pe/lima/transporte/metropolitano-tope-quejas-problemas-alta-demanda-noticia-514487-noticia/>

Higgott, O. (2022) PyMatching: A Python Package for Decoding Quantum Codes with Minimum-Weight Perfect Matching. Recuperado de: <https://dl.acm.org/doi/10.1145/3505637>

León, G. (s. f.). Metropolitano: las irregularidades y los motivos detrás del retraso de su ampliación hasta Carabayllo. Recuperado de :

<https://data.larepublica.pe/metropolitano-irregularidades-y-motivos-detras-del-retraso-de-ampliacion-hasta-carabayllo/>

Metropolitano. (s. f.). Recuperado 26 de: <http://www.metropolitano.gob.pe/conocenos/sistema/>

Villalobos, A. R. (2006). Grafos: herramienta informática para el aprendizaje y resolución de problemas reales de teoría de grafos. In X Congreso de Ingeniería de Organización. Recuperado de:

<http://adingor.es/congresos/web/articulo/detalle/a/908>