	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Jesus Cruz Navarro
Asignatura:	Estructura de Datos y Algoritmos II
Grupo:	Grupo 1
No de Práctica(s):	Práctica 7 – Algoritmos de grafos. Parte 2
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	22
No. de Lista o Brigada:	22
Semestre:	2022 - 2
Fecha de entrega:	27 marzo 2022
Observaciones:	

CALIFICACIÓN: _____

Algoritmos de grafos. Parte 2

Objetivos

1. Implementar los algoritmos BFS y DFS en la clase grafo.
2. Generar un programa que, utilizando el metro de la CDMX como un grafo, encuentre la ruta entre dos estaciones mediante los algoritmos BFS y DFS.
3. Comparar las rutas generadas entre BFS y DFS.

Desarrollo

En el programa "practica7.py" se implementó las clases **Grafo** para crear un grafo con múltiples nodos y **Nodo** para crear nodos. El contenido de las clases es:

- **Nodo**
  - Contiene un **nombre** (asignado por el programador), una **lista de vecinos** inicialmente vacía, un **color**, una **distancia** y un **padre** con valores inicialmente *None*.
  - *AgregarVecino(nodo),* agrega un nodo vecino si y solo si no se ha agregado con anterioridad.

- **Grafo**
  - Contiene un **diccionario de nodos** que guarda el nombre del nodo y el nodo.
  - *AgregarNodo(nombreNodo),* recibe como parámetro el nombre del nodo, el método agrega un nodo con el nombre del parámetro recibido en donde se guarda en el **diccionario vértice**.
  - *AgregarArista(nombre_nodo1, nombre_nodo2),* recibe como parámetro el nombre de dos nodos, si estos nodos fueron agregados previamente a través del método *AgregarNodo(),* se crea una "conexión" (arista) entre estos dos nodos y se guarda en la lista de vecinos (que tiene cada nodo) el nombre del nodo contrario.
  - *breadth_first_search*(*nombre_nodo_indicial*): Recorre los nodos del grafo a partir del nodo recibido como parámetro a través del algoritmo **Breadth First Search (BFS)** y guarda las distancias que existen entre los nodos del grafo con el nodo inicial.
  - *encolar(queue, nodo_s):* encola un nodo en la cola.
  - *desencolar(queue):* desencola un nodo de la cola.
  - *depth_first_search(nombre_nodo_incial):* Recorre los nodos del grafo a partir del nodo recibido como parámetro a través del algoritmo **Depth First Search(DFS)** y guarda las distancias que existen entre los nodos del grafo con el nodo inicial.
  - *dfs_visitar(nodo_s):* Recorre los nodos guardados en la lista de vecinos de un nodo y obtiene las distancias entre estas.
  - *encontrar_camino_bfs(nombre_nodo_inicial, nombre_nodo_final):* Encuentra el camino más corto del grafo para desplazarse desde un nodo inicial a un nodo final, utiliza el algoritmo BFS. Retorna el camino.
  - *encontrar_camino_dfs(nombre_nodo_inicial, nombre_nodo_final):* Encuentra el camino más corto del grafo para desplazarse desde un nodo inicial a un nodo final, utiliza el algoritmo DFS. Retorna el camino.
  - *imprimir_camino(camino),* imprime el nombre de los nodos del camino que se recorre para ir desde un nodo A a un nodo B. Recibe como parámetro el camino a recorrer.

El programa es a prueba de errores cuando:

- Un vecino ya existe con ese nombre, cuando sucede se le notifica al usuario. En este ejemplo, ya existen los vecinos Mixcoac (para el nodo San Antonio) y San Antonio (para el nodo Mixcoac).

```
El nodo vecino Mixcoac ya existe
Por lo tanto no se agrega

El nodo vecino San Antonio ya existe
Por lo tanto no se agrega
```

- Se repite el nombre de un nodo, cuando sucede se le notifica al usuario. En este caso, se repite el nombre de aquellos nodos que se encuentran en más de dos líneas del metro, por ejemplo, el nodo Pino Suárez, Hidalgo, Balderas, Candelaria, La raza, etc.

```
El nodo Pino Suárez ya existe
Por lo tanto no se agrego el nodo

El nodo Hidalgo ya existe
Por lo tanto no se agrego el nodo

El nodo Balderas ya existe
Por lo tanto no se agrego el nodo

El nodo Candelaria ya existe
Por lo tanto no se agrego el nodo

El nodo La Raza ya existe
Por lo tanto no se agrego el nodo
```

- Se desea agregar una arista y uno de los nodos no existe, cuando sucede se le notifica al usuario. En este ejemplo, no existe el nodo S4n Ant1ni0.

```
No existe uno de los nodos, S4n Ant1ni0 o Mixcoac
Por lo tanto no se agrego la arista
```

Para hacer uso de las clases, en la función *run()* se instancia un objeto de la clase grafo en la cual, se van a agregar 195 nodos para crear el sistema del metro de la Ciudad de México, el nombre de los nodos es el nombre de cada estación del metro, después se crean aristas entre los nodos de cada línea del metro, como se muestra en la imagen.

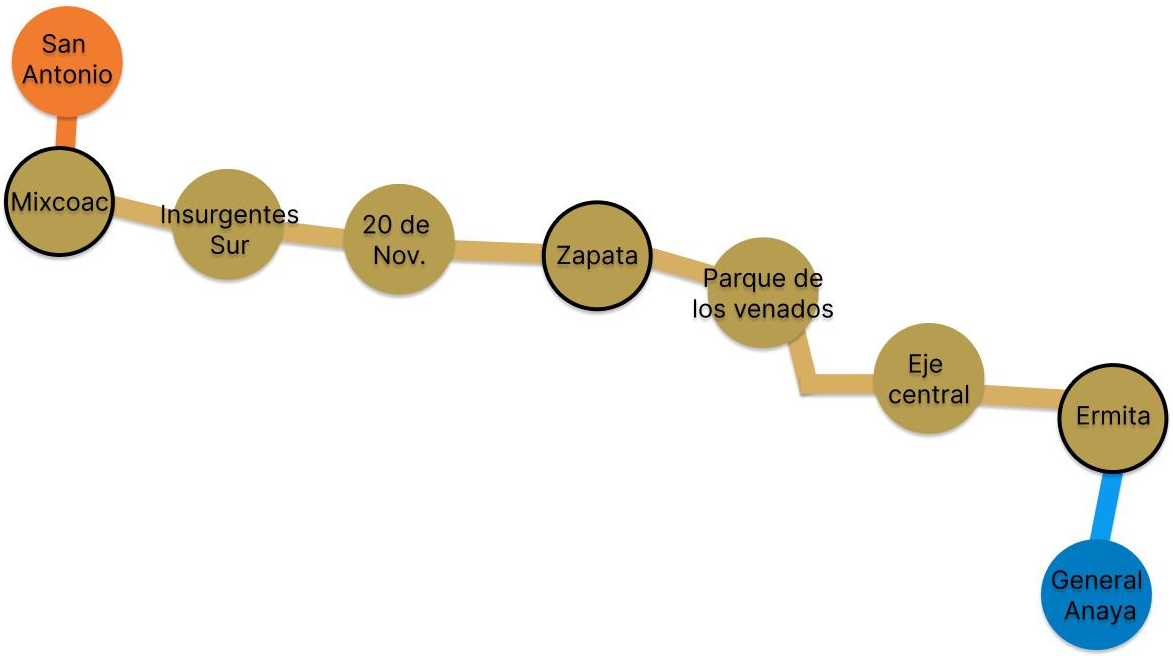

N
El Rosario
Tezozómoc
Ferrería / Arena Ciudad de México
Aquiles Serdán
Azcapotzalco
Norte 45
Vallejo
Instituto del Petróleo
Politécnico
Indios Verdes
Lindavista
Deportivo 18 de Marzo
Martín Carrera
La Villa-Basílica
Autobuses del Norte
Potrero
La Raza
Talismán
Bondojito
Misterios
Valle Gómez
Consulado
Tlatelolco
Guerrero
Garibaldi / Lagunilla
Buenavista
Canal del Norte
Tepito
Morelos
Eduardo Molina
Aragón
Oceanía
Deportivo Oceanía
Romero Rubio
Flores Magón
Terminal Aérea
Hangares
Ciudad Azteca
Plaza Aragón
Olímpica
Ecatepec
Múzquiz
Río de los Remedios
Impulsora
Nezahualcóyotl
Villa de Aragón
Bosque de Aragón
Cuatro Caminos
Panteones
Tacuba
Camarones
Refinería
Cuitláhuac
Popotla
Colegio Militar
Normal
San Joaquín
San Cosme
Hidalgo
Revolución
Bellas Artes
Lagunilla
Allende
Zócalo
Juárez
San Juan de Letrán
Salto del Agua
Balderas
Cuauhtémoc
Pino Suárez
Merced
Isabel la Católica
Doctores
Fray Servando
Candelaria
San Lázaro
Moctezuma
Balbuena
Boulevard Pto. Aéreo
Gómez Farías
Pantitlán
Zaragoza
Velódromo
Ciudad Deportiva
Puebla
Agrícola Oriental
Canal de San Juan
Tepalcates
Guelatao
Peñón Viejo
Acatitla
Santa Marta
Los Reyes
La Paz
Polanco
Auditorio
Insurgentes
Sevilla
Chapultepec
Juanacatlán
Constituyentes
Tacubaya
Observatorio
Patriotismo
Chilpancingo
Centro Médico
Lázaro Cárdenas
Hospital General
Obrera
Niños Héroes
San Antonio Abad
Chabacano
Jamaica
Mixiuhca
La Viga
Santa Anita
Viaducto
Xola
Villa de Cortés
Nativitas
Portales
Ermita
Eje Central
Etiopía / Plaza de la Transparencia
Eugenia
División del Norte
Zapata
Parque de los Venados
Hospital 20 de Noviembre
Mixcoac
Insurgentes Sur
Sn. Pedro de los Pinos
Sn. Antonio
Barranca del Muerto
Coyuya
Iztacalco
Apatlaco
Aculco
Escuadrón 201
Atlalilco
Mexicaltzingo
Iztapalapa
C. de la Estrella
U.A.M. I
Const. de 1917
Culhuacán
Coyoacán
Viveros / Derechos Humanos
Miguel Ángel de Quevedo
Copilco
Universidad
General Anaya
Tasqueña
Conexión con Tren Ligero
Dirección Xochimilco
San Andrés Tomatlán
Lomas Estrella
Calle 11
Periférico Ote.
Tezonco
Olivos
Nopalera
Zapotitlán
Tlaltenco
Tláhuac
GOBIERNO DE LA CIUDAD DE MÉXICO

Ahora, para hacer uso de los algoritmos BFS y DFS queremos obtener el camino más corto para ir desde la estación "**San Antonio**" a "**General Anaya**" utilizando los métodos *encontrar_camino_bfs()* y *encontrar_camino_dfs()*. El programa imprime el siguiente resultado.

```
**San Antonio -> General Anaya**

-----
Breadth First Search

Numero de estaciones:  9
Camino:
San Antonio -> Mixcoac -> Insurgentes Sur -> 20 de Noviembre -> Zapata -> Parque de los Venados -> Eje Central -> Ermita -> General Anaya

-----
Depth First Search

Numero de estaciones: 36
Camino:
San Antonio -> San Pedro de los Pinos -> Tacubaya -> Juanacatlán -> Chapultepec -> Sevilla -> Insurgentes -> Cuauhtémoc -> Balderas -> Sa
lto del Agua -> Isabel la Católica -> Pino Suárez -> Merced -> Candelaria -> San Lázaro -> Moctezuma -> Balbuena -> Boulevard Puerto Aére
o -> Gómez Farías -> Zaragoza -> Pantitlán -> Puebla -> Ciudad Deportiva -> Velódromo -> Mixiuhca -> Jamaica -> Santa Anita -> La Viga ->
 Chabacano -> Viaducto -> Xola -> Villa de Cortés -> Nativitas -> Portales -> Ermita -> General Anaya
```

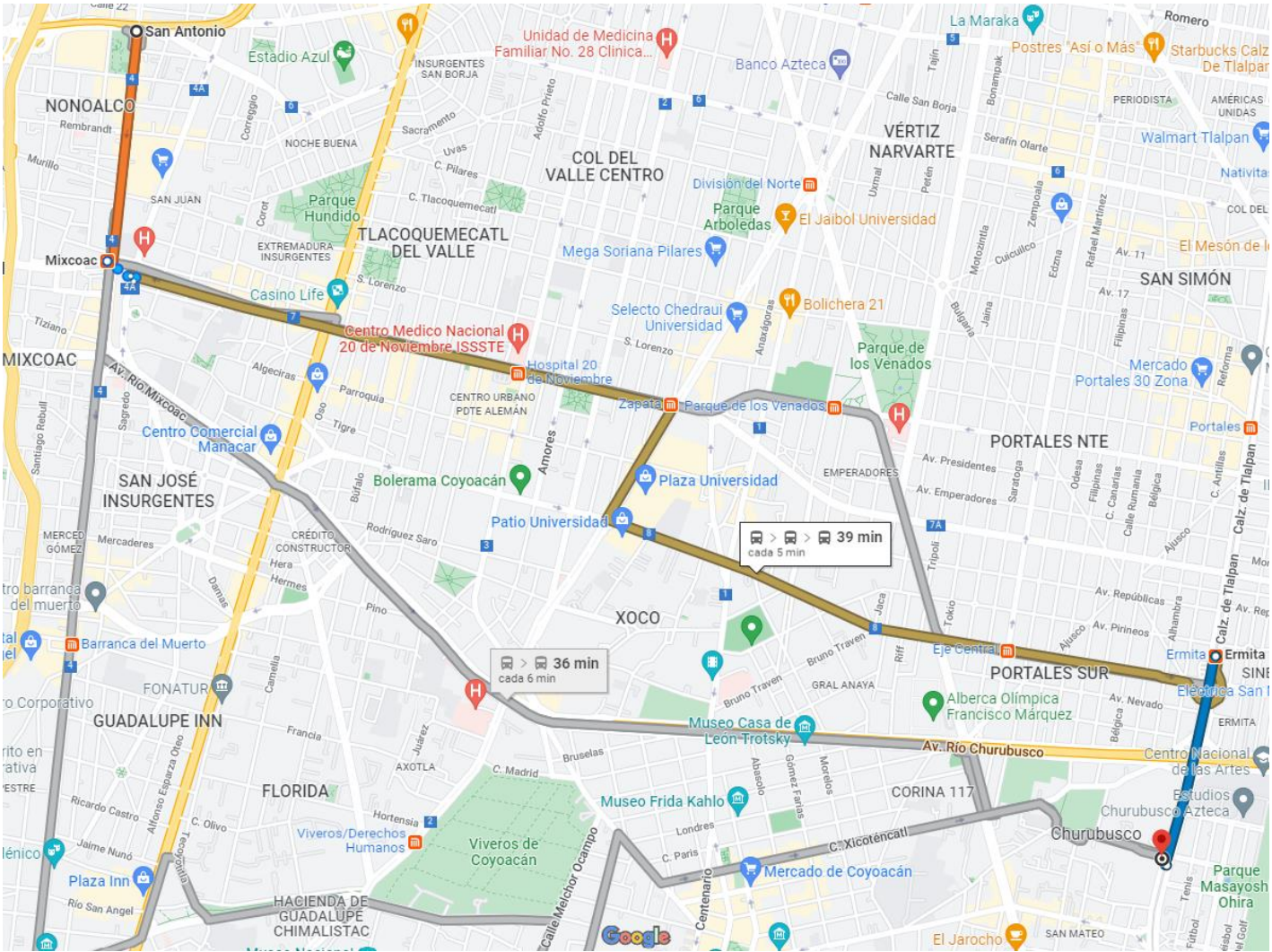
En el grafo creado se encuentran los nodos a recorrer de la siguiente manera.

La representación del camino a recorrer a través del algoritmo BFS es el siguiente.

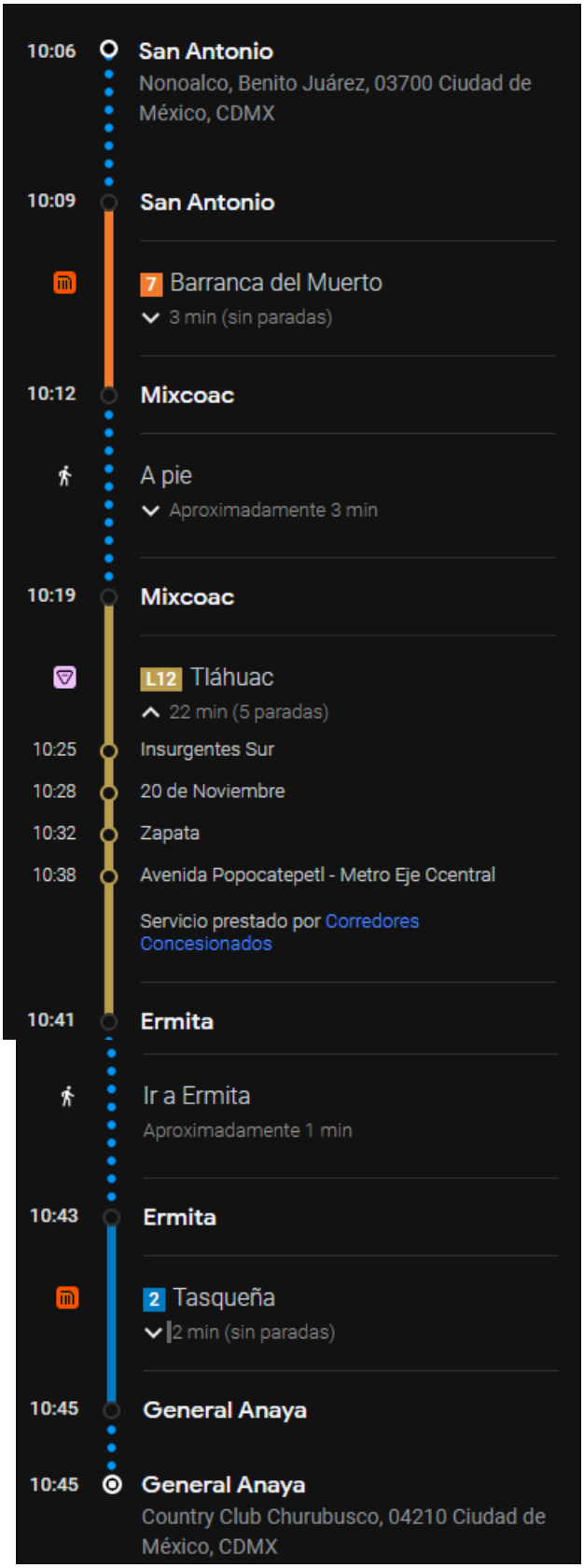


Podemos observar que el camino es corto, consta de 9 estaciones.

Google Maps tiene un sistema en el cual cuando una persona desea desplazarse desde un punto A a un punto B muestra el camino a recorrer usando el transporte público de la ciudad. Con la ayuda de este sistema podemos comprobar el resultado de nuestro programa con el resultado que nos muestra Google.

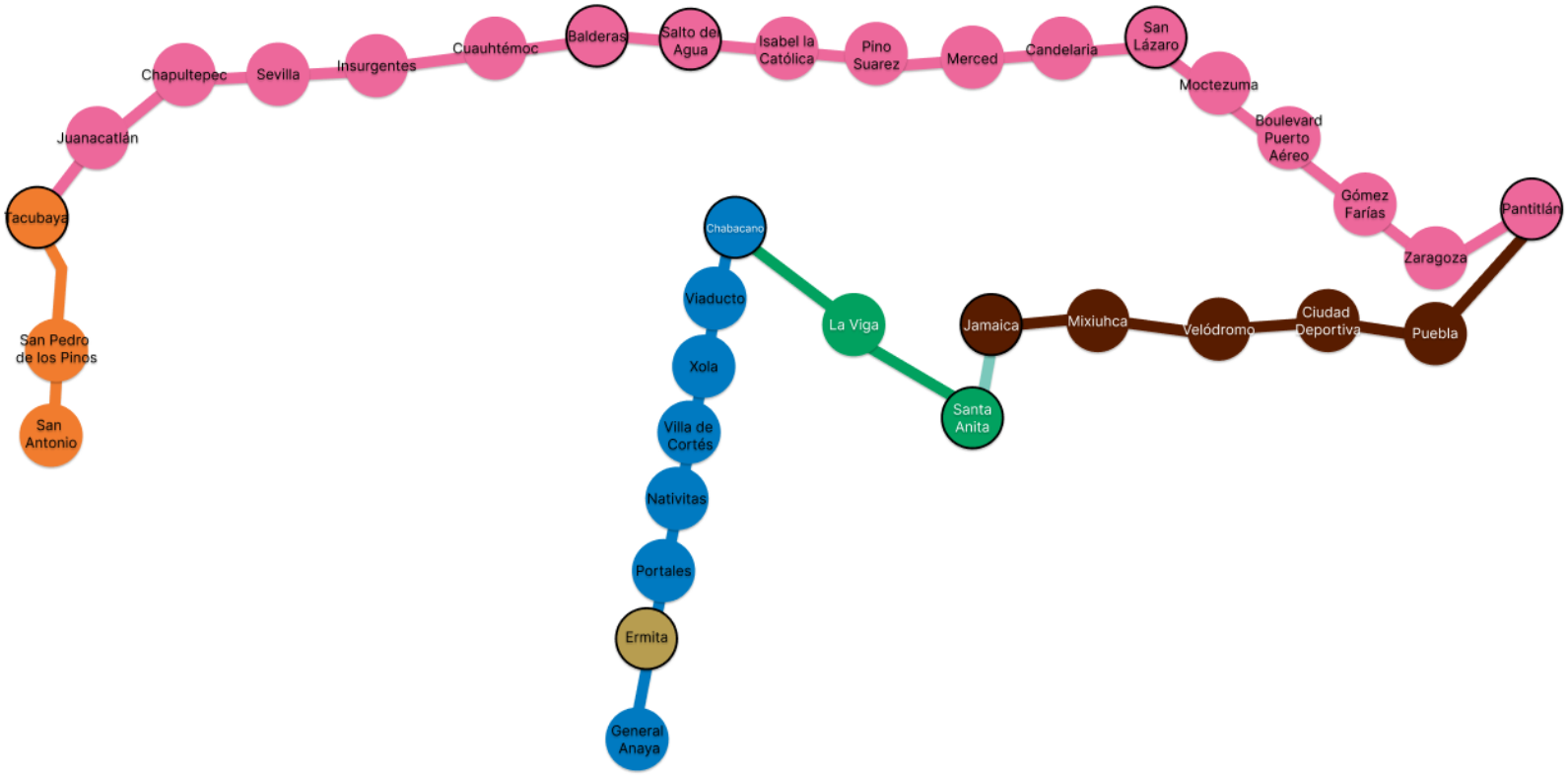


Se puede observar que los caminos son muy similares, sin embargo debemos de comprobar los nombres de las estaciones. Google te indica que estaciones debes de tomar, como se muestra en las siguientes imágenes.



El nombre de las estaciones a recorrer que nos da Google Maps y nuestro programa son las mismas, por lo tanto, podemos concluir que el programa funciona correctamente.

La representación del camino a recorrer a través algoritmo DFS es el siguiente.



Podemos observar que el camino es mucho más largo, consta de 36 estaciones.

Por último, con el programa se buscará el camino más corto con el método BFS y DBS desde “Aquiles Serdán” – “Iztapalapa”, “San Antonio” – “Aragon” y “San Antonio” – “Insurgentes”. Estos son los resultados que nos da el programa.

```
**Aquiles Serdán -> Iztapalapa**

-----
Breadth First Search

Numero de estaciones:  21
Camino:
Aquiles Serdán -> Camarones -> Refinería -> Tacuba -> San Joaquín -> Polanco -> Auditorio -> Constituyentes -> Tacubaya -> San Pedro de l
os Pinos -> San Antonio -> Mixcoac -> Insurgentes Sur -> 20 de Noviembre -> Zapata -> Parque de los Venados -> Eje Central -> Ermita -> M
exicaltzingo -> Atlalilco -> Iztapalapa

-----
Depth First Search

Numero de estaciones: 52
Camino:
Aquiles Serdán -> El Rosario -> Tezozómoc -> Azcapotzalco -> Ferrería/Arena Ciudad de México -> Norte 45 -> Vallejo -> Instituto del Petr
óleo -> Autobuses del Norte -> La Raza -> Potrero -> Deportivo 18 de Marzo -> La Villa-Basílica -> Martín Carrera -> Talismán -> Bondojit
o -> Consulado -> Canal del Norte -> Morelos -> Candelaria -> Merced -> Pino Suárez -> Isabel la Católica -> Salto del Agua -> Balderas -
> Cuauhtémoc -> Insurgentes -> Sevilla -> Chapultepec -> Juanacatlán -> Tacubaya -> San Pedro de los Pinos -> San Antonio -> Mixcoac -> I
nsurgentes Sur -> 20 de Noviembre -> Zapata -> División del Norte -> Eugenia -> Etiopía/Plaza de la Transparencia -> Centro Médico -> Láz
aro Cárdenas -> Chabacano -> Viaducto -> Xola -> Villa de Cortés -> Nativitas -> Portales -> Ermita -> Mexicaltzingo -> Atlalilco -> Izta
palapa
```

```
**San Antonio -> Aragón**

-----
Breadth First Search

Numero de estaciones:  16
Numero de estaciones:  16
Camino:
San Antonio -> San Pedro de los Pinos -> Tacubaya -> Patriotismo -> Chilpancingo -> Centro Médico -> Lázaro Cárdenas -> Chabacano -> Ja
ica -> Fray Servando -> Candelaria -> Morelos -> Canal del Norte -> Consulado -> Eduardo Molina -> Aragón

-----
Depth First Search

Numero de estaciones: 25
Camino:
San Antonio -> San Pedro de los Pinos -> Tacubaya -> Juanacatlán -> Chapultepec -> Sevilla -> Insurgentes -> Cuauhtémoc -> Balderas ->
lto del Agua -> Isabel la Católica -> Pino Suárez -> Merced -> Candelaria -> San Lázaro -> Moctezuma -> Balbuena -> Boulevard Puerto A
o -> Gómez Farías -> Zaragoza -> Pantitlán -> Hangares -> Terminal Aérea -> Oceanía -> Aragón
```

```
**Vallejo -> Insurgentes**

-----
Breadth First Search

Numero de estaciones:  11
Camino:
Vallejo -> Instituto del Petróleo -> Autobuses del Norte -> La Raza -> Tlatelolco -> Guerrero -> Hidalgo -> Juárez -> Balderas -> Cuauhté
moc -> Insurgentes

-----
Depth First Search

Numero de estaciones: 26
Camino:
Vallejo -> Norte 45 -> Ferrería/Arena Ciudad de México -> Azcapotzalco -> Tezozómoc -> El Rosario -> Aquiles Serdán -> Camarones -> Refin
ería -> Tacuba -> Cuitláhuac -> Popotla -> Colegio Militar -> Normal -> San Cosme -> Revolución -> Hidalgo -> Bellas Artes -> Allende ->
Zócalo -> Pino Suárez -> Isabel la Católica -> Salto del Agua -> Balderas -> Cuauhtémoc -> Insurgentes
```

Con los resultados que nos da el programa al usar el algoritmo DFS y BFS podemos observar que el camino más corto se encuentra con el algoritmo BFS debido a que explora todos los vecinos de un nodo mientras que DFS explora los nodos desde el nodo padre hacia el nodo hijo.

Conclusiones

La complejidad de ambos algoritmos es de O(V + E), V es el número de vértices y E el número de aristas.

BFS encuentra el camino más corto desde un nodo A a un nodo B usando el menor número de nodos (comúnmente se aplica para sistemas de navegación de GPS), mientras que DFS es ideal para probar si una solución es ideal entre varias posibles soluciones (comúnmente se aplica en resolución de rompecabezas).

BFS utiliza la estructura de datos “cola” y se basa en nodos, mientras que DFS utiliza la estructura de datos “pila” y se basa en las aristas/bordes.

La practica fue realizada en su totalidad, se cumplieron los objetivos y los ejercicios propuestos por el profesor. En lo personal, me gustó la práctica y quiero implementar el sistema de metro de otras ciudades. Lo que más se me dificultó fue la implementación del algoritmo DFS.