Ejercicio No 4: Algoritmo A*

Nombre:

Walter Bau

Enunciado:

• Diseñe un grafo similar al que se ha presentado en los ejercicios de búsqueda por amplitud y profundidad, partiendo de las siguientes coordenadas de latitud y longitud: -2.8801604,-79.0071712. Para ello deberá realizar las siguientes tareas:

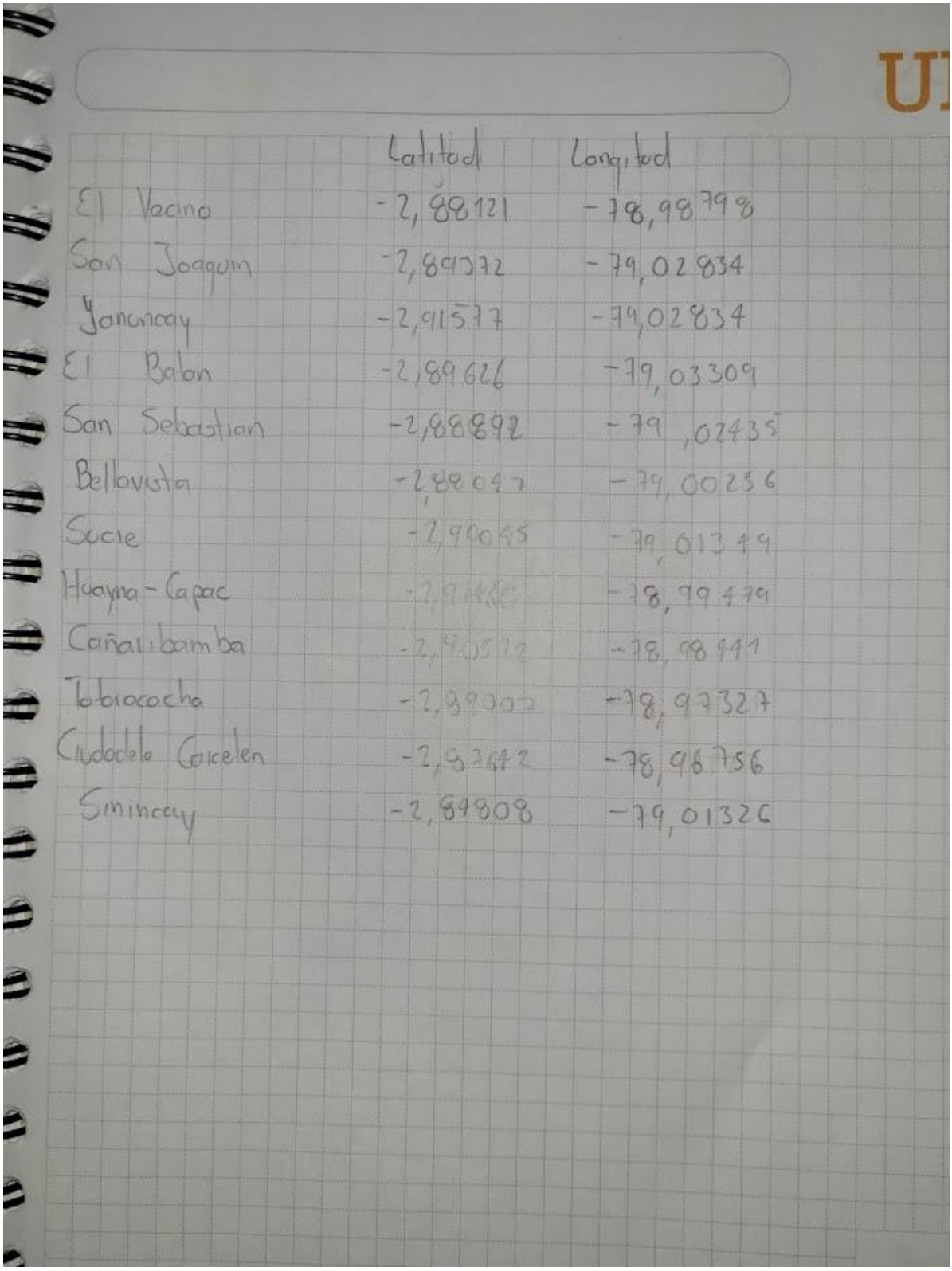
Emplear la herramienta Google Maps (R) con las coordenadas antes indicadas.

In [1]:
```
#IMPORTAR py2neo
from py2neo import Node, Relationship, Graph


# connect to authenticated graph database
graph = Graph("bolt://localhost:7687", aut="neo4j", password="lu
```

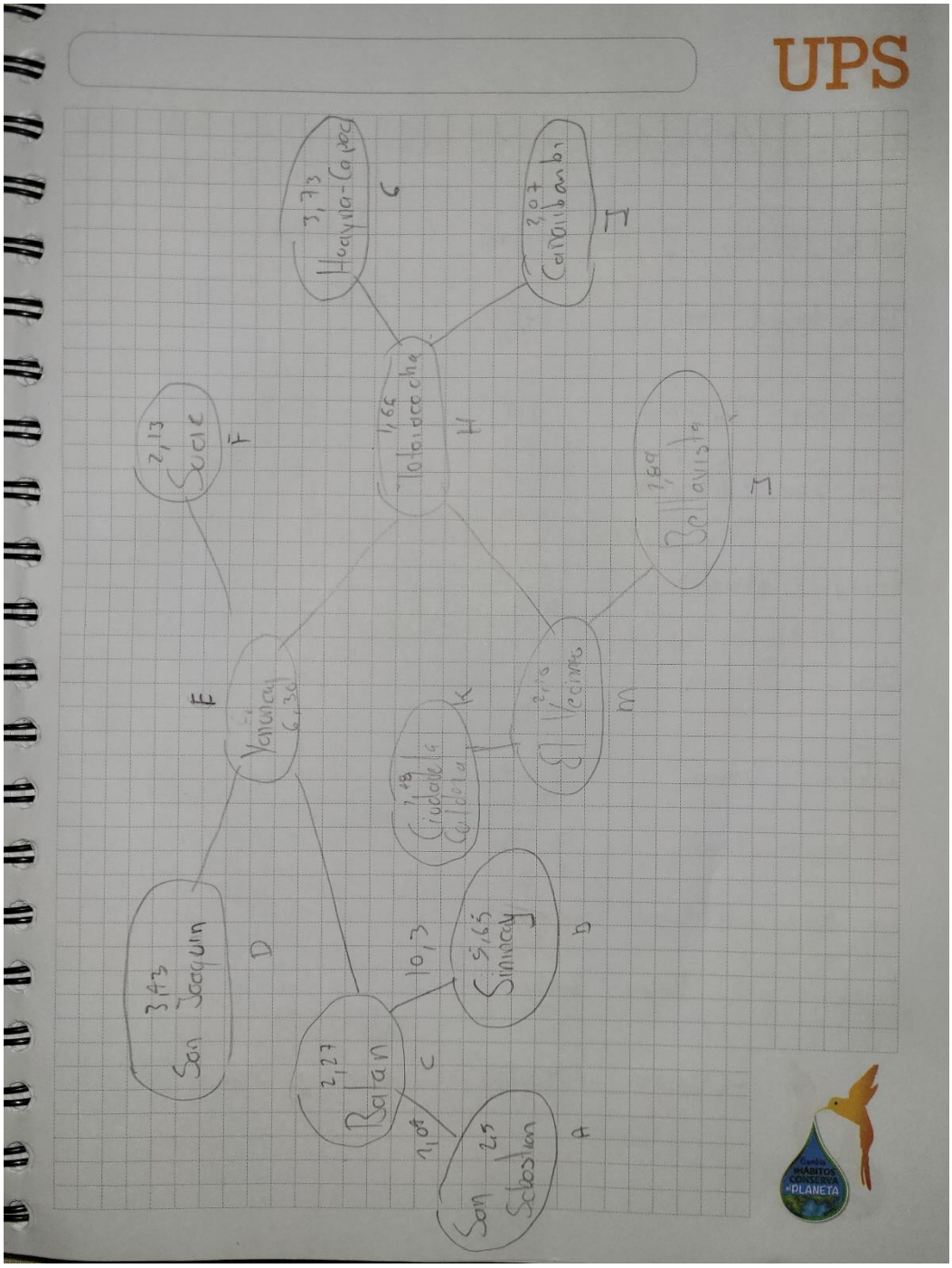
Definir 11 puntos de y armar el grafo.

11 puntos de interés



	Latitud	Longitud
El Vecino	-2,88121	-78,98798
San Joaquin	-2,89272	-79,02834
Yanuncay	-2,91577	-79,02834
El Balon	-2,89626	-79,03309
San Sebastian	-2,88892	-79,02435
Bellavista	-2,88097	-79,00256
Sucre	-2,90045	-79,01349
Huayna-Capac	-2,91446	-78,99479
Cañaribamba	-2,90571	-78,98941
Tobracocha	-2,89907	-78,97327
Ciudadela Carcelen	-2,93647	-78,96756
Sinincay	-2,89808	-79,01326

Grafo

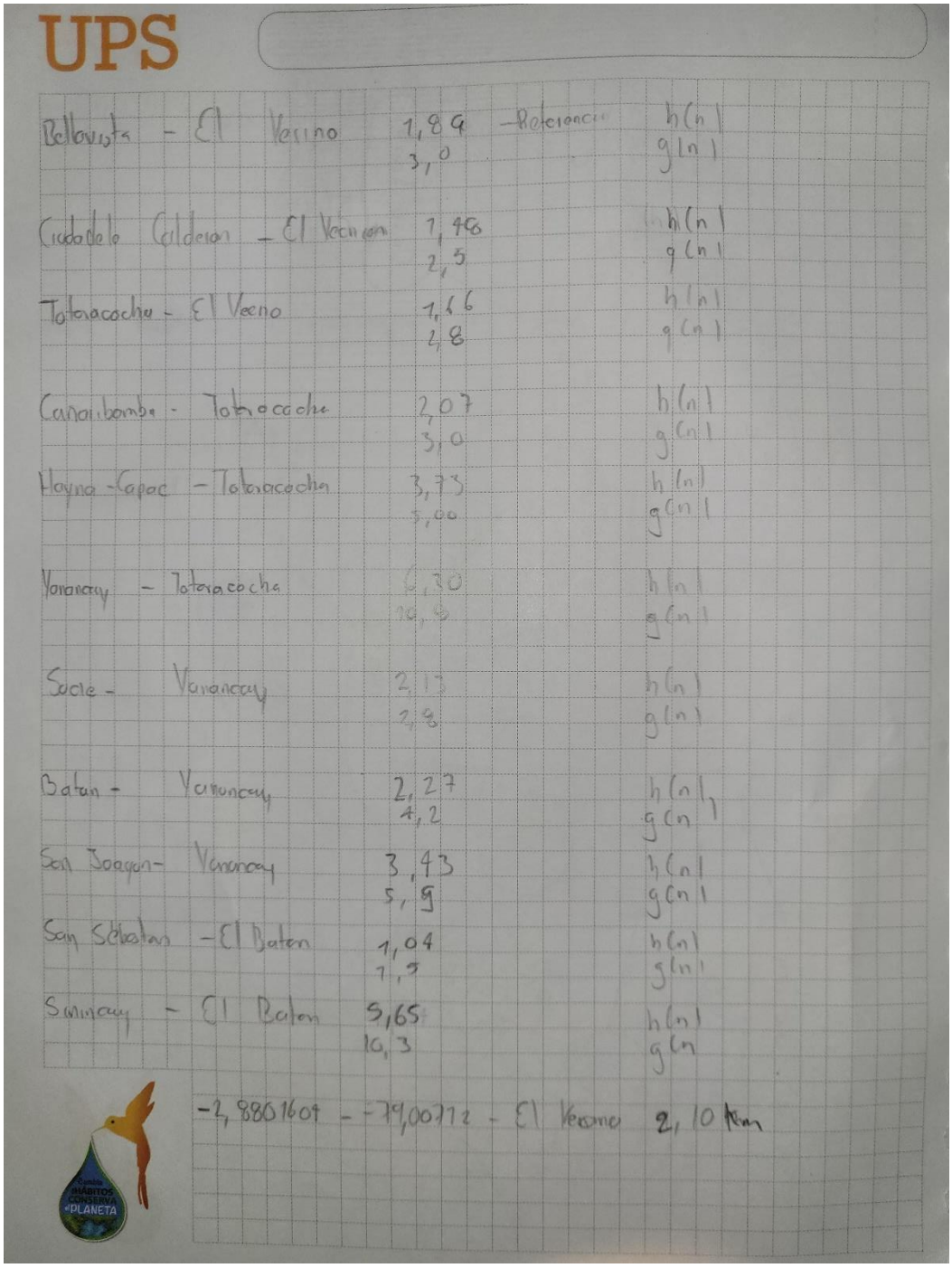


Especificar como punto de partida al sector "San Sebastián" y como objetivo "Totoracocha".

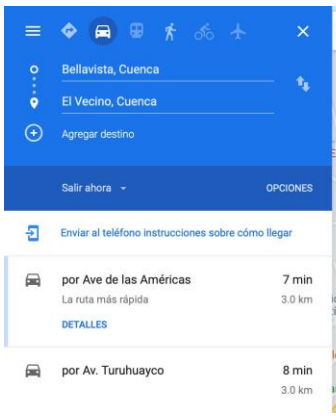
1) Punto de partida: San Sebastián"
2) Punto objetivo: Totoracocha

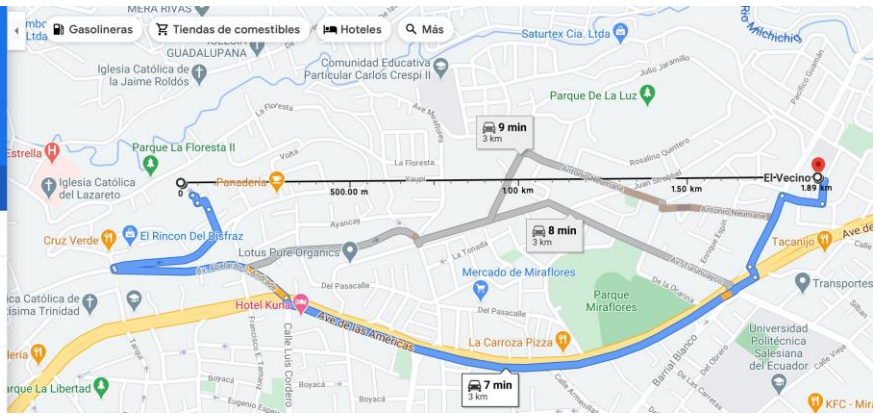
Estimar la distancia entre dos puntos datos usando la herramienta de regla que provee Google Maps y definirla como h(n).

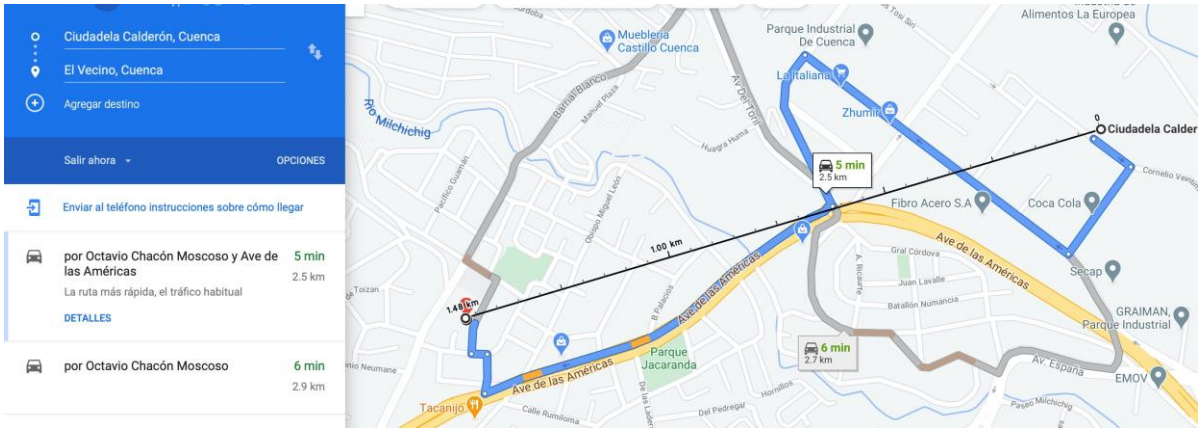
Calcular la distancia que existe entre los puntos de interés
Para ello debe usar la "ir de un punto a otro" de Google Maps (Direcciones o Indicaciones).


UPS
Bellavista - El Vecino 1,89
3,0
Ciudadela Calderon - El Vecino 1,48
2,5
Totoracocha - El Vecino 1,66
4,8
Canaribamba - Totoracocha 2,07
3,0
Huayna-Capac - Totoracocha 3,73
5,00
Varanday - Totoracocha
Socle - Varanday
Batan - Varanday 2,27
4,2
San Joaquin - Varanday 3,43
5,9
San Sebastian - El Batan 1,04
1,5
Sinincay - El Batan 5,65
10,3
-2,8801604 - -79,0077 - El Vecino 2,10 Km

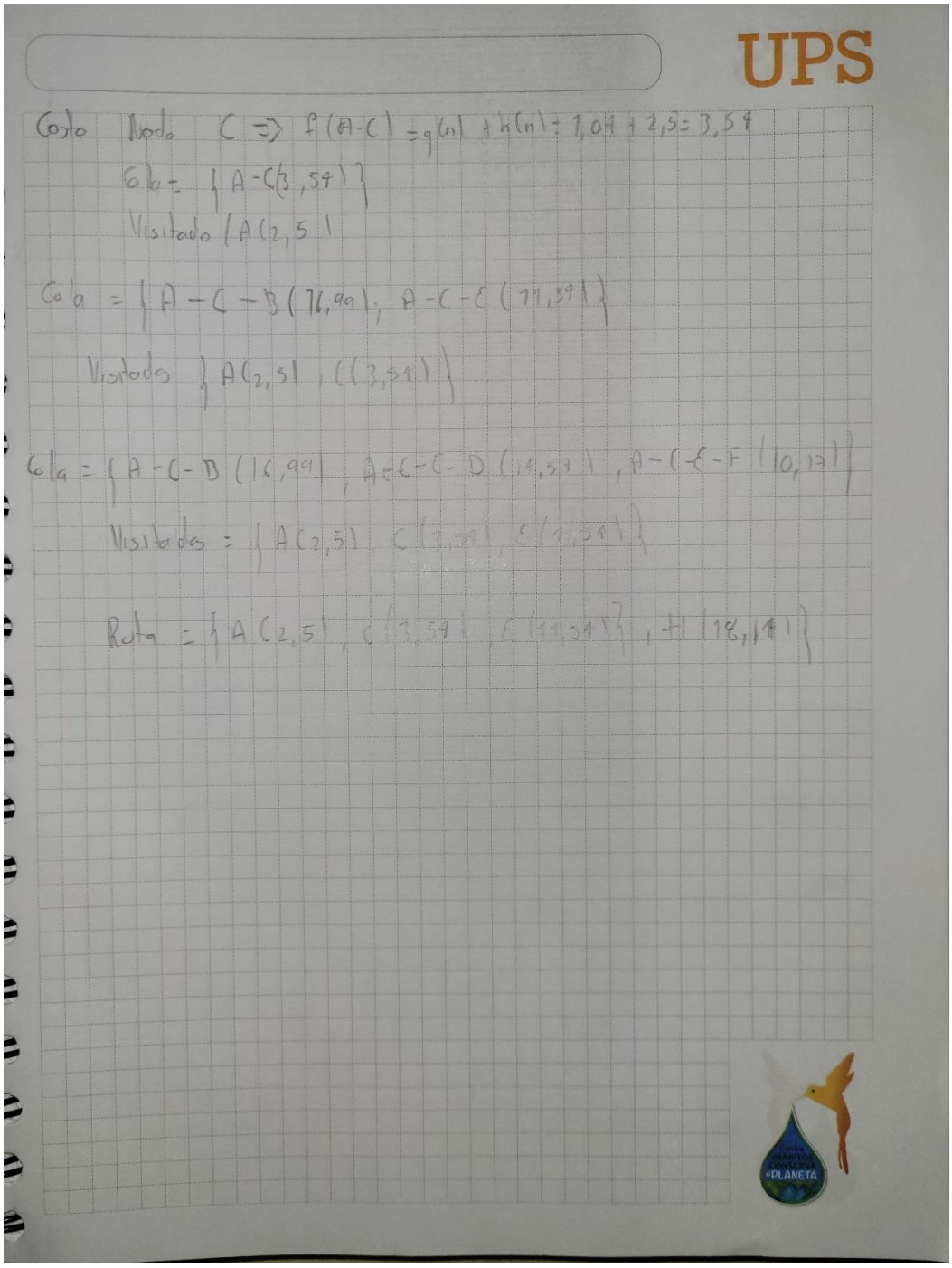
Ejemplos:


Bellavista, Cuenca
El Vecino, Cuenca
Agregar destino
Salir ahora
OPCIONES
Enviar al teléfono instrucciones sobre cómo llegar
por Ave de las Américas
La ruta más rápida
DETALLES
7 min
3.0 km
por Av. Turuhuayco
8 min
3.0 km


Gasolineras
Tiendas de comestibles
Hoteles
Más
Saturtex Cia. Ltda
GUADALUPANA
Iglesia Católica de la Jaime Roldós
Comunidad Educativa Particular Carlos Crespi II
Parque De La Luz
Parque La Floresta II
Iglesia Católica del Lazareto
Panadería
El-Vecino
9 min
3 km
8 min
3 km
7 min
3 km
Cruz Verde
El Rincon Del Disfraz
Lotus Pure Organics
Mercado de Miraflores
Parque Miraflores
Hotel Kuna
Ave de las Américas
La Carroza Pizza
Tacanjo
Transportes
Universidad Politécnica Salesiana del Ecuador
Barrial Blanco
Calle Luis Cordero
Río Machichio



Realizar el proceso de búsqueda de forma similar a cómo se a explicado en este apartado, almacenando para ello los datos de la lista Visitados y de la Cola.



Creacion de Nodos Lugares y con relaciones CONNECTION.

Importar la API py2neo

Para el ingreso de los datos que se encuentran dentro de la lista

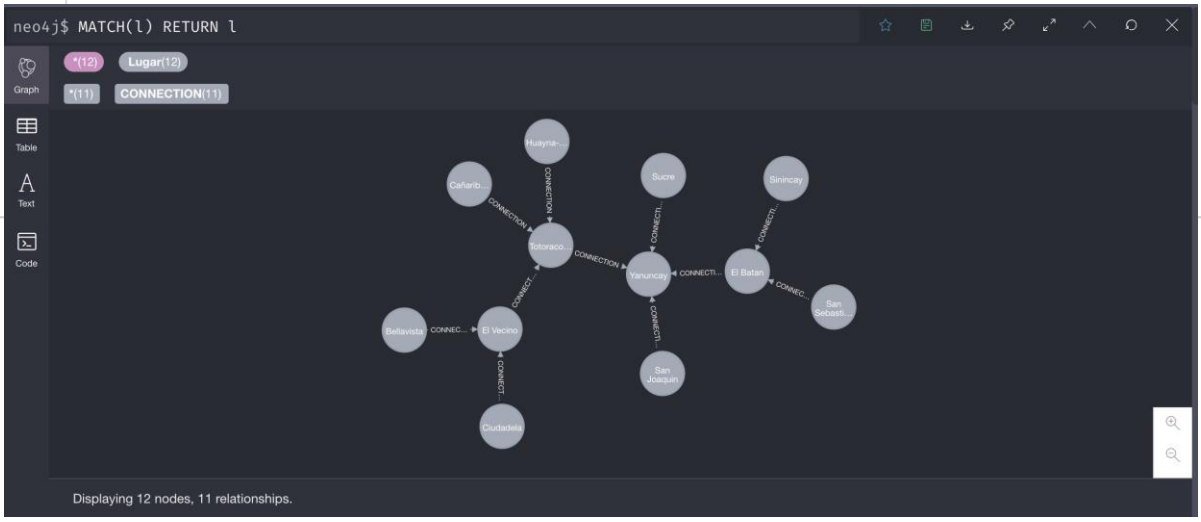
Conexión con Neo4j

creación de los 11 lugares con sus relaciones.

In [9]:
```
graph.run(" CREATE (a:Lugar {name: 'El Vecino', latitude: -2.881 "(b:Lugar {name: 'San Joaquin', latitude: -2.89372, longi "(c:Lugar {name: 'Yanuncay', latitude: -2.91577, longitud "(d:Lugar {name: 'El Batan',latitude: -2.89626, longitude "(e:Lugar {name: 'San Sebastian',latitude: -2.88892, long "(f:Lugar {name: 'Bellavista',latitude: -2.88047, longitu "(g:Lugar {name: 'Sucre',latitude: -2.90045, longitude: "(h:Lugar {name: 'Huayna-Capac',latitude: -2.91460, longi "(i:Lugar {name: 'Cañaribamba',latitude: -2.90512, longit "(j:Lugar {name: 'Totoracocha',latitude: -2.89002, longit "(k:Lugar {name: 'Ciudadela Calderon',latitude: -2.87642, "(m:Lugar {name: 'Sinincay',latitude: -2.84808, longitude "(e)-[:CONNECTION {g: 1.04}]->(d),"+ "(m)-[:CONNECTION {g: 10.3}]->(d),"+ "(d)-[:CONNECTION {g: 4.2}]->(c),"+ "(b)-[:CONNECTION {g: 5.9}]->(c),"+ "(g)-[:CONNECTION {g: 2.8}]->(c),"+ "(j)-[:CONNECTION {g: 10.8}]->(c),"+ "(h)-[:CONNECTION {g: 5.0}]->(j),"+ "(i)-[:CONNECTION {g: 3.0}]->(j),"+ "(a)-[:CONNECTION {g: 2.8}]->(j),"+ "(f)-[:CONNECTION {g: 3.0}]->(a),"+ "(k)-[:CONNECTION {g: 2.5}]->(a) ").data()
```

Out[9]: []

Consultar la creacion corecta de los nodos:



Lo siguiente ejecutará el algoritmo y transmitirá los resultados:

```
MATCH (start:Lugar {name: 'San Sebastian'}), (end:Lugar {name: 'Totoracocha'})
CALL gds.alpha.shortestPath.astar.stream({
  nodeProjection: {
    Lugar: {
      properties: ['longitude', 'latitude','h']
    }
  },
  relationshipProjection: {
    CONNECTION: {
      type: 'CONNECTION',
      orientation: 'UNDIRECTED',
      properties: 'g'
```

neo4j$ MATCH (start:Lugar {name: 'San Sebastian'}), (end:Lugar {name: 'Totoracocha'}) CAL…

Table | Text | Warn | Code

	lugares	cost
1	"San Sebastian"	0.0
2	"El Batan"	1.0
3	"Yanuncay"	2.0
4	"Totoracocha"	3.0

Started streaming 4 records after 1 ms and completed after 9 ms.

In [12]:
```
graph.run("MATCH (start:Lugar {name: 'San Sebastian'}), (end:Lug "CALL
gds.alpha.shortestPath.astar.stream({"+ "nodeProjection: {"+ "Lugar: {"+ "properties: ['longitude',
'latitude','h']"+ "}"+ "},"+ "relationshipProjection: {"+ "CONNECTION: {"+ "type: 'CONNECTION',"+
"orientation: 'UNDIRECTED',"+ "properties: 'g'"+ "}"+ "},"+ "startNode: start,"+ "endNode:
end,"+ "propertyKeyLat: 'h',"+ "propertyKeyLon: 'h'"+ "})"+ " YIELD nodeId, cost"+ " RETURN
gds.util.asNode(nodeId).name AS lugares, cost").data()
```

Out[12]:
```
[{'lugares': 'San Sebastian', 'cost': 0.0},
 {'lugares': 'El Batan', 'cost': 1.0},
 {'lugares': 'Yanuncay', 'cost': 2.0},
 {'lugares': 'Totoracocha', 'cost': 3.0}]
```

In [ ]:
```
1
```