

Ejercicio No 2: Búsqueda por profundidad

Nombre:

Fernando Sanchez

Enunciado:

- Diseñe un grafo similar al que se ha presentado en este ejercicio 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 (Link).

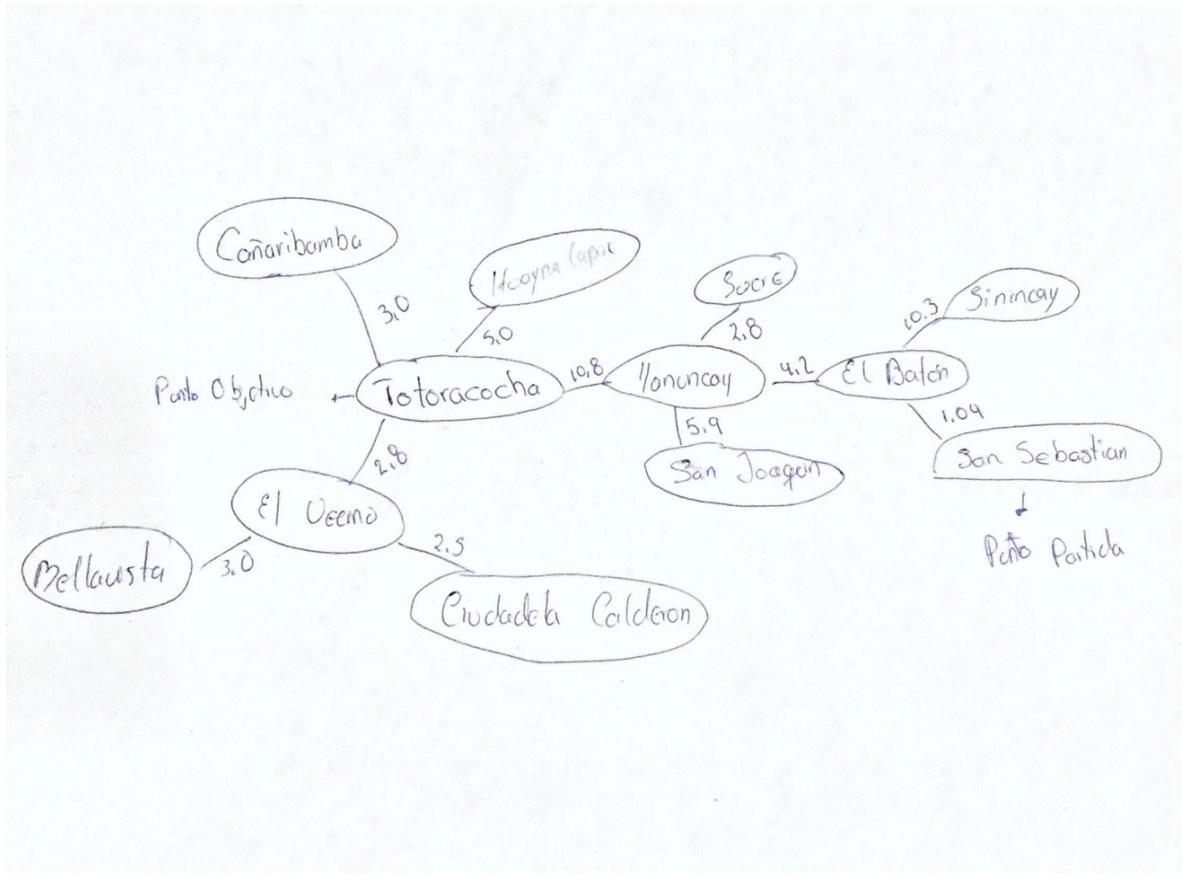
Definir 11 puntos de interés (El Vecino, Bellavista, Loja Argelia, Misicata, etc.) y armar el grafo.

11 Puntos de interes

UPS

	Latitud	Longitud
El vecino =	-2,88121	-78,98798
San Joaquín =	-2,89372	-79,05041
Yanocay =	-2,91577	-79,02834
El Batán =	-2,89626	-79,03309
San Sebastián =	-2,88892	-79,02435
Bella Vista =	-2,88047	-79,00256
Sucre =	-2,90045	-79,01349
Huayna-Capdc =	-2,91960	-78,99479
Cañabambas =	-2,90512	-78,98441
Totoracocha =	-2,89002	-78,97327
Ciudadela Calderón =	-2,87642	-78,96756
Sinincay =	-2,84808	-79,01326

Grafo



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

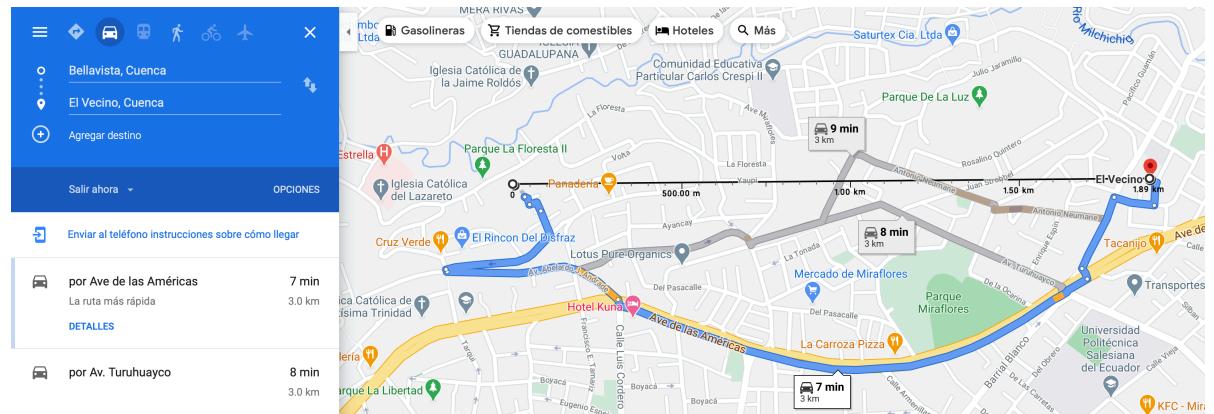
- 1) Punto de partida: San Sebastián"
- 2) Punto objetivo: Totoracocha

Establecer los arcos o caminos en 1 sola dirección, por ejemplo, del nodo "Bellavista" al nodo "Loja Argelia".

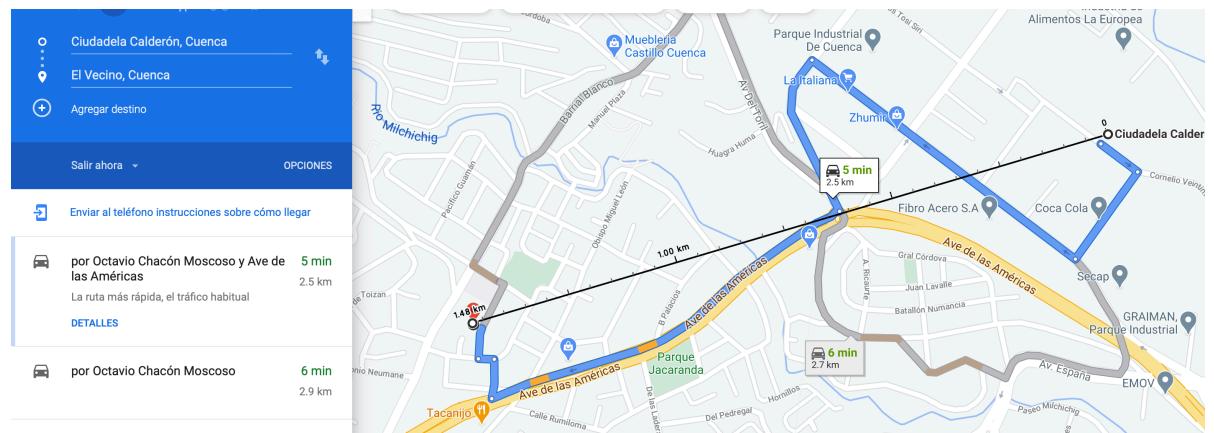
Calcular la distancia que existe entre los puntos de interés. Para ello puede usar la herramienta de medida (click con el botón derecho del ratón y seleccionar la opción "Medir").

Ejemplos:

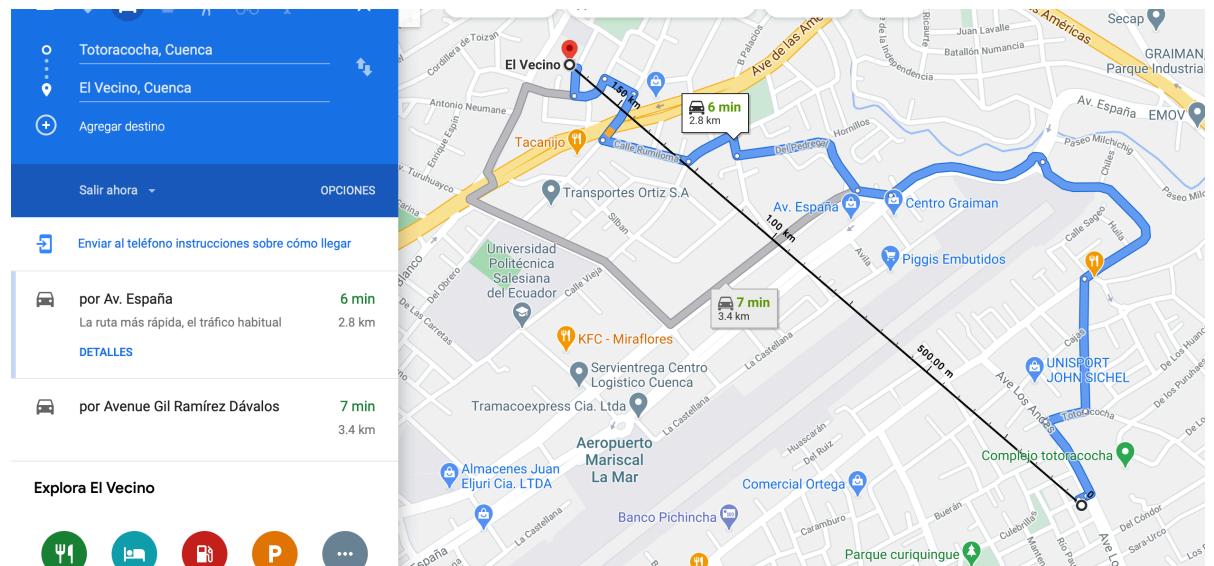
Bellavista – El vecino



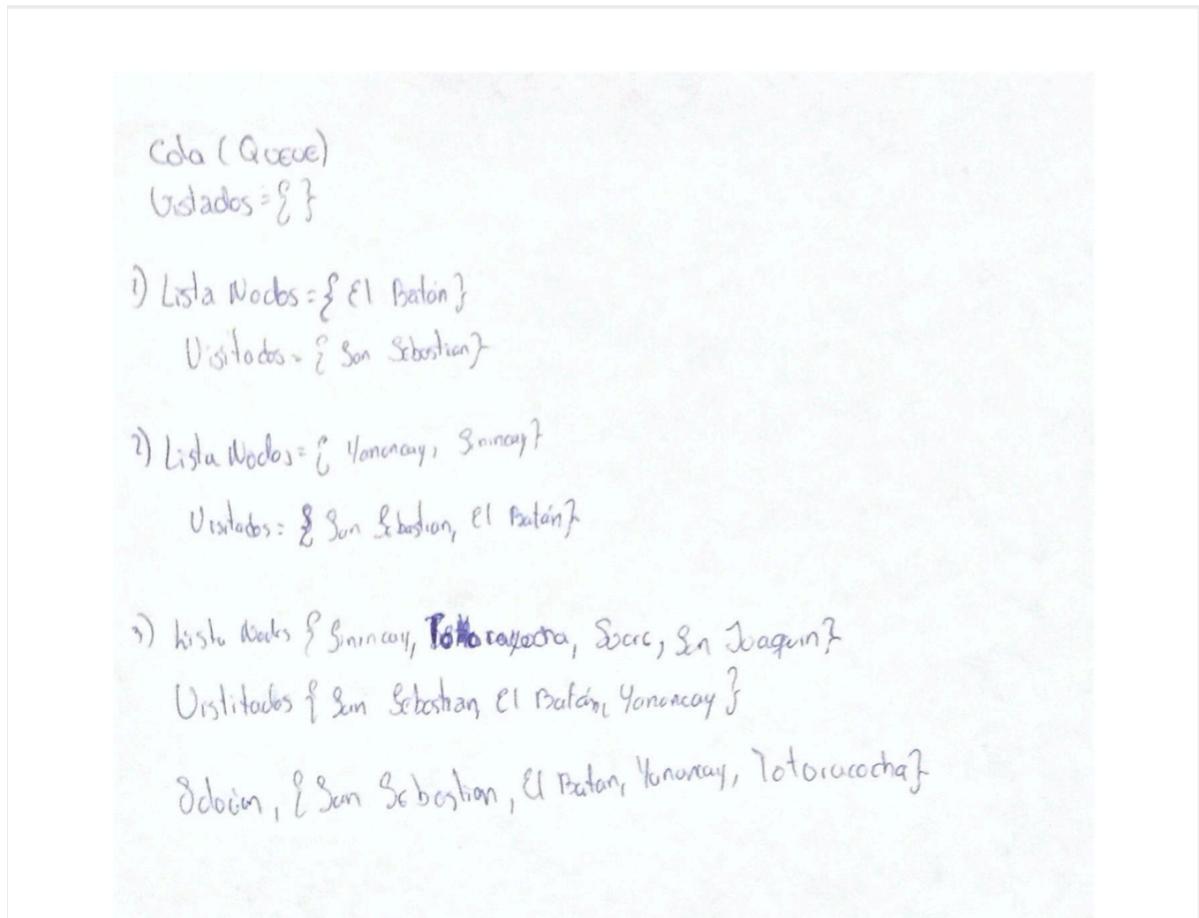
Ciudadela Calderon – El vecino



Totoracocha – El vecino



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



Creacion de Nodos Lugares y con relaciones REL.

Importar la API py2neo

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

Conexión con Neo4j

Configure la URL de conexión con la base de datos de Neo4j:

```
graph = Graph("bolt://localhost:7687", aut="neo4j",
password="profundidad", secure=False)
```

Creacion de los 11 lugares con sus relaciones.

In [1]:

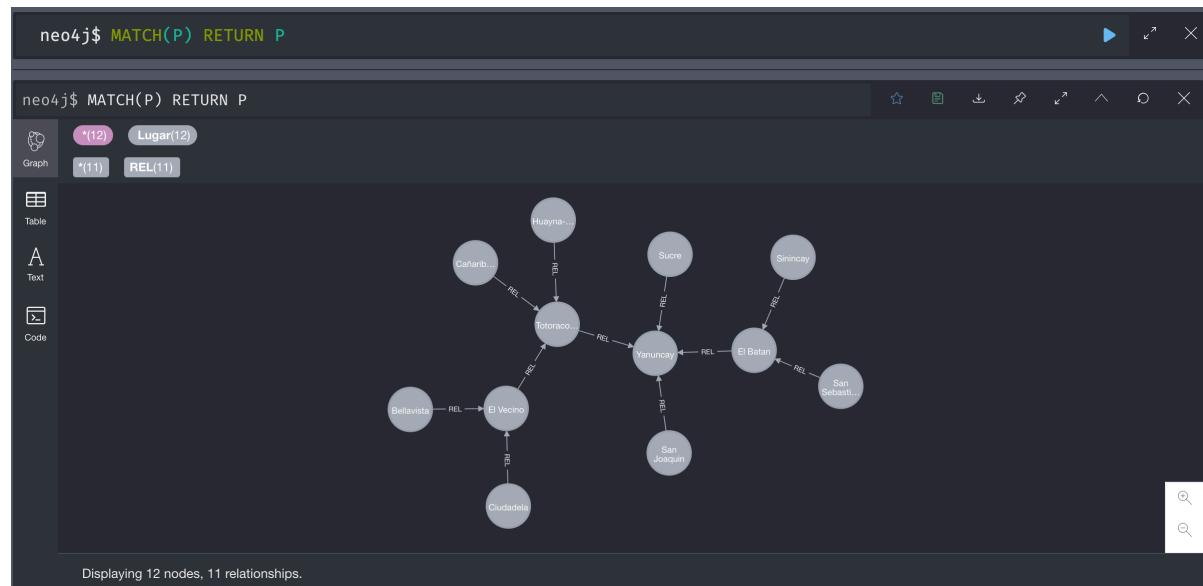
```
1 #IMPORTAR py2neo
2 from py2neo import Node, Relationship, Graph
3
4
5 # connect to authenticated graph database
6 graph = Graph("bolt://localhost:7687", aut="neo4j", password="p
```

In [2]:

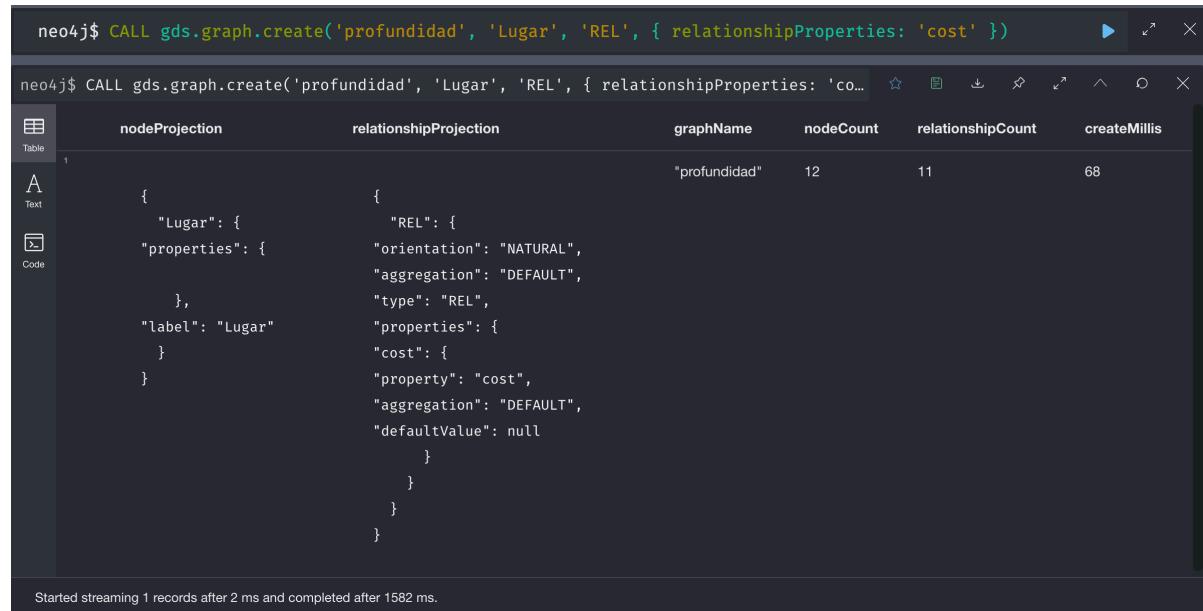
```
1 graph.run(" CREATE (a:Lugar {name: 'El Vecino', latitude: -2.88
2     "(b:Lugar {name: 'San Joaquin', latitude: -2.89372, long
3     "(c:Lugar {name: 'Yanuncay', latitude: -2.91577, longitu
4     "(d:Lugar {name: 'El Batan',latitude: -2.89626, longitud
5     "(e:Lugar {name: 'San Sebastian',latitude: -2.88892, lon
6     "(f:Lugar {name: 'Bellavista',latitude: -2.88047, longit
7     "(g:Lugar {name: 'Sucre',latitude: -2.90045, longitude:
8     "(h:Lugar {name: 'Huayna-Capac',latitude: -2.91460, long
9     "(i:Lugar {name: 'Cañaribamba',latitude: -2.90512, longi
10    "(j:Lugar {name: 'Totoracocha',latitude: -2.89002, longi
11    "(k:Lugar {name: 'Ciudadela Calderon',latitude: -2.87642
12    "(m:Lugar {name: 'Sinincay',latitude: -2.84808, longitud
13    "(e)-[:REL {cost: 1.04}]->(d),"+
14    "(m)-[:REL {cost: 10.3}]->(d),"+
15    "(d)-[:REL {cost: 4.2}]->(c),"+
16    "(b)-[:REL {cost: 5.9}]->(c),"+
17    "(g)-[:REL {cost: 2.8}]->(c),"+
18    "(j)-[:REL {cost: 10.8}]->(c),"+
19    "(h)-[:REL {cost: 5.0}]->(j),"+
20    "(i)-[:REL {cost: 3.0}]->(j),"+
21    "(a)-[:REL {cost: 2.8}]->(j),"+
22    "(f)-[:REL {cost: 3.0}]->(a),"+
23    "(k)-[:REL {cost: 2.5}]->(a) ").data()
24
25
26
27
28
```

Out[2]: []

Consultar la creacion correcta de los nodos:



Crear el gráfico el cual almacenará un catálogo de gráficos



Lo siguiente ejecutará el algoritmo y transmitirá los resultados:

```
⚠ 1 MATCH (a:Lugar{name:'San Sebastian'}), (d:Lugar{name:'Totoracocha'})  
2 WITH id(a) AS startNode, [id(d)] AS targetNodes  
3 CALL gds.alpha.dfs.stream('profundidad', {startNode: startNode, targetNodes: targetNodes})  
4 YIELD path  
5 UNWIND [ n in nodes(path) | n.name ] AS nombre  
6 RETURN nombre
```

neo4j\$ MATCH (a:Lugar{name:'San Sebastian'}), (d:Lugar{name:'Totoracocha'}) WITH id(a) AS...

Started streaming 3 records after 1 ms and completed after 34 ms.

In [3]:

```
1 graph.run("MATCH (a:Lugar{name:'San Sebastian'}), (d:Lugar{name:  
2 "WITH id(a) AS startNode, [id(d)] AS targetNodes"+  
3 "CALL gds.alpha.dfs.stream('profundidad', {startNode  
4 "YIELD path"+  
5 "UNWIND [ n in nodes(path) | n.name ] AS nombre"+  
6 "RETURN nombre"}).data()
```

Out[3]:

```
[{'nombre': 'San Sebastian'}, {'nombre': 'El Batan'}, {'nombre': 'Yanuncay'}]
```

In []:

1	
---	--