



Ejercicio No 1: Búsqueda por anchura

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.
- Especificar como punto de partida al sector "San Sebastián" y como objetivo "Totoracocha".
- Establecer los arcos o caminos en 1 sola dirección, por ejemplo, del nodo "Bellavista" al nodo "Loja Argelia".
- 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**.

El trabajo deberá desarrollarse de forma manual en el cuaderno.

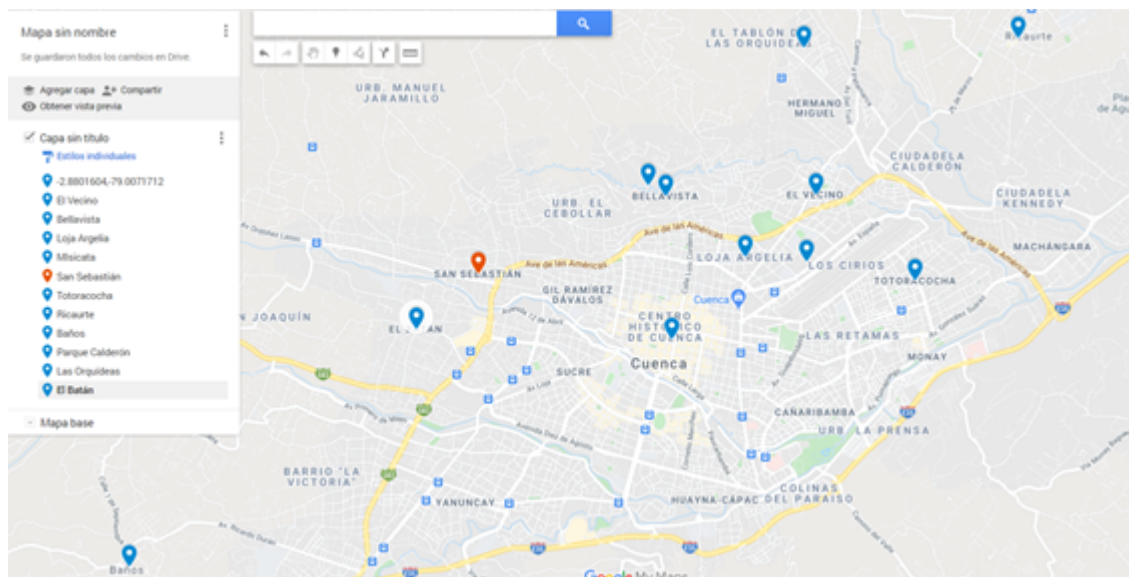


Ejercicio No 2: Búsqueda por profundidad

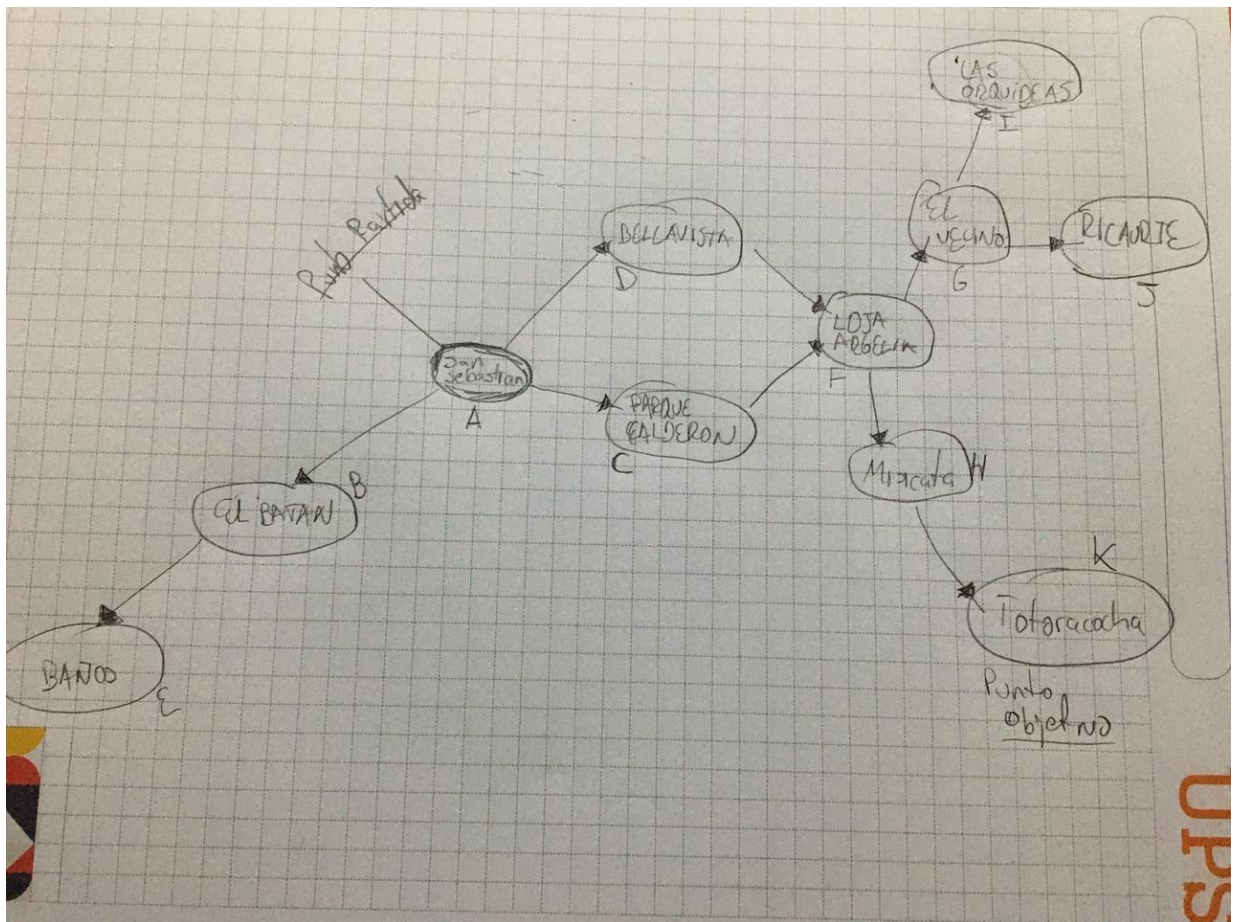
Diseñe un grafo similar al que se ha presentado en este ejercicio partiendo de las siguientes coordenadas de latitud y longitud: **-2.8801604,-79.0071712**. Para ello deberá realizar las siguientes tareas:

- 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.
- Especificar como punto de partida al sector "San Sebastián" y como 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").
- 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**.

El trabajo deberá desarrollarse de forma manual en el cuaderno.



	Nombre	Descripción
1	-2.8801604,-79.0071712	PUNTO DE REFERENCIA
2	El Vecino	
3	Bellavista	
4	Loja Argelia	
5	Misicata	
6	San Sebastián	PUNTO DE PARTIDA
7	Totoracocha	PUNTO OBJETIVO
8	Ricaurte	
9	Baños	
10	Parque Calderón	
11	Las Orquídeas	
12	El Batán	



Origen	Destino	4(n)	6(n)
A	B	1,03 Km	2,5 Km
A	C	2,57 Km	2,9 Km
A	D	2,56 Km	3,9 Km
B	E	4,68 Km	7,9 Km
C	F	1,38 Km	1,9 Km
D	F	1,26 Km	2,4 Km
F	G	1,18 Km	1,5 Km
F	H	0,76985 Km	1,9 Km
G	I	1,85 Km	2,7 Km
G	J	3,22 Km	4,7 Km
H	K	1,39 Km	2,8 Km
,			

```
In [1]: from neo4j import GraphDatabase
```

```

In [2]: #CLASE PAR CREAR NODO CENTRAR-PARQUE CENTRAL
class CLASE_NEO4J(object):
    def __init__(self):
        self._driver = GraphDatabase.driver("bolt:neo4j://localhost:7687", auth=
    def close(self):
        self._driver.close()
    def CREAR_LUGAR(self, message,lugar, latitud, longitud):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_LUGAR, message,lug
            print(greeting)
    def CREAR_RUTA(self,origen,destino,costo,hn):
        with self._driver.session() as session:
            greeting2 = session.write_transaction(self._VALIDAR_RUTA,origen,desti
            print(greeting2)
    #METODO PARA CREAR LOS NODOS DE LUGARES
    @staticmethod
    def _VALIDAR_LUGAR(tx, message,lugar, latitud, longitud):
        #SE BUSCA SI EL LUGAR DEL ARREGLO EXISTE EN LA BASE NEO4J
        result2 = tx.run("match(l:Lugares {nombre:'"+lugar+"'}) return l.nombre")
        #CONDICION PARA VERIFICAR SI EXISTE
        if int(len(result2)) == 0:
            print("SE CREA EL LUGAR EN LA BASE.....")
            #SE CREA NODO LUGAR
            result = tx.run("CREATE("+lugar+":Lugares {nombre:'"+lugar+"',latitu
                "SET "+lugar+".message = $message "
                "RETURN "+lugar+".message + ', from node ' + id("+lugar+"
            elif int(len(result2)) == 1:
                print("EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR.....")
    #METODO PARA CREAR LAS RELACIONES CON EL COSTE Y HN PARA LA RUTA
    @staticmethod
    def _VALIDAR_RUTA(tx,origen,destino,costo,hn):
        #SE BUSCA SI LA RUTA A CREAR YA DEL ARREGLO EXISTE EN LA BASE NEO4J
        result = tx.run("match(l1:Lugares{nombre:'"+origen+"'})-[r:RUTA_DE{costo:
        if int(len(result)) == 0:
            print("SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
            result2 = tx.run(" match("+origen+":Lugares {nombre:'"+origen+"'}) ma
        elif int(len(result)) == 1:
            print("YA EXISTE LA RUTA*****")
    # MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
    #SE INICIALIZA LA CLASE DE LOS METODOS DE NEO4J
    grafo=CLASE_NEO4J()

```

```

In [3]: #SE CREA LA LISTA DE NODOS LUGARES
listal = ([ "El_Vecino", -2.88112, -78.9882],
           [ "Bellavista", -2.88129, -79.00516],
           [ "Loja_Angelia", -2.88817, -78.99612],
           [ "Misicata", -2.88866, -78.98923],
           [ "San_Sebastian", -2.89008, -79.02636],
           [ "Totoracocha", -2.89082, -78.97689],
           [ "Ricaurte", -2.86347, -78.96523],
           [ "Baños", -2.92317, -79.06591],
           [ "Parque_Calderon", -2.89741, -79.00448],
           [ "Las_Orquideas", -2.86452, -78.98954],
           [ "El_Batan", -2.89628, -79.03342])

cont = 0
for ll in listal:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    grafo.CREAR_LUGAR("SE GENERA UN NODO LUGAR EN LA BASE >>>>>>>>> ",str(ll[0]),
    cont+=1
    print(cont)

```

SE CREA EL LUGAR EN LA BASE.....

None

1

SE CREA EL LUGAR EN LA BASE.....

None

2

SE CREA EL LUGAR EN LA BASE.....

None

3

SE CREA EL LUGAR EN LA BASE.....

None

4

SE CREA EL LUGAR EN LA BASE.....

None

5

SE CREA EL LUGAR EN LA BASE.....

None

6

SE CREA EL LUGAR EN LA BASE.....

None

7

SE CREA EL LUGAR EN LA BASE.....

None

8

SE CREA EL LUGAR EN LA BASE.....

None

9

SE CREA EL LUGAR EN LA BASE.....

None

10

SE CREA EL LUGAR EN LA BASE.....

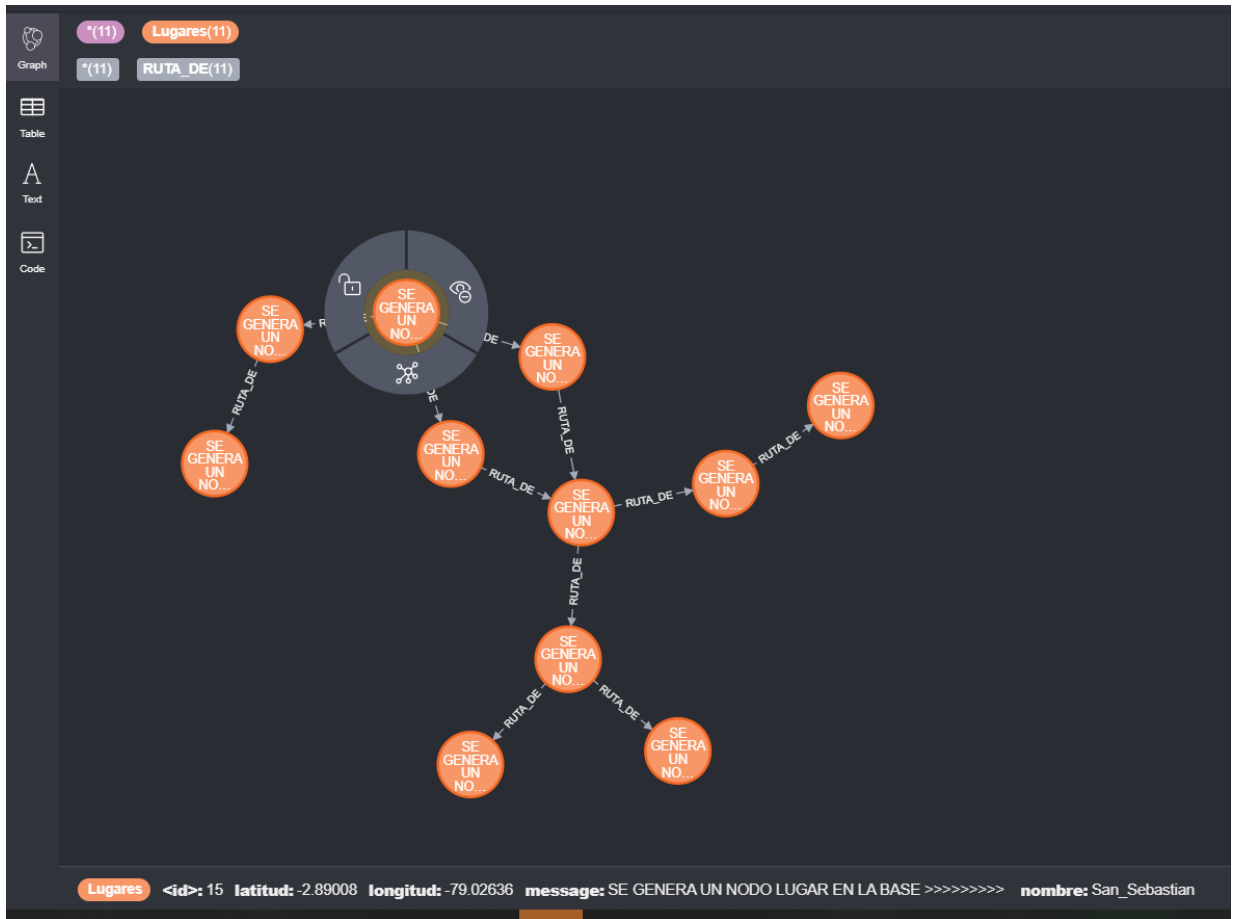
None

11

```
In [4]: # SE GENERA LAS RELACIONES DE LOS LUGARES
#SE CREA LA LISTA DE LAS RELACIONES Y LOS NODOS
listaL = ([ "San_Sebastian", "El_Batan", 1.03, 2.5],
           [ "San_Sebastian", "Parque_Calderon", 2.57, 2.9],
           [ "San_Sebastian", "Bellavista", 2.56, 3.9],
           [ "El_Batan", "Baños", 4.68, 7.9],
           [ "Parque_Calderon", "Loja_Angelia", 1.38, 1.9],
           [ "Bellavista", "Loja_Angelia", 1.26, 2.4],
           [ "Loja_Angelia", "El_Vecino", 1.18, 1.5],
           [ "Loja_Angelia", "Misicata", 0.76985, 1.9],
           [ "El_Vecino", "Las_Orquideas", 1.85, 2.7],
           [ "El_Vecino", "Ricaurte", 3.22, 4.7],
           [ "Misicata", "Totoracocha", 1.39, 2.8])

for ll in listaL:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    grafo.CREAR_RUTA(str(ll[0]),str(ll[1]),str(ll[2]),str(ll[3]))
```

```
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
None
```



Búsqueda por anchura

```
CALL gds.graph.create('myGraphBFS', 'Lugares', 'RUTA_DE', {
  relationshipProperties: 'costo' })
```

	nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	createMillis
Table						
Text						
Code						
	{ "Lugares": { "properties": { }, "label": "Lugares" } }	{ "RUTA_DE": { "orientation": "NATURAL", "aggregation": "DEFAULT", "type": "RUTA_DE", "properties": { "costo": { "property": "costo", "aggregation": "DEFAULT", "defaultValue": null } } } }	"myGraphBFS"	11	11	274

Started streaming 1 records after 1 ms and completed after 279 ms.

```
MATCH (San_Sebastian:Lugares{nombre:'San_Sebastian'}),
      (Totoracocha:Lugares{nombre:'Totoracocha'})
WITH id(San_Sebastian) AS startNode, [id(Totoracocha)] AS targetNodes
CALL gds.alpha.bfs.stream('myGraphBFS', {startNode: startNode, targetNodes:
targetNodes})
YTFID nath
```

```
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

<div><div>Table</div><div>Text</div><div>Warn</div><div>Code</div></div>	<div><div>tags</div><div>1"San_Sebastian"</div><div>2"Bellavista"</div><div>3"Parque_Calderon"</div><div>4"El_Batan"</div><div>5"Loja_Argelia"</div><div>6"Baños"</div><div>7"El_Vecino"</div><div>8"Misicata"</div><div>9"Ricaurte"</div><div>10"Las_Orquideas"</div><div>11"Totoracocha"</div></div> <div>Started streaming 11 records in less than 1 ms and completed after 15 ms.</div>
--	---


Búsqueda por profundidad


```
CALL gds.graph.create('myGraphDFS', 'Lugares', 'RUTA_DE', {
relationshipProperties: 'costo' })
```


	nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	createMillis
Table	1					
Text						
Code						
	<pre>{ "Lugares": { "properties": { }, "label": "Lugares" } }</pre>	<pre>{ "RUTA_DE": { "orientation": "NATURAL", "aggregation": "DEFAULT", "type": "RUTA_DE", "properties": { "costo": { "property": "costo", "aggregation": "DEFAULT", "defaultValue": null } } } }</pre>	"myGraphDFS"	11	11	16


Started streaming 1 records in less than 1 ms and completed after 19 ms.

```
MATCH (San_Sebastian:Lugares{nombre:'San_Sebastian'}),
(Totoracocha:Lugares{nombre:'Totoracocha'})
WITH id(San_Sebastian) AS startNode, [id(Totoracocha)] AS targetNodes
CALL gds.alpha.dfs.stream('myGraphDFS', {startNode: startNode, targetNodes:
targetNodes})
YIELD path
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

Table

Text

Warn

Code

tags

1	"San_Sebastian"
2	"El_Batan"
3	"Baños"
4	"Parque_Calderon"
5	"Loja_Argelia"
6	"Misicata"
7	"Totoracocha"

Started streaming 7 records after 1 ms and completed after 4 ms.

In []:

