

Ejercicio No 4: Algoritmo A*

Nombre:

Fernando Sanchez

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 (Link).

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="l
```

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 = $-2,89372, -79,05041$

Yanuncay = $-2,91577, -79,02834$

El Batán = $-2,89626, -79,03309$

San Sebastián = $-2,88892, -79,02435$

Bella Vista = $-2,88047, -79,00256$

Sucre = $-2,90045, -79,01349$

Hoayna-Capdc = $-2,91960, -78,99479$

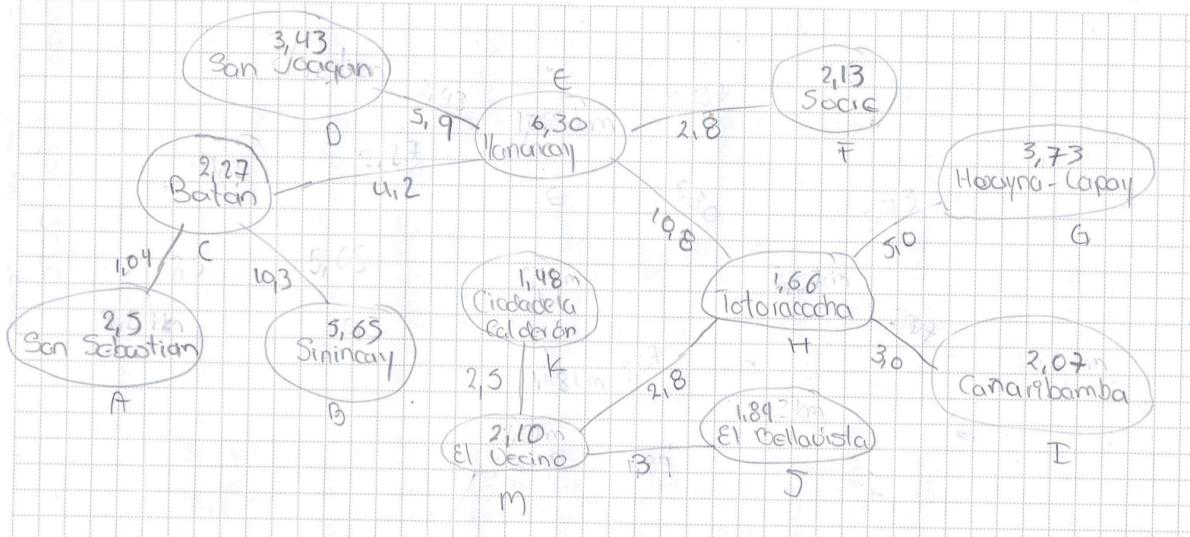
Cañaribamba = $-2,90512, -78,98441$

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

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).

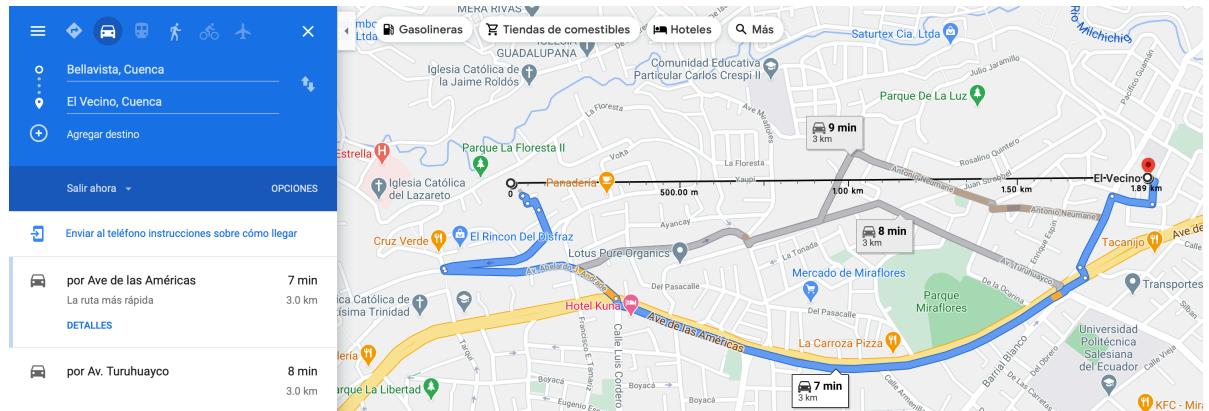
Nombre: Fernando Jardines
 Materia: IA
 Carrera: Ing de Sistemas
 Fecha: 14/02/2020

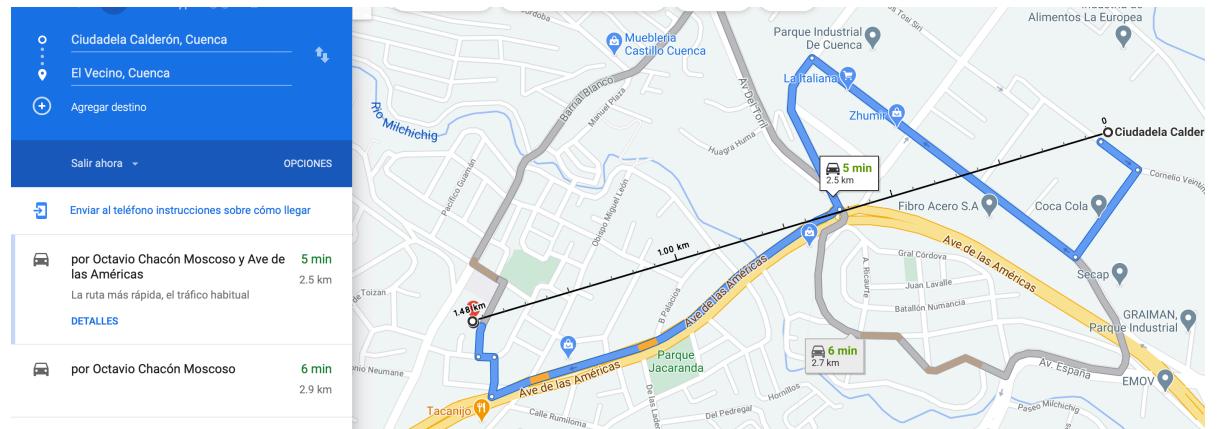
Calificación:

UPS

Bellavista - El Vecino	1,89 km → Referencia 3,0 km	$h(n)$ $g(n)$
Ciudadela Colcazán - El Vecino	1,48 km → Referencia 2,5 km	$h(n)$ $g(n)$
Totoracocha - El Vecino	1,66 km → Referencia 2,8 km	$h(n)$ $g(n)$
Cañaribamba - Totoracocha	2,07 km → Referencia 3,0 km	$h(n)$ $g(n)$
Hoayna - Cápac - Totoracocha	3,73 km → Referencia 5,00 km	$h(n)$ $g(n)$
Vilcabamba - Totoracocha	6,30 km → Referencia 19,8 km	$h(n)$ $g(n)$
Yanocay - Totoracocha	2,13 km → Referencia 2,8 km	$h(n)$ $g(n)$
Socie - Yanocay	2,27 km → Referencia 4,7 km	$h(n)$ $g(n)$
El Batán - Yanocay	3,43 km → Referencia 5,9 km	$h(n)$ $g(n)$
San Joaquín - Yanocay	1,04 km → Referencia 2,5 km	$h(n)$ $g(n)$
Smimcay - El Batán	5,65 km → Referencia 19,3 km	$h(n)$ $g(n)$
7,8801609, -79,007172 El Vecino	9,70 km	$g(n)$

Ejemplos:





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.

$$\text{Coste del paso por el nodo 'C'} \Rightarrow f(A-C) = g(n) + h(n) = 7,04 + 2,5 = 9,54$$

Cola = { A → C (9,54) }

Visitados = { A (7,04), C (9,54) }

i) Cola = { A → C → B (16,99), A → C → E (11,54) }
 Visitados = { A (7,04), C (9,54) }

3) Cola = { A → C → B (16,99), A → C → E → D (14,57), A → C → E → F (19,17), A → C → E → H (18,11) }
 Visitados = { A (7,04), C (9,54), E (11,54) }

Vistados = { A (7,04), C (9,54), E (11,54), H (18,11) }
 Ruta = { A (7,04), C (9,54), E (11,54), H (18,11) }

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

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

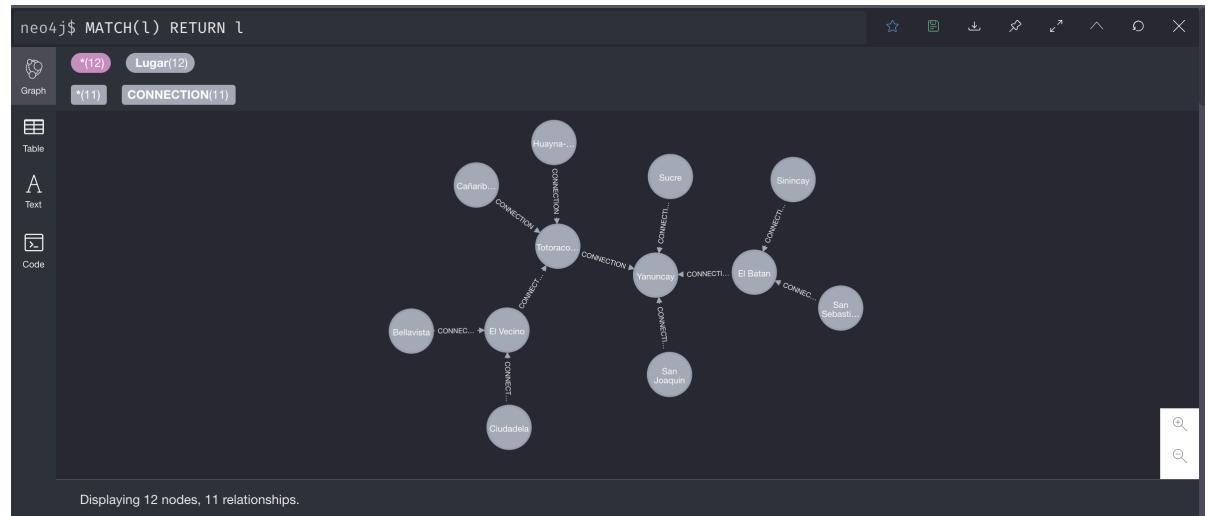
```
graph = Graph("bolt://localhost:7687", aut="neo4j",
password="lugares", secure=False)
```

Creacion de los 11 lugares con sus relaciones.

```
In [9]: 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)-[:CONNECTION {g: 1.04}]->(d), "+"
14          "(m)-[:CONNECTION {g: 10.3}]->(d), "+"
15          "(d)-[:CONNECTION {g: 4.2}]->(c), "+"
16          "(b)-[:CONNECTION {g: 5.9}]->(c), "+"
17          "(g)-[:CONNECTION {g: 2.8}]->(c), "+"
18          "(j)-[:CONNECTION {g: 10.8}]->(c), "+"
19          "(h)-[:CONNECTION {g: 5.0}]->(j), "+"
20          "(i)-[:CONNECTION {g: 3.0}]->(j), "+"
21          "(a)-[:CONNECTION {g: 2.8}]->(j), "+"
22          "(f)-[:CONNECTION {g: 3.0}]->(a), "+"
23          "(k)-[:CONNECTION {g: 2.5}]->(a) ").data()
24
25
26
27
28
```

Out [9]: []

Consultar la creacion correcta de los nodos:



Lo siguiente ejecutará el algoritmo y transmitirá los resultados:

```

1 MATCH (start:Lugar {name: 'San Sebastian'}), (end:Lugar {name: 'Totoracocha'})
2 CALL gds.alpha.shortestPath.astar.stream({
3   nodeProjection: {
4     Lugar: {
5       properties: ['longitude', 'latitude','h']
6     }
7   },
8   relationshipProjection: {
9     CONNECTION: {
10       type: 'CONNECTION',
11       orientation: 'UNDIRECTED',
12       properties: 'g'
13     }
14   }
15 })
16 
```

neo4j\$ 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' } } })

	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]: 1 graph.run("MATCH (start:Lugar {name: 'San Sebastian'}), (end:Lugar)
2 "CALL gds.alpha.shortestPath.astar.stream({"+
3   "nodeProjection: {"+
4     "Lugar: {"+
5       "properties: ['longitude', 'latitude', 'h']"+

6     "}" +
7   }, "+

8   "relationshipProjection: {"+
9     "CONNECTION: {"+
10      "type: 'CONNECTION', "+

11      "orientation: 'UNDIRECTED', "+

12      "properties: 'g'"+

13    "}" +
14  }, "+

15  "startNode: start," +
16  "endNode: end," +
17  "propertyKeyLat: 'h'," +
18  "propertyKeyLon: 'h'" +
19 })" +
20 " YIELD nodeId, cost" +
21 " 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
```