- Recorrido BFS desde Sevilla

```
MATCH (Se:Ciudad{nombre:'Sevilla'})
WITH id(Se) AS inicio
CALL gds.bfs.stream('AnchuraSevilla', {sourceNode: inicio})
YIELD path
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

	tags
1	"Sevilla"
2	"Cádiz"
3	"Jaén"
4	"Granada"
5	"Madrid"
6	"Murcia"

Started streaming 17 records after 10 ms and completed after 19 ms.

```
neo4j$ CALL gds.graph.project('AnchuraSevilla','Ciudad','distancia',{relationshipProperties:'km'})
```

	nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
1	{ "Ciudad": {	{ "distancia": {	"AnchuraSevilla"	17	46	6450

- Recorrido DFS desde Sevilla

```
MATCH (Se:Ciudad{nombre:'Sevilla'})
WITH id(Se) AS inicio
CALL gds.bfs.stream('ProfundidadSevilla', {sourceNode: inicio})
YIELD path
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

	tags
1	"Sevilla"
2	"Cádiz"
3	"Jaén"
4	"Granada"
5	"Madrid"
6	"Murcia"

Started streaming 17 records after 3 ms and completed after 12 ms.

```
neo4j$ CALL gds.graph.project('ProfundidadSevilla','Ciudad','distancia',{relationshipProperties:'km'})
```

	nodeProjection	relationshipProjection	graphName	nodeCount	rela
1	{	{	"ProfundidadSevilla"	17	46

- Recorrido BFS especificando nodos destino Coruña

```
MATCH (Se:Ciudad{nombre:'Sevilla'}),(Co:Ciudad{nombre:'Coruña'})
WITH id(Se)  AS inicio , id(Co) AS fin
CALL gds.bfs.stream('SevillaCoruña', {sourceNode:inicio ,targetNodes:fin })
YIELD path
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

	tags
1	"Sevilla"
2	"Cádiz"
3	"Jaén"
4	"Granada"
5	"Madrid"
6	"Murcia"
7	

Started streaming 15 records after 11 ms and completed after 25 ms.

```
neo4j$ CALL gds.graph.project('SevillaCoruña','Ciudad','distancia',{relationshipProperties:'km'})
```

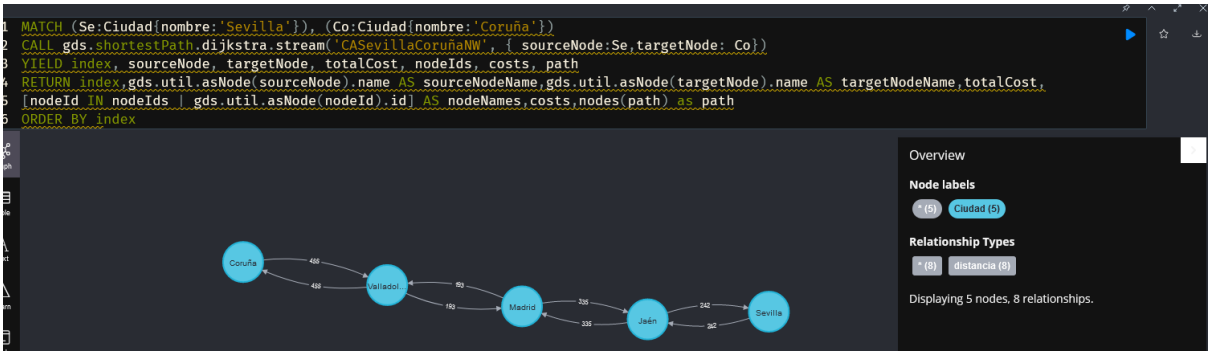
- Recorrido DFS especificando nodos destino Coruña

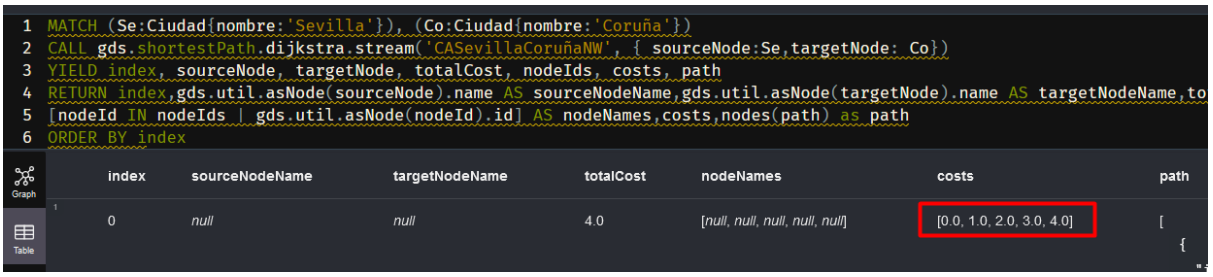
```
MATCH (Se:Ciudad{nombre:'Sevilla'}),(Co:Ciudad{nombre:'Coruña'})
WITH id(Se)  AS inicio , id(Co) AS fin
CALL gds.dfs.stream('SevillaCoruña', {sourceNode:inicio ,targetNodes:fin })
YIELD path
UNWIND [ n in nodes(path) | n.nombre ] AS tags
RETURN tags
```

	tags
1	"Sevilla"
2	"Granada"
3	"Murcia"
4	"Valencia"
5	"Barcelona"
6	"Zaragoza"
7	

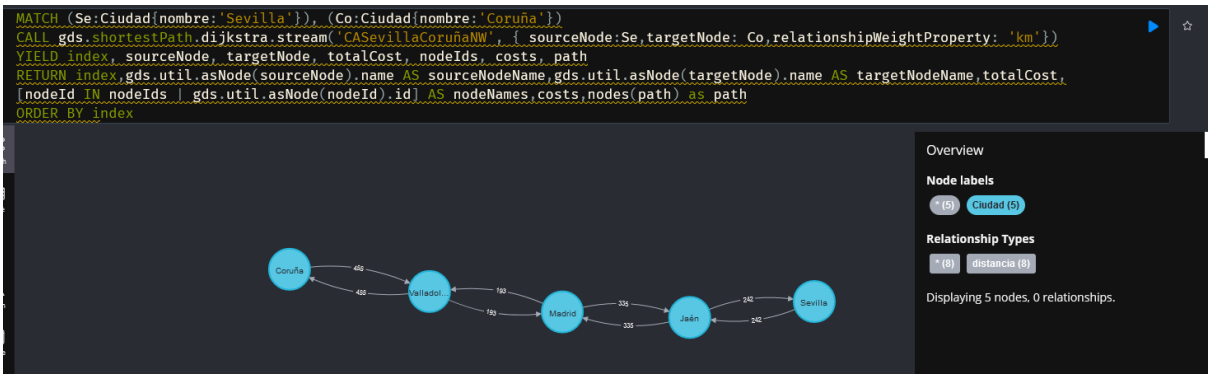
Started streaming 11 records after 8 ms and completed after 27 ms.

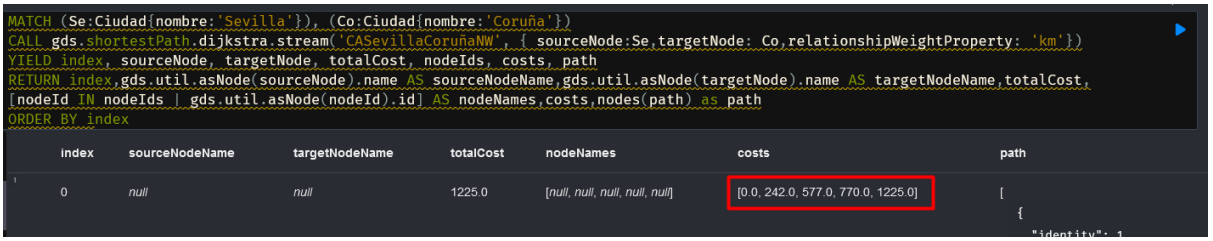
- Camino mínimo (Dijkstra) entre Sevilla y Coruña (no pesado)





- Camino mínimo (Dijkstra) entre Sevilla y Coruña (pesado)





(Sobre el grafo importado a partir de archivos CSV. Apartado 4.2.2. del pdf)

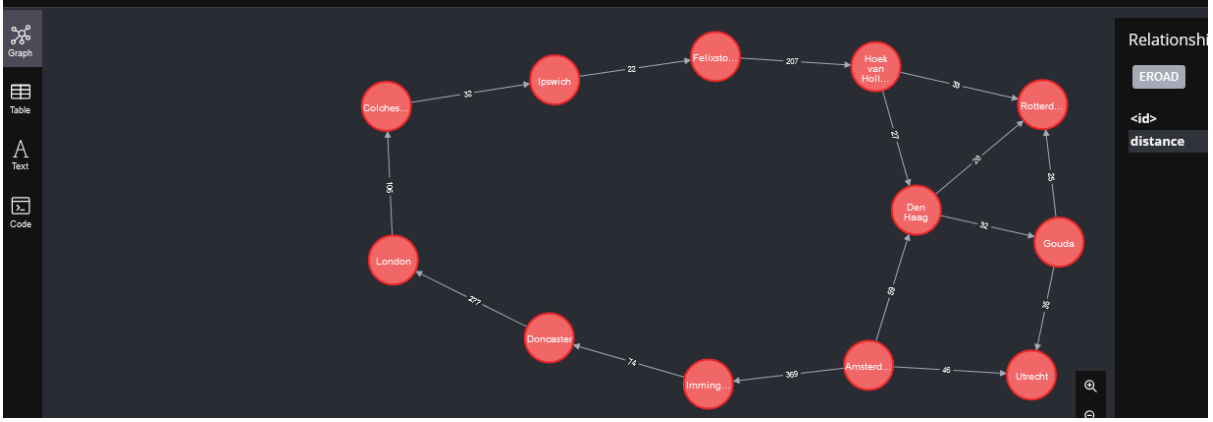
```
WITH "https://github.com/neo4j-graph-analytics/book/raw/master/data/transport-nodes.csv"
AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (place:Place {id:row.id})
SET place.latitude = toFloat(row.latitude),
place.longitude = toFloat(row.latitude),
place.population = toInteger(row.population)
```

Added 12 labels, created 12 nodes, set 48 properties, completed after 2830 ms.

```
1 WITH "https://github.com/neo4j-graph-analytics/book/raw/master/data/transport-relationships.csv" AS uri
2 LOAD CSV WITH HEADERS FROM uri AS row
3 MATCH (origin:Place {id: row.src})
4 MATCH (destination:Place {id: row.dst})
5 MERGE (origin)-[:EROAD {distance: toInteger(row.cost)}]→(destination)
```

Set 15 properties, created 15 relationships, completed after 435 ms.

```
neo4j$ MATCH p=()-[r:EROAD]→() RETURN p LIMIT 25
```



- Camino mínimo (A*) entre Amsterdam y Londres

```
1 CALL gds.graph.project('myGraph2','Place','EROAD', {nodeProperties: ['latitude', 'longitude'],relationshipProperties: 'distance'})
2
```

nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
{ "Place": { "label": "Place", "properties": { "latitude": { "defaultValue": null, "property": "latitude" }, "longitude": { "defaultValue": null, "property": "longitude" } } } }	{ "EROAD": { "orientation": "NATURAL", "aggregation": "DEFAULT", "type": "EROAD", "properties": { "distance": { "defaultValue": null, "property": "distance", "aggregation": "DEFAULT" } } } }	"myGraph2"	12	15	54

```
MATCH (source:Place{id:'Amsterdam'}), (target:Place{id:'London'})
CALL gds.shortestPath.astar.stream('myGraph2',
{sourceNode:source,targetNode:target,latitudeProperty:'latitude',longitudeProperty:'longitude',
relationshipWeightProperty: 'distance'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
RETURN
index,gds.util.asNode(sourceNode).id AS sourceNodeName,gds.util.asNode(targetNode).id AS targetNodeName,totalCost,[nodeId IN
nodeIds | gds.util.asNode(nodeId).id] AS nodeNames,costs,nodes(path) as path
ORDER BY index
```

Overview

Node labels

* (4) Place (4)

Relationship Types

* (3) EROAD (3)

Displaying 4 nodes, 3 relationships.