

Actividad 05. Recorrido de grafos

Recorridos y caminos mínimos

(Sobre el grafo de la actividad anterior)

- Recorrido BFS desde Sevilla

```
MATCH (s:Ciudad {nombre: 'Sevilla'})  
CALL gds.bfs.stream('grafoActividad', { sourceNode: id(s) })  
YIELD nodeIds  
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Recorrido_BFS;
```

The screenshot shows the Neo4j Browser interface with a query window containing the provided Cypher code. Below the results table, there is a 'Table' tab and a 'Text' tab. The 'Text' tab displays the output of the query, which is a list of city names starting from Sevilla and branching out to other cities like Granada, Jaén, Cádiz, Murcia, Madrid, Valencia, Albacete, Valladolid, Bilbao, Zaragoza, Badajoz, Barcelona, Coruña, Vigo, Oviedo, and Gerona.

nodeId	nombre
1	Sevilla
2	Granada
3	Jaén
4	Cádiz
5	Murcia
6	Madrid
7	Valencia
8	Albacete
9	Valladolid
10	Bilbao
11	Zaragoza
12	Badajoz
13	Barcelona
14	Coruña
15	Vigo
16	Oviedo
17	Gerona

- Recorrido DFS desde Sevilla

```
MATCH (s:Ciudad {nombre: 'Sevilla'})  
CALL gds.dfs.stream('grafoActividad', { sourceNode: id(s) })  
YIELD nodeIds  
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Recorrido_DFS;
```

The screenshot shows the Neo4j Browser interface with a query window containing the provided Cypher code. Below the results table, there is a 'Table' tab and a 'Text' tab. The 'Text' tab displays the output of the query, which is a list of city names starting from Sevilla and branching out to other cities like Cádiz, Madrid, Badajoz, Albacete, Murcia, Valencia, Barcelona, Gerona, Zaragoza, Bilbao, Oviedo, Valladolid, Vigo, Coruña, and Granada.

nodeId	nombre
1	Sevilla
2	Cádiz
3	Jaén
4	Madrid
5	Badajoz
6	Albacete
7	Murcia
8	Valencia
9	Barcelona
10	Gerona
11	Zaragoza
12	Bilbao
13	Oviedo
14	Valladolid
15	Vigo
16	Coruña
17	Granada

- Recorrido BFS especificando nodos destino Coruña

```
MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})  
CALL gds.bfs.stream('grafoActividad', { sourceNode: id(s), targetNodes: [id(t)] })  
YIELD nodeIds  
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS BFS_a_Coruna;
```

The screenshot shows the Neo4j Browser interface with a query window containing the provided Cypher code. Below the results table, there is a 'Table' tab and a 'Text' tab. The 'Text' tab displays the output of the query, which is a list of city names starting from Sevilla and branching out to other cities like Granada, Jaén, Cádiz, Murcia, Madrid, Valencia, Albacete, Valladolid, Bilbao, Zaragoza, Badajoz, Barcelona, and Coruña.

nodeId	nombre
1	Sevilla
2	Granada
3	Jaén
4	Cádiz
5	Murcia
6	Madrid
7	Valencia
8	Albacete
9	Valladolid
10	Bilbao
11	Zaragoza
12	Badajoz
13	Barcelona
14	Coruña

- Recorrido DFS especificando nodos destino Coruña

```
MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})  
CALL gds.dfs.stream('grafoActividad', { sourceNode: id(s), targetNodes: [id(t)] })  
YIELD nodeIds  
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS DFS_a_Coruna;
```

```
neo4j$ MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'}) CALL gds.dfs.stream('grafoActividad', { source
```

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

```

MATCH (s:Ciudad { nombre: 'Sevilla' }), (t:Ciudad { nombre: 'Coruña' })
CALL gds.shortestPath.dijkstra.stream('grafoActividad', { sourceNode: id(s), targetNode: id(t) })
YIELD nodeId, totalCost
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost
AS Saltos;

```

```
1 MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})  
2 CALL gds.shortestPath.dijkstra.stream( grafoActividad, { sourceNode: id(s), targetNode: id(t) })  
3 YIELD nodeId, totalCost  
4 RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost AS Saltos;
```

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

```
MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})  
CALL gds.shortestPath.dijkstra.stream('grafoActividad', {  
    sourceNode: id(s),  
    targetNode: id(t),  
    relationshipWeightProperty: 'distancia'  
})  
YIELD nodeIds, totalCost  
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost  
AS Kilometros;
```

```
1 MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})  
2 CALL gds shortestPath.dijkstra.stream('grafoActividad', {  
3   sourceNode: id(s)  
4   targetNode: id(t),  
5   relationshipWeightProperty: 'distancia'  
6 })  
7 YIELD nodeId, totalCost  
8 RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost AS Kilometros;
```

- Camino mínimo (A^*) entre Sevilla y Coruña. Comenta el resultado. ¿Qué tendrías que modificar en el grafo actual para obtener un resultado diferente?. Verifícalo.

```
MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})
CALL gds.shortestPath.astar.stream('grafoActividad', {
    sourceNode: id(s),
    targetNode: id(t),
```

```

latitudeProperty: 'lat',
longitudeProperty: 'lon',
relationshipWeightProperty: 'distancia'
})
YIELD nodeIds, totalCost
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino_AStar, tot
alCost AS Distancia;

```

```

1 MATCH (s:Ciudad {nombre: 'Sevilla'}), (t:Ciudad {nombre: 'Coruña'})
2 CALL gds.shortestPath.astar.stream('grafoActividad', {
3   sourceNode: id(s),
4   targetNode: id(t),
5   latitudeProperty: 'lat',
6   longitudeProperty: 'lon',
7   relationshipWeightProperty: 'distancia'
8 })
9 YIELD nodeIds, totalCost
10 RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino_AStar, totalCost AS Distancia;

```

	Camino_AStar	Distancia
A	["Sevilla", "Jaén", "Madrid", "Valladolid", "Coruña"]	1225.0

Para obtener un resultado diferente, tendrías que modificar la distancia de las aristas del camino actual. Por ejemplo, si aumentas drásticamente la distancia del tramo Madrid-Valladolid, el algoritmo descartará esa ruta y buscará una alternativa con menor coste total. En el caso de A*, también podrías alterar las coordenadas (lat/lon) para "engañar" a la heurística y cambiar la prioridad de exploración.