

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 Cypher query being executed in the Neo4j interface. Below the query, the result is displayed as a table with one row containing a list of city names.

Recorrido_BFS
["Sevilla", "Granada", "Jaén", "Cádiz", "Murcia", "Madrid", "Valencia", "Albacete", "Valladolid", "Bilbao", "Zaragoza", "Badajoz", "Barcelona", "Coruña", "Vigo", "Oviedo", "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 Cypher query being executed in the Neo4j interface. Below the query, the result is displayed as a table with one row containing a list of city names.

Recorrido_DFS
["Sevilla", "Cádiz", "Jaén", "Madrid", "Badajoz", "Albacete", "Murcia", "Valencia", "Barcelona", "Gerona", "Zaragoza", "Bilbao", "Oviedo", "Valladolid", "Vigo", "Coruña", "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 Cypher query being executed in the Neo4j interface. Below the query, the result is displayed as a table with one row containing a list of city names.

BFS_a_Coruna
["Sevilla", "Granada", "Jaén", "Cádiz", "Murcia", "Madrid", "Valencia", "Albacete", "Valladolid", "Bilbao", "Zaragoza", "Badajoz", "Barcelona", "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', { sourceNode: id(s), targetNode: id(t) })
```

DFS_a_Coruña	
1	["Sevilla", "Cádiz", "Jaén", "Madrid", "Badajoz", "Albacete", "Murcia", "Valencia", "Barcelona", "Gerona", "Zaragoza", "Bilbao", "Oviedo", "Valladolid", "Vigo", "Coruña"]

- 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 nodeIds, 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 nodeIds, totalCost
4 RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost AS Saltos;
```

Camino	
1	["Sevilla", "Jaén", "Madrid", "Valladolid", "Coruña"]

- 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 nodeIds, totalCost
8 RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino, totalCost AS Kilometros;
```

Camino	
1	["Sevilla", "Jaén", "Madrid", "Valladolid", "Coruña"]

- 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 nodeId, totalCost
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino_AStar, totalCost AS Distancia;

```

```

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

```

Camino_AStar	Distancia
["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.