Ingeniería en Sistemas Computacionales

Minería de Grafos

Mapa mental de los algortimos de caminos y búsqueda

Marco Ricardo Cordero Hernández

Tlaquepaque, Jal., 13 de noviembre de 2023

Minimum Weight Spanning Tree

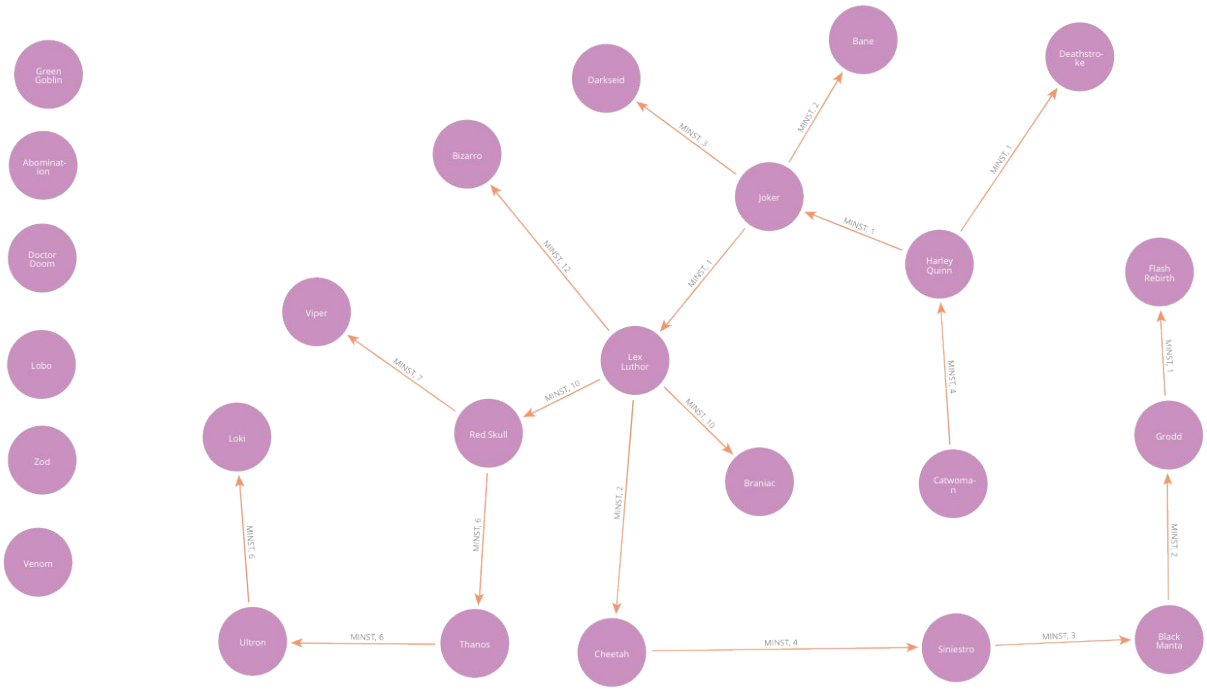
Código de implementación

```
// Minimum Weight Spanning Tree
// Hecho en la versión 2.5.3


// 1. Creación de subgrafo
// Subgrafo no dirigido (el algoritmo ignora las direcciones de las relaciones)
CALL gds.graph.project(
    'myUndirectedGraph',
    'Villian',
    {
        COMPANION: {
            orientation: 'UNDIRECTED',
            Properties: 'Weight'
        }
    }
);


// 2. Creación de las relaciones
MATCH (source:Villian {Name: 'Catwoman'})
CALL gds.spanningTree.write(
    'myUndirectedGraph',
    {
        sourceNode: ID(source),
        relationshipWeightProperty: 'Weight',
        writeProperty: 'writeCost',
        writeRelationshipType: 'MINST'
    }
)
YIELD preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount
RETURN preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount;
```

Ejecución del código

preProcessingMillis	computeMillis	writeMillis	effectiveNodeCount
2	6	214	19

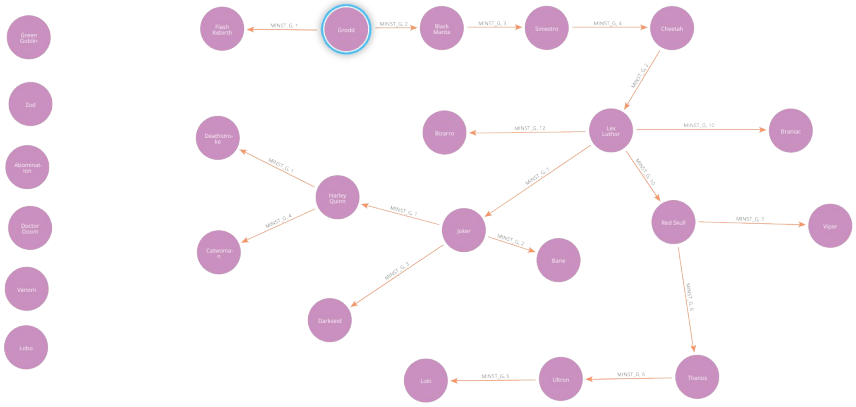
Demostración en Bloom



Ejercicios adicionales

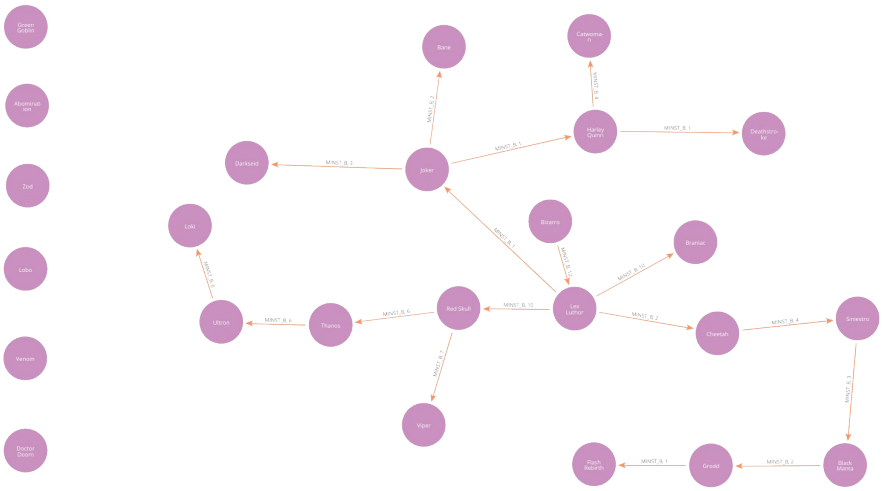
```
// Probar con Grodd y guardar el camino resultante
MATCH (source:Villian {Name: 'Grodd'})
CALL gds.spanningTree.write(
    'myUndirectedGraph',
    {
        sourceNode: ID(source),
        relationshipWeightProperty: 'Weight',
        writeProperty: 'writeCost',
        writeRelationshipType: 'MINST_G'
    }
)
YIELD preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount
RETURN preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount;
```

preProcessingMillis	computeMillis	writeMillis	effectiveNodeCount
0	3	43	19



```
// Probar con Bizarro y guardar el camino resultante
MATCH (source:Villian {Name: 'Bizarro'})
CALL gds.spanningTree.write(
    'myUndirectedGraph',
    {
        sourceNode: ID(source),
        relationshipWeightProperty: 'Weight',
        writeProperty: 'writeCost',
        writeRelationshipType: 'MINST_B'
    }
)
YIELD preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount
RETURN preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount;
```

preProcessingMillis	computeMillis	writeMillis	effectiveNodeCount
0	4	24	19



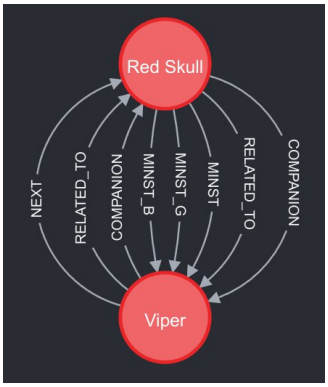
Depth First Search

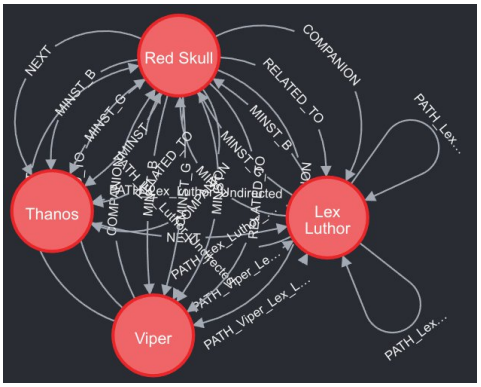
Código de implementación

```
// Depth First Search


// 1. Creación de subgrafos
// Subgrafo no dirigido
CALL gds.graph.project(
    'myUndirectedGraph',
    'Villian',
    {
        COMPANION: {
            orientation: 'UNDIRECTED',
            Properties: 'Weight'
        }
    }
);


// Subgrafo dirigido
CALL gds.graph.project(
    'myDirectedGraph',
    'Villian',
    {
        COMPANION: {
            orientation: 'NATURAL',
            Properties: 'Weight'
        }
    }
);


// 2. Ejecución del algoritmo como frase de búsqueda en bloom
// Las siguientes frases asumen la ejecución del paso anterior
// 2.1 Frase: DFS desde $villian a $n nivel(es)
//        $villian: String (label-key Villian:Name)
//        $n: Integer
MATCH (source:Villian {Name: $villian})
CALL gds.dfs.stream(
    'myUndirectedGraph',
    {
        sourceNode: source,
        maxDepth: $n
```

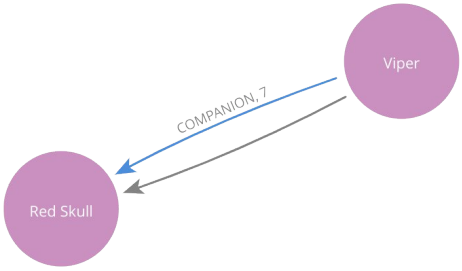
```
    }
)
YIELD path RETURN path;


// 2.2 Frase: DFS dirigido desde $villian a $n nivel(es)
//        $villian: String (label-key Villian:Name)
//        $n: Integer
MATCH (source:Villian {Name: $villian})
CALL gds.dfs.stream(
    'myDirectedGraph',
    {
        sourceNode: source,
        maxDepth: $n
    }
)
YIELD path RETURN path;
```

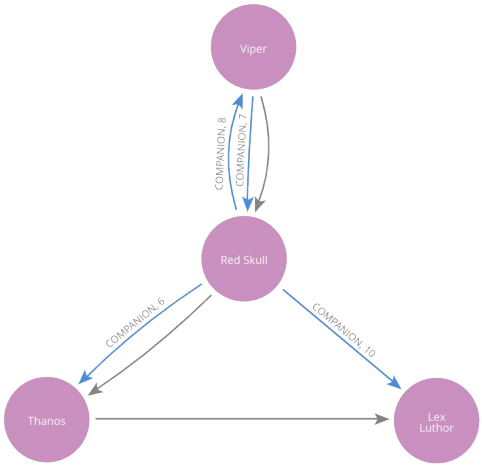
Ejecución del código





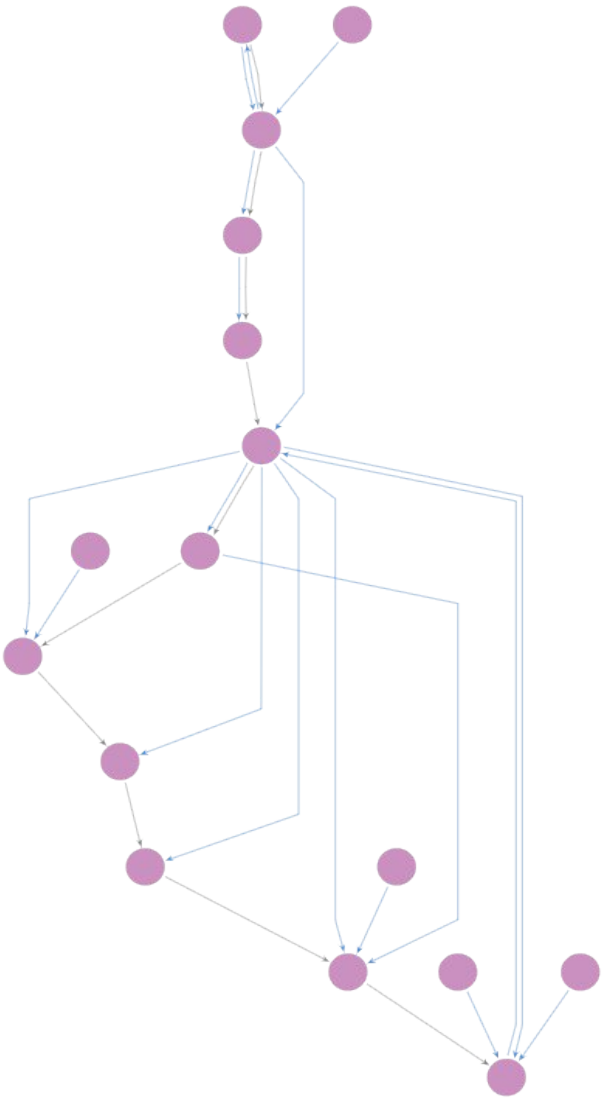
Demostración en Bloom





Ejercicios adicionales

DFS dirigido desde Viper con nivel de profundidad 3 (Hecho con el mismo código anterior).



Con la adición de un nuevo nivel de profundidad, el camino resultante desde el origen resulta más extenso, y, al analizar las relaciones pertenecientes a cada nivel, nuevos nodos se van descubriendo. Con el aumento del parámetro ***n***, la cantida de nuevos nodos en la estructura de árbol también aumenta proprocionalmente.

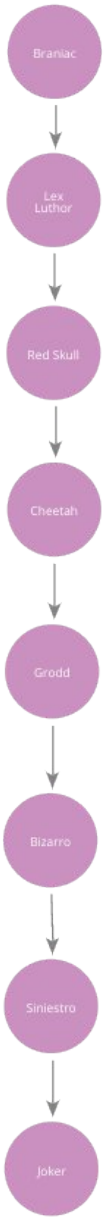
```
// Probar con Braniac
//      - n = 1
```

```
//        - n = 2
//        - n = 2 ND
//        - n = 3
```

Profundidad = 1

Profundidad = 2

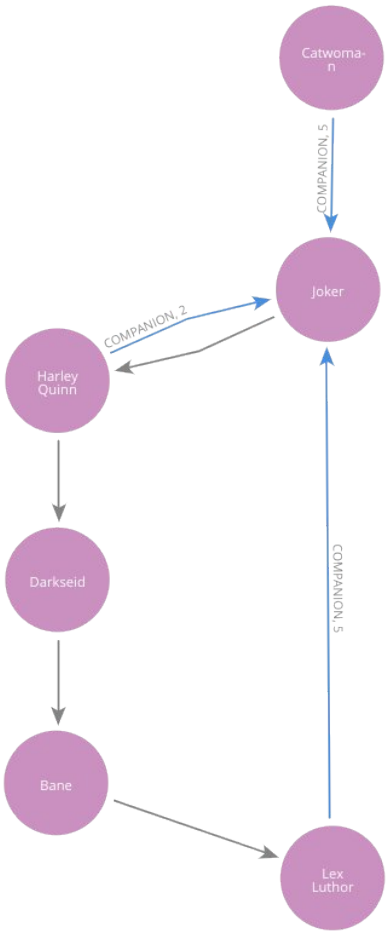
Profundidad = 3



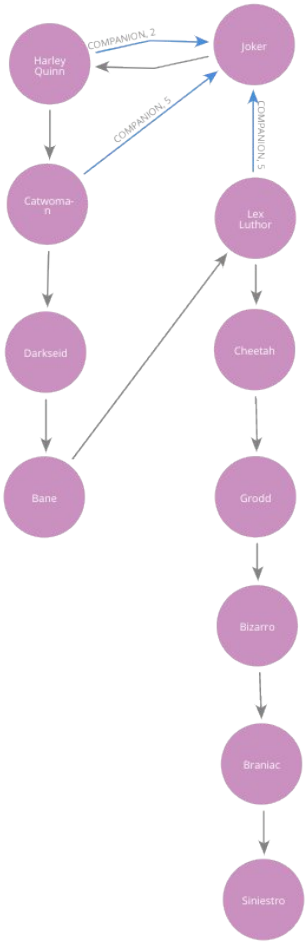
Profundidad = 3

No dirigido

```
// En Bloom, haz lo necesario para que reciba el nombre del villano como parámetro
// y prueba con Joker a 1 y 2 niveles de profundidad
// (Misma frase de búsqueda ya creada)
```



Profundidad = 1



Profundidad = 2