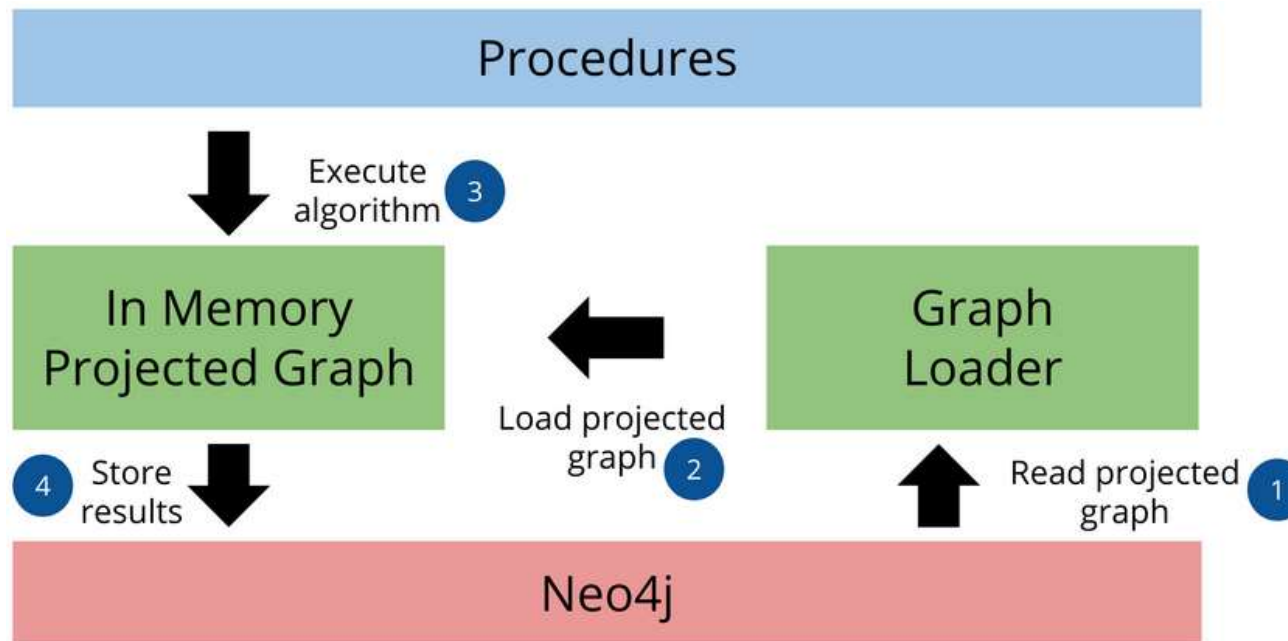


<https://neo4j.com/docs/graph-data-science/current/>



Algorithmen-Ausprägungen

- stream
Liefert ein Ergebnis wie eine normal Cypher Query
- mutate
Schreibt Ergebnisse in den In-Memory Graph
- write
Schreibt Ergebnisse in die Neo4j-Datenbank

<https://neo4j.com/docs/graph-data-science/current/graph-create-cypher/>

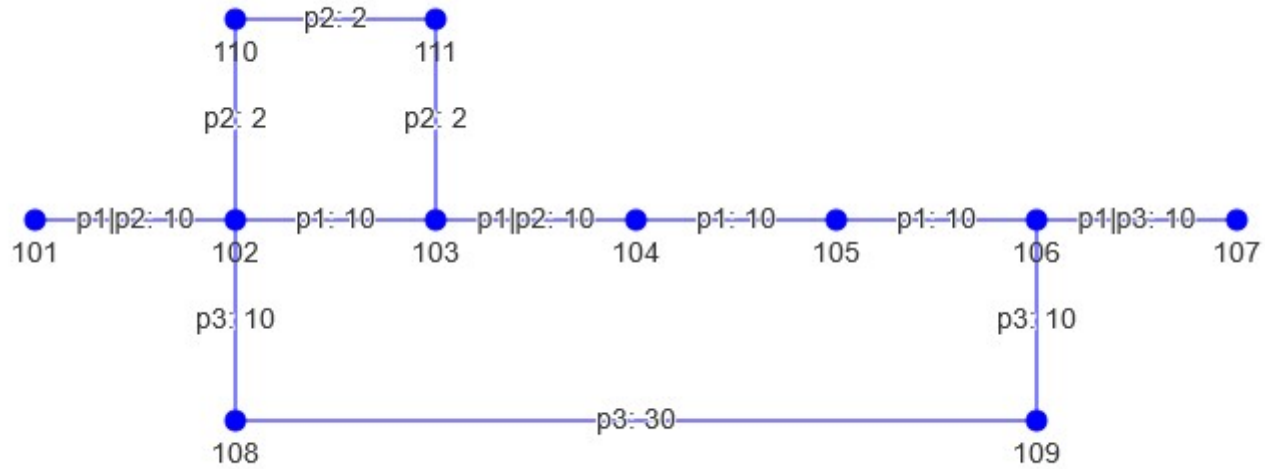
Unidirektional

```
CALL gds.graph.create.cypher(  
  'g',  
  'MATCH (n:Node) RETURN id(n) AS id',  
  'MATCH (n1:Node)-[e:Edge]->(n2:Node) RETURN id(n1) AS source, id(n2) AS target, e.w AS w'  
);
```

Symmetrisch

```
CALL gds.graph.create.cypher(  
  'g',  
  'MATCH (n:Node) RETURN id(n) AS id',  
  'MATCH (n1:Node)-[e:Edge]->(n2:Node) RETURN id(n1) AS source, id(n2) AS target, e.w AS w  
  UNION  
  MATCH (n1:Node)<-[e:Edge]-(n2:Node) RETURN id(n1) AS source, id(n2) AS target, e.w AS w'  
);
```

<https://neo4j.com/docs/graph-data-science/current/algorithms/dijkstra-source-target/>



```
MATCH (n1:Node {l: '101'}), (n2:Node {l: '107'})
```

```
CALL gds.shortestPath.dijkstra.stream('g', {
```

```
  sourceNode: n1,
```

```
  targetNode: n2
```

```
})
```

```
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
```

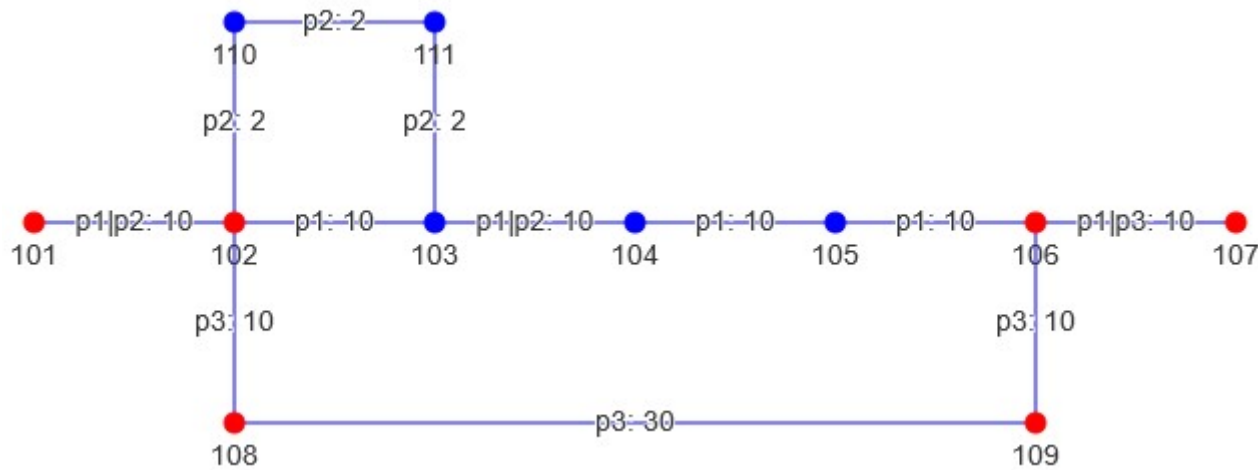
```
with [nodeId IN nodeIds | gds.util.asNode(nodeId).nid] AS p
```

```
with p
```

```
UNWIND p as nid
```

```
RETURN nid;
```

"nid"
101
102
108
109
106
107



Freitag, 18. März 2022 17:11

```
MATCH (n1:Node {l: '101'}), (n2:Node {l: '107'})
```

```
CALL gds.shortestPath.dijkstra.stream('g', {
```

```
  sourceNode: n1,
```

```
  targetNode: n2,
```

```
  relationshipWeightProperty: 'w'
```

```
})
```

```
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
```

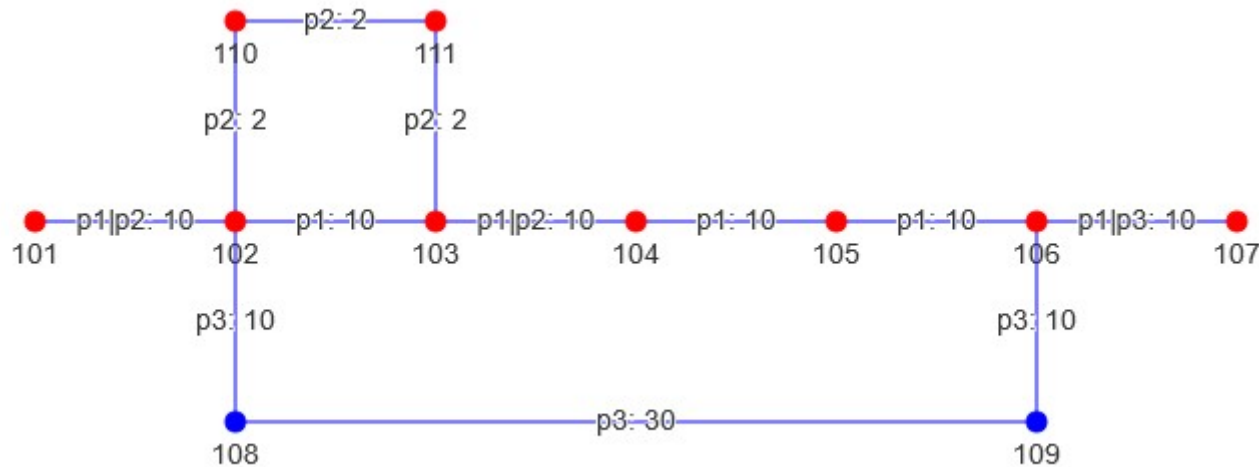
```
with [nodeId IN nodeIds | gds.util.asNode(nodeId).nid] AS p
```

```
with p
```

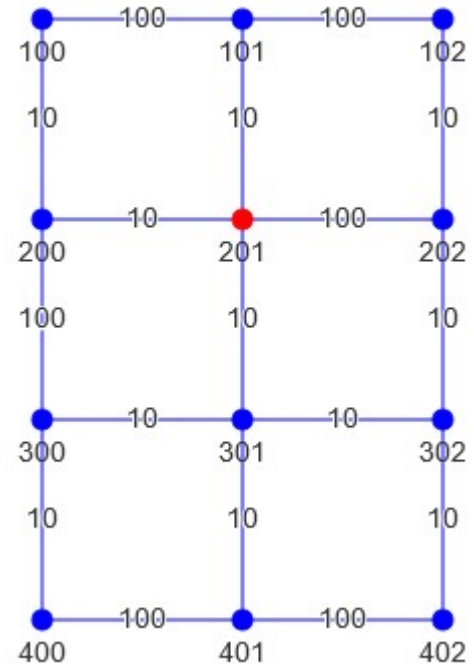
```
UNWIND p as nid
```

```
RETURN nid;
```

"nid"
101
102
110
111
103
104
105
106
107



<https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/minimum-weight-spanning-tree/>

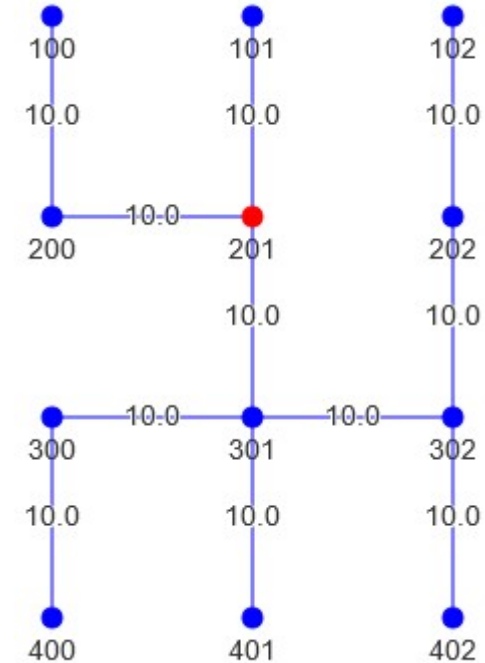


Schreibt in den Graphen hinein

- Ursprungsgraph hat Kantentyp Edge
- Neue Kanten zum Kantentyp Edge1 werden angelegt

```
MATCH (n:Node {nid: 201})
CALL gds.alpha.spanningTree.minimum.write('g', {
  startNodeid: id(n),
  relationshipWeightProperty: 'w',
  writeProperty: 'Edge1',
  weightWriteProperty : 'w'
})
YIELD effectiveNodeCount
return effectiveNodeCount;
```

Graph mit Edge1 wird ausgelesen und angezeigt



Closeness Centrality

<https://neo4j.com/docs/graph-data-science/current/algorithms/closeness-centrality/>

$$C(u) = \frac{n - 1}{\sum_{v=1}^{n-1} d(u, v)}$$

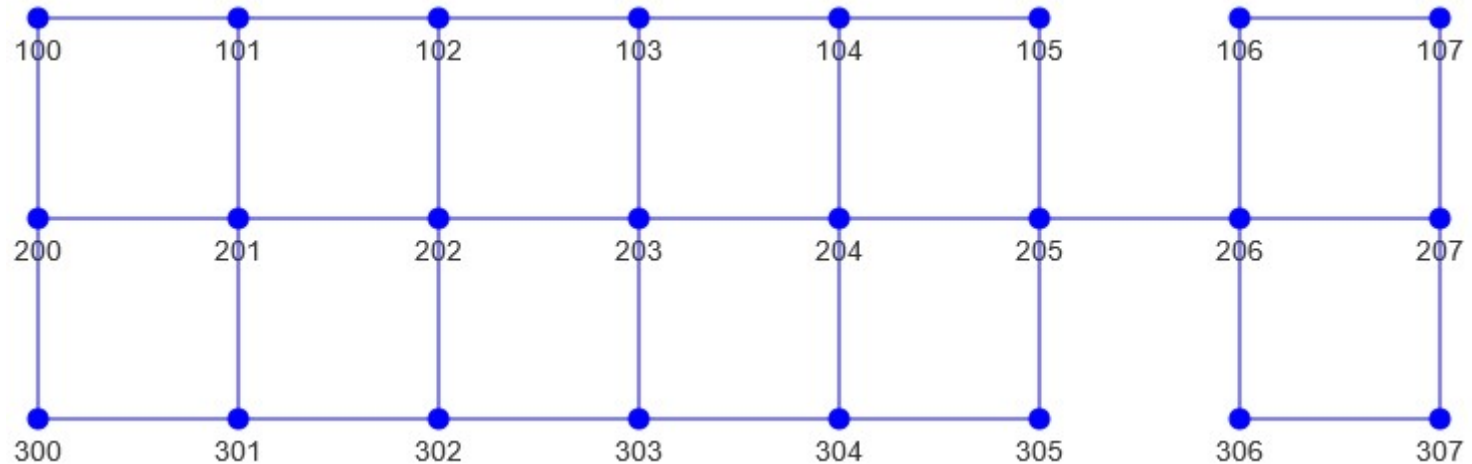
Betweenness Centrality

<https://neo4j.com/docs/graph-data-science/current/algorithms/betweenness-centrality/>

$$B(u) = \sum_{s \neq u \neq t} \frac{p(s, u, t)}{p(s, t)}$$

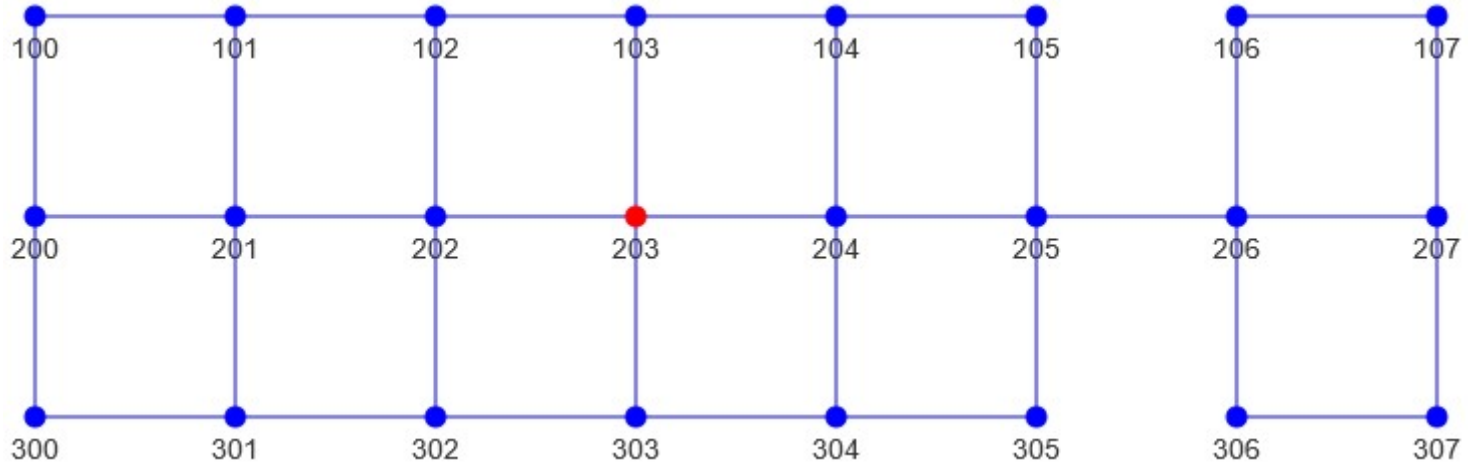
u	Knoten
n	Anzahl Knoten
d(u,v)	Distanz des kürzesten Pfades zwischen u und v

u	Knoten
p	Anzahl der kürzesten Pfade zwischen s und t
p(u)	Anzahl der kürzesten Pfade zwischen s und t



	nid	cc
0	203	0.359375
1	204	0.359375
2	202	0.328571
3	205	0.328571
4	103	0.302632
5	104	0.302632
6	303	0.302632
7	304	0.302632
8	102	0.280488
9	105	0.280488
10	201	0.280488
11	206	0.280488
12	302	0.280488
13	305	0.280488
14	101	0.244681
15	301	0.244681
16	200	0.230000
17	207	0.230000
18	106	0.225490
19	306	0.225490
20	100	0.205357
21	300	0.205357
22	107	0.191667
23	307	0.191667

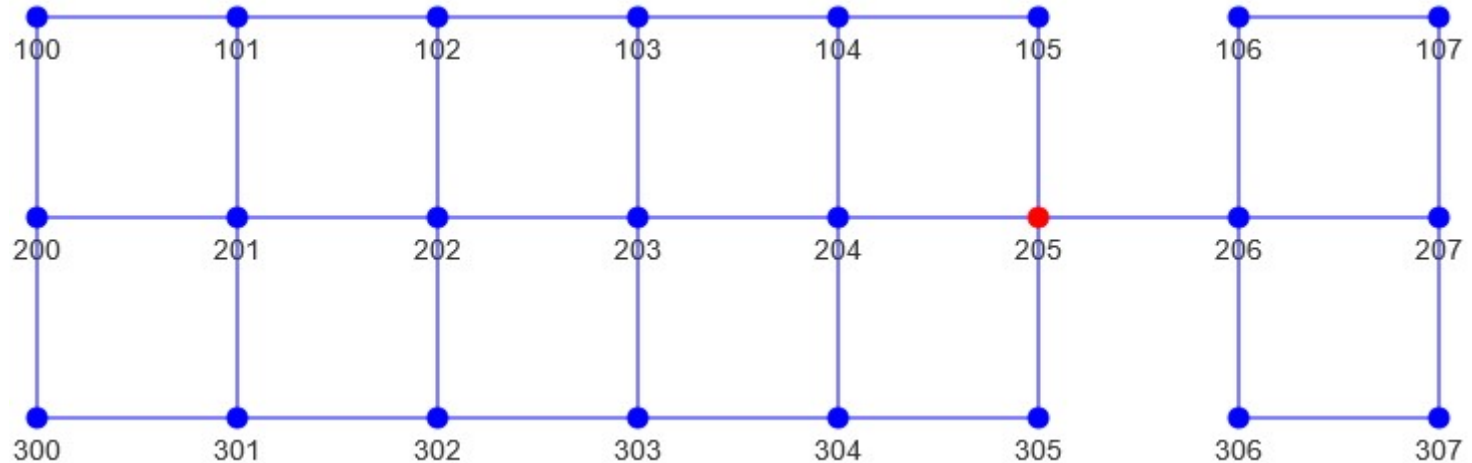
```
CALL gds.alpha.closeness.stream('g')
YIELD nodeld, centrality
RETURN gds.util.asNode(nodeld).nid AS nid, centrality as cc
ORDER BY cc DESC;
```



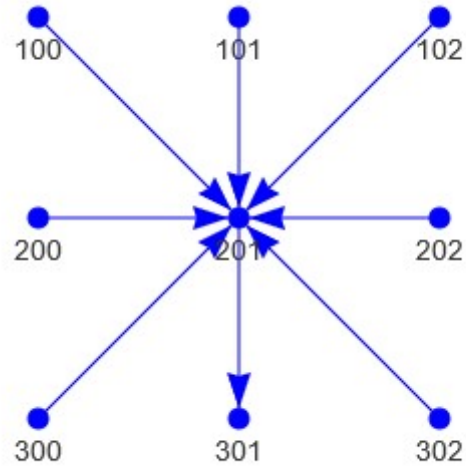
Mittwoch, 16. März 2022 15:45

	nid	bc
0	205	220.542857
1	204	189.504762
2	206	186.666667
3	203	162.885714
4	202	128.085714
5	201	81.104762
6	103	65.490476
7	303	65.490476
8	302	60.890476
9	102	60.890476
10	104	53.514286
11	304	53.514286
12	207	42.666667
13	101	40.714286
14	301	40.714286
15	105	21.728571
16	305	21.728571
17	200	20.542857
18	106	19.666667
19	306	19.666667
20	300	4.328571
21	100	4.328571
22	107	1.666667
23	307	1.666667

```
CALL gds.betweenness.stream('g')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).nid AS nid, score as bc
ORDER BY score DESC;
```

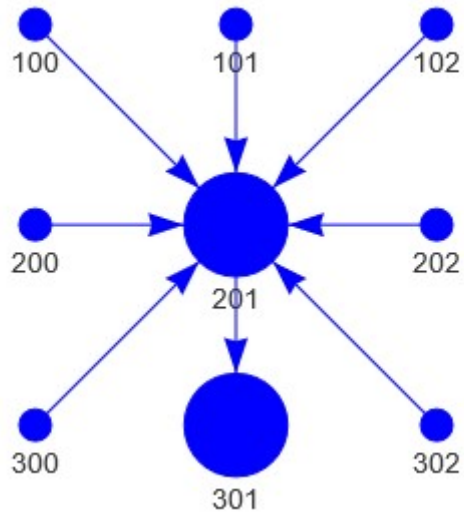


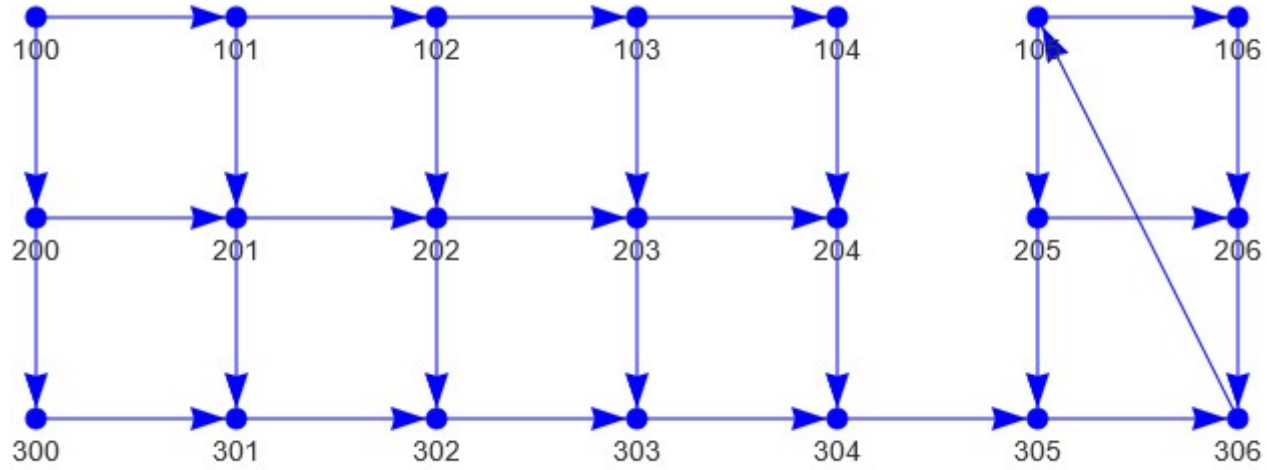
<https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/>

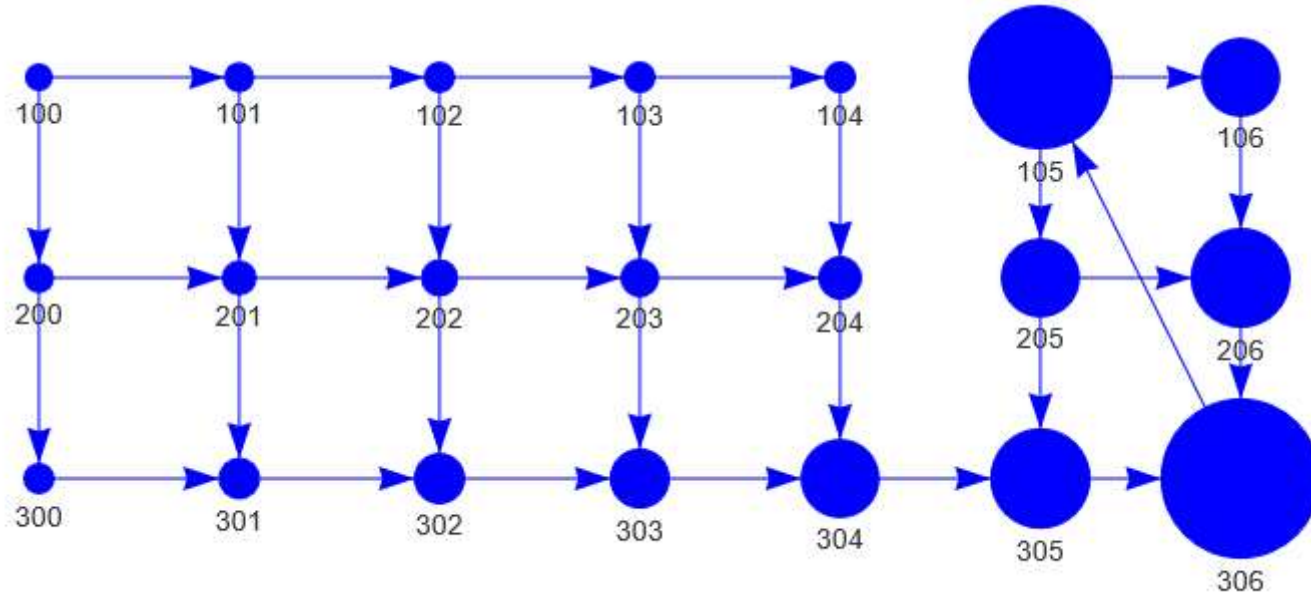


$$PR(A) = (1 - d) + d\left(\frac{PR(T_1)}{C(T_1)} + \dots + \frac{PR(T_n)}{C(T_n)}\right)$$

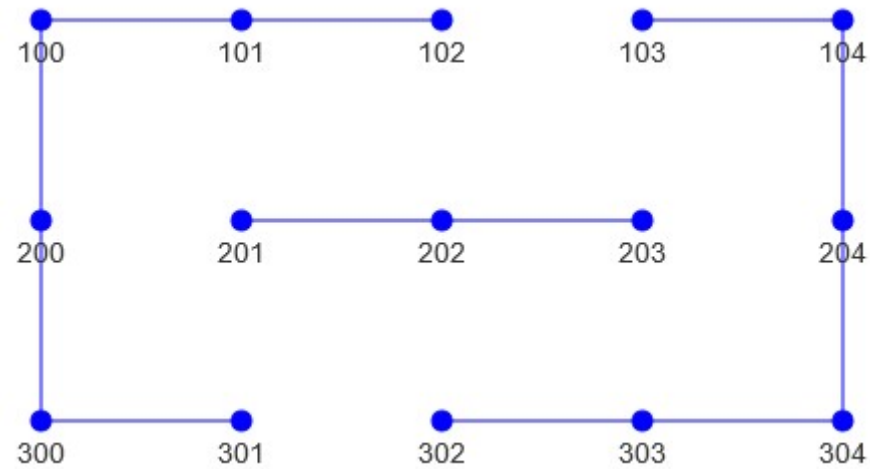
```
CALL gds.pageRank.stream('g')  
YIELD nodelid, score  
RETURN gds.util.asNode(nodelid).nid AS nid, score  
ORDER BY score DESC;
```







<https://neo4j.com/docs/graph-data-science/current/algorithms/strongly-connected-components/>



```
CALL gds.alpha.scc.stream('g')  
YIELD nodeId, componentId  
RETURN gds.util.asNode(nodeId).nid AS nid, componentId AS cid  
ORDER BY cid;
```

