The AB

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (a)-[:TO]->(b)

CALL gds.graph.project('ab',['Node'],'TO')

MATCH (a:Node {name:'a'})
CALL gds.randomWalk.stream(
   'ab',
   {walkLength: 2,
   walksPerNode:100,
   sourceNodes: a})
YIELD nodeIds, path
UNWIND [node IN nodes(path) | node.name ] AS node
RETURN node, round(count(*),3) AS score ORDER BY node


MATCH (a:Node {name:'a'})
CALL gds.pageRank.stream('ab', {
   maxIterations:1,
   sourceNodes: a,
   dampingFactor: 0.99})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,3) AS score, 0.99 AS dampingFactor
ORDER BY node

MATCH (a:Node {name:'a'})
CALL gds.eigenvector.stream('ab', {
   maxIterations:100,
   sourceNodes: a})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,3) AS score, 100 AS iteration,
'eigenvector' AS algo
ORDER BY node
```

The AB,AC

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (c:Node {name:'c'})
MERGE (a)-[:TO]->(b)
MERGE (a)-[:TO]->(c)

CALL gds.graph.project('abac',['Node'],'TO')
```

The ABCA

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (c:Node {name:'c'})
MERGE (a)-[:TO]->(b)
MERGE (b)-[:TO]->(c)
MERGE (c)-[:TO]->(a)

CALL gds.graph.project('abca',['Node'],'TO')

MATCH (a:Node {name:'a'})
CALL gds.pageRank.stream('abca', {
   maxIterations:1000,
   tolerance:0.0001,
   sourceNodes: a,
   scaler:'max',
   dampingFactor: 0.999})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,2) AS score
ORDER BY node


MATCH (a:Node {name:'a'})
CALL gds.eigenvector.stream('abca', {
   maxIterations:1000,
   tolerance:0.0001,
   sourceNodes: a,
   scaler:'max'})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,2) AS score
ORDER BY node
```

The AB,BA,BC

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (c:Node {name:'c'})
MERGE (a)-[:TO]->(b)
MERGE (b)-[:TO]->(a)
MERGE (b)-[:TO]->(c)

CALL gds.graph.project('loop',['Node'],'TO')

MATCH (s:Node {name:'a'})
CALL gds.randomWalk.stream(
   'loop',
   {walkLength: 1000,
   walksPerNode: 100,
   sourceNodes: s})
YIELD nodeIds, path
UNWIND [node IN nodes(path) | node.name ] AS node
RETURN node, count(*) AS count ORDER BY node


MATCH (a:Node {name:'a'})
CALL gds.pageRank.stream('loop', {
   maxIterations:1000,
   sourceNodes: a,
   scaler:'max',
   dampingFactor: 0.999})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,2) AS score
ORDER BY node
```

The AB,AC,BA,BC (Bored)

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (c:Node {name:'c'})
MERGE (s:Node {name:'s'})
MERGE (a)-[:TO {weight:0.333}]->(b)
MERGE (a)-[:TO {weight:0.666}]->(c)
MERGE (b)-[:TO {weight:0.666}]->(a)
MERGE (b)-[:TO {weight:0.333}]->(c)
MERGE (s)-[:TO {weight:0.333}]->(a)
MERGE (s)-[:TO {weight:0.666}]->(b)

CALL gds.graph.project('bored',['Node'],{TO: { properties: "weight" }})

MATCH (s:Node) WHERE s.name in ['s']
WITH collect(s) AS sources
CALL gds.pageRank.stream('bored', {
   maxIterations:1000,
   sourceNodes: sources,
   relationshipWeightProperty: 'weight',
   scaler:'max',
   dampingFactor: 0.999})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,2) AS score
ORDER BY node
```

The great Y

```
MERGE (a:Node {name:'a'})
MERGE (b:Node {name:'b'})
MERGE (c:Node {name:'c'})
MERGE (d:Node {name:'d'})
MERGE (e:Node {name:'e'})
MERGE (f:Node {name:'f'})
MERGE (a)-[:TO]->(b)
MERGE (a)-[:TO]->(c)
MERGE (b)-[:TO]->(d)
MERGE (b)-[:TO]->(e)
MERGE (c)-[:TO]->(f)
MERGE (d)-[:TO]->(f)
MERGE (e)-[:TO]->(f)

CALL gds.graph.project('y',['Node'],'TO')


MATCH (a:Node {name:'a'})
CALL gds.randomWalk.stream(
   'y',
   {walkLength: 1000,
   walksPerNode: 1000,
   sourceNodes: a})
YIELD nodeIds, path
UNWIND [node IN nodes(path) | node.name ] AS node
RETURN node, count(*) AS count ORDER BY node

MATCH (a:Node {name:'a'})
CALL gds.pageRank.stream('y', {
   maxIterations:1000,
   sourceNodes: a,
   scaler:'max',
   dampingFactor: 0.999})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,2) AS score
ORDER BY node
```

The social network

```
MERGE (mi:Person {name:'Michael'})
MERGE (br:Person {name:'Bridget'})
MERGE (do:Person {name:'Doug'})
MERGE (al:Person {name:'Alice'})
MERGE (da:Person {name:'David'})
MERGE (ja:Person {name:'James'})
MERGE (am:Person {name:'Amy'})
MERGE (ch:Person {name:'Charles'})
MERGE (ma:Person {name:'Mark'})
MERGE (ja)-[:FOLLOWS]->(da)
MERGE (da)-[:FOLLOWS]->(am)
MERGE (ch)-[:FOLLOWS]->(do)
MERGE (ma)-[:FOLLOWS]->(do)
MERGE (ma)-[:FOLLOWS]->(al)
MERGE (do)-[:FOLLOWS]->(ma)
MERGE (mi)-[:FOLLOWS]->(do)
MERGE (mi)-[:FOLLOWS]->(al)
MERGE (mi)-[:FOLLOWS]->(br)
MERGE (br)-[:FOLLOWS]->(mi)
MERGE (br)-[:FOLLOWS]->(do)
MERGE (br)-[:FOLLOWS]->(al)
MERGE (al)-[:FOLLOWS]->(mi)
MERGE (al)-[:FOLLOWS]->(br)
MERGE (al)-[:FOLLOWS]->(ch)
MERGE (al)-[:FOLLOWS]->(do)

CALL gds.graph.project('social',['Person'],'FOLLOWS')

MATCH (j:Person {name:'James'})
CALL gds.randomWalk.stream(
   'social',
   {walkLength: 1000,
   walksPerNode: 10000,
   sourceNodes: j})
YIELD nodeIds, path
UNWIND [node IN nodes(path) | node.name ] AS node
RETURN node, count(*) AS count ORDER BY node

MATCH (j:Person {name:'James'})
CALL gds.pageRank.stream('social', {
   maxIterations:10000,
   sourceNodes: j,
   dampingFactor: 0.999})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, round(score,5) AS score ORDER BY node
```