Graph Analysis using Spark, GraphX and Pregel

A graph is a mathematical representation for a Social Network. The graph consists of Vertices (which represent entities or people) and Edges connecting vertices (which represents the relationship between nodes).

A graph is useful as it provides a high level of information about certain features of the graph like:

1. **Connectedness** – The number of edges from each node. This will represent the 'popularity' of a node.

2. **Average Path Length** – The 'distance' between any two nodes. This can be used to represent the level of 'friendship' between two people, whether they know each other through common friends.

3. **Number of triangles** – Any 3 nodes connected in a network is called a triangle. A network with more triangles is considered more stable than a network with less triangles and more open nodes.

In this project, we will be using the GraphX library in Spark which provides new datatypes like VertexRDD, EdgeRDD and Edge to replicate a graph. We will also be using the Pregel API which will help us traverse the graph and find out properties like degrees of separation.

We will be working with two datasets. The first dataset contains the Superhero names and their unique ID, which we will use as a Vertex ID. The second dataset we will be using is the Superhero Co-appearance dataset, which contains all the superheroes in the Marvel Universe. Each row of the dataset contains the Unique ID of a Superhero followed by the IDs of the Superheroes he/she has made a co-appearance with in a comic.

Creating the Superhero Vertices:

The first step would be to parse the Superhero names dataset and get their names along with their IDs. We will be using the function parseNames to perform this task. This function will work in the following way:

1. For each row in the dataset, split based on the '\' delimiter.

2. Check if the row has length greater than 1. If yes, then trim the first field and convert it to Long and store it a heroID.

3. Check if the heroID is less than 6487 (The ones greater than that are arbitrary numbers). If yes, then return the first field (containing Hero ID) trimmed and converted to long and the second field (containing the Hero name).

This function will be called during the formation of the Vertices of our network. The HeroID will work as the VertexID.

Creating the Superhero Network Edges:

Now we will create the Superhero network. For this we will create a function makeEdges. The function will input each row of the Superhero Co-appearance file and return an Edge datatype list using the number of connections of the first ID in a row with the following IDs.

1. The function will first create a mutable List Buffer 'edges'.

2. It will split the input line based on the delimiter ' '.

3. The first ID in the line will be stored in the variable 'origin'.

4. For all the other IDs, an Edge variable will be created with the origin and current ID, both converted to Long. This Edge variable will be added to the edges list.

5. The edges list will be returned.

The Edge datatype also has an 'attribute' variable which can help keep track of physical distances and other parameters. But we have not used it.

Performing the Network Analytics:

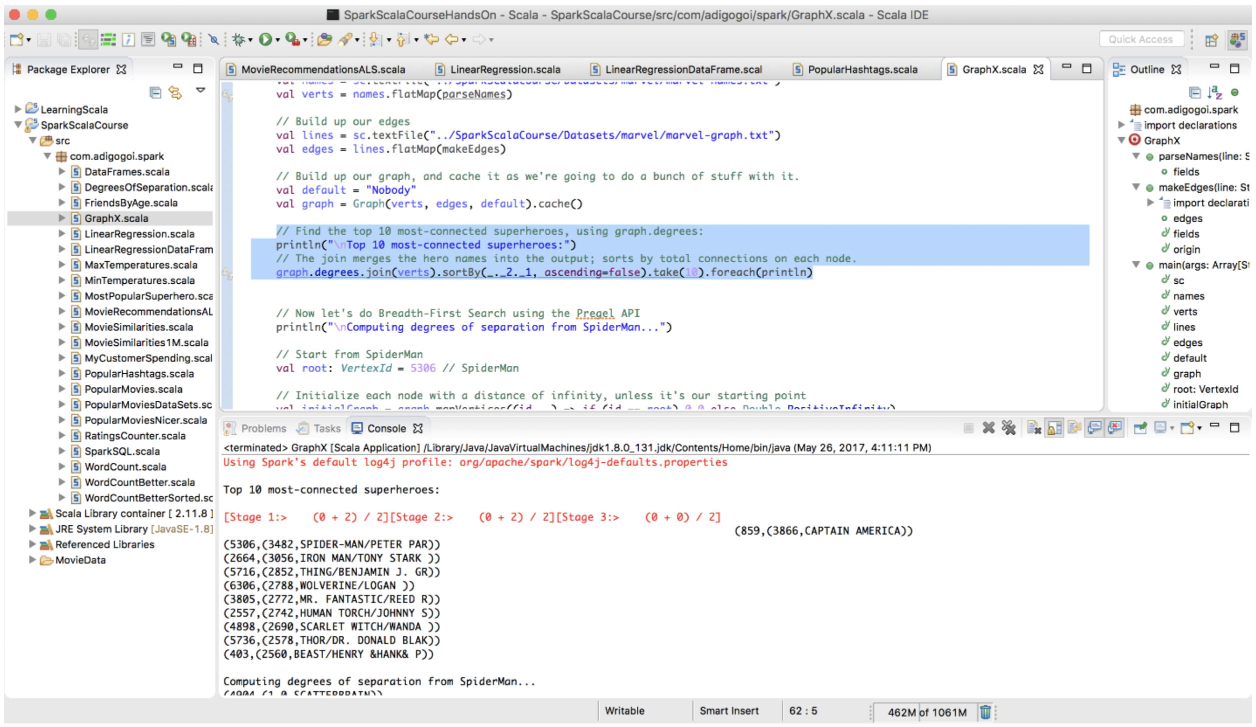
Our main function will first read the names and network files and store them in datasets in the right format. Then we will call the functions mentioned above to create the network.

The vertices and edges dataset will be merged into the 'graph' dataset, which will have the final network. We will cache it as we may need to use it more than once.

We will use Pregel as our network traversing API for this network. Pregel provides the general mechanism to perform general algorithms on our graph like Breadth First Algorithm that we will be using for this project. The explanation of this algorithm is out of scope for this project but in general it involves keeping a node as root node the sending of messages between nodes and recording the distance between them with respect to the root node.

Some of the analytics that we have performed on the network are:

1. **Getting the 'most-connected' Superheroes** – The nodes with the most edges connected to them represent the entities that have more connections, which can be friends or co-workers. In this operation, we are finding out the most famous hero based on the number of co-appearances they have made.

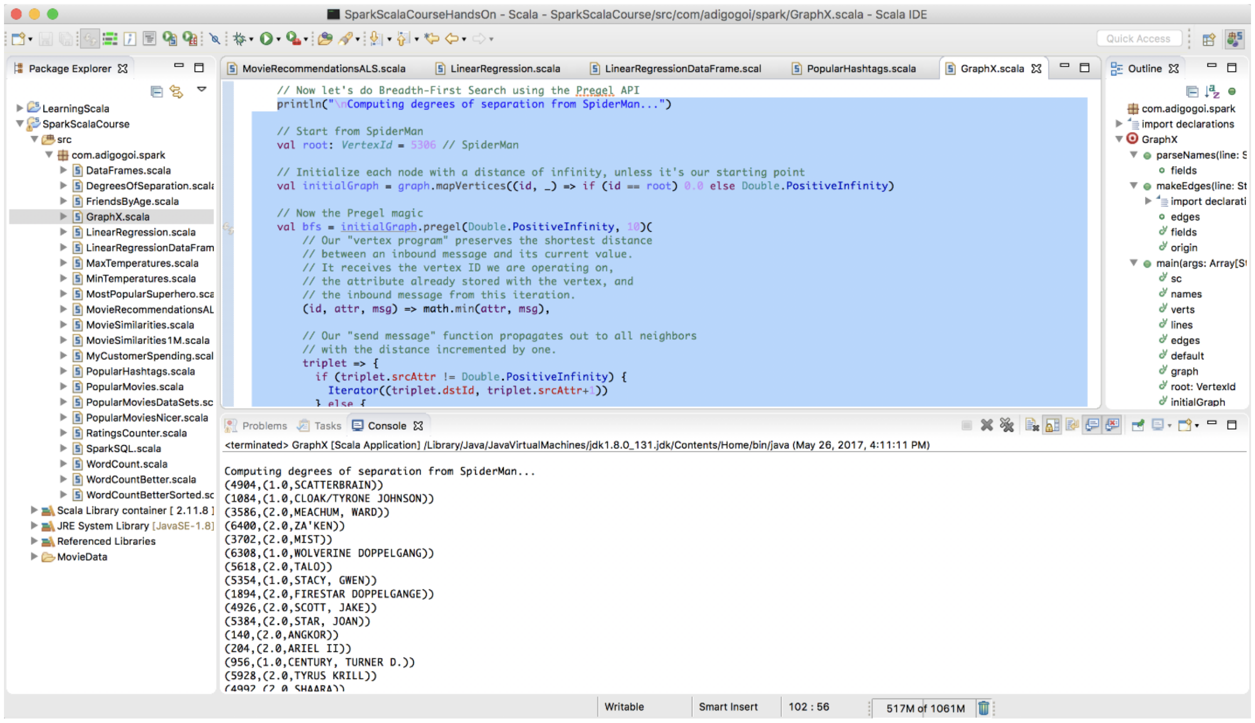


```
    val verts = names.flatMap(parseNames)

    // Build up our edges
    val lines = sc.textFile("../SparkScalaCourse/Datasets/marvel/marvel-graph.txt")
    val edges = lines.flatMap(makeEdges)

    // Build up our graph, and cache it as we're going to do a bunch of stuff with it.
    val default = "Nobody"
    val graph = Graph(verts, edges, default).cache()

    // Find the top 10 most-connected superheroes, using graph.degrees:
    println("\nTop 10 most-connected superheroes:")
    // The join merges the hero names into the output; sorts by total connections on each node.
    graph.degrees.join(verts).sortBy(_._2._1, ascending=false).take(10).foreach(println)


    // Now let's do Breadth-First Search using the Pregel API
    println("\nComputing degrees of separation from SpiderMan...")

    // Start from SpiderMan
    val root: VertexId = 5306 // SpiderMan

    // Initialize each node with a distance of infinity, unless it's our starting point
```

```
<terminated> GraphX [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 26, 2017, 4:11:11 PM)
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

Top 10 most-connected superheroes:

[Stage 1:>    (0 + 2) / 2][Stage 2:>    (0 + 2) / 2][Stage 3:>    (0 + 0) / 2]
                                                                                (859,(3866,CAPTAIN AMERICA))
(5306,(3482,SPIDER-MAN/PETER PAR))
(2664,(3056,IRON MAN/TONY STARK ))
(5716,(2852,THING/BENJAMIN J. GR))
(6306,(2788,WOLVERINE/LOGAN ))
(3805,(2772,MR. FANTASTIC/REED R))
(2557,(2742,HUMAN TORCH/JOHNNY S))
(4898,(2690,SCARLET WITCH/WANDA ))
(5736,(2578,THOR/DR. DONALD BLAK))
(403,(2560,BEAST/HENRY &HANK& P))

Computing degrees of separation from SpiderMan...
```

We can see that Captain America is the most famous hero with 3866 co-appearances with other heroes, followed by Spider-Man, with 3482 co-appearances. This is interesting, as I always thought that Spider-Man would be the most famous superhero of all times. But this is just based on the co-appearances made by each hero, so more data might be required to make a conclusion.

2. **Finding connections between Spider-Man and other heroes** – Having my faith that Spider-Man would be the most famous hero of all-times, I decided to find out the degrees of separation between him and the rest of the Marvel Superheroes. This I did using the afore-mentioned Breadth First Search algorithm of the Pregel API.

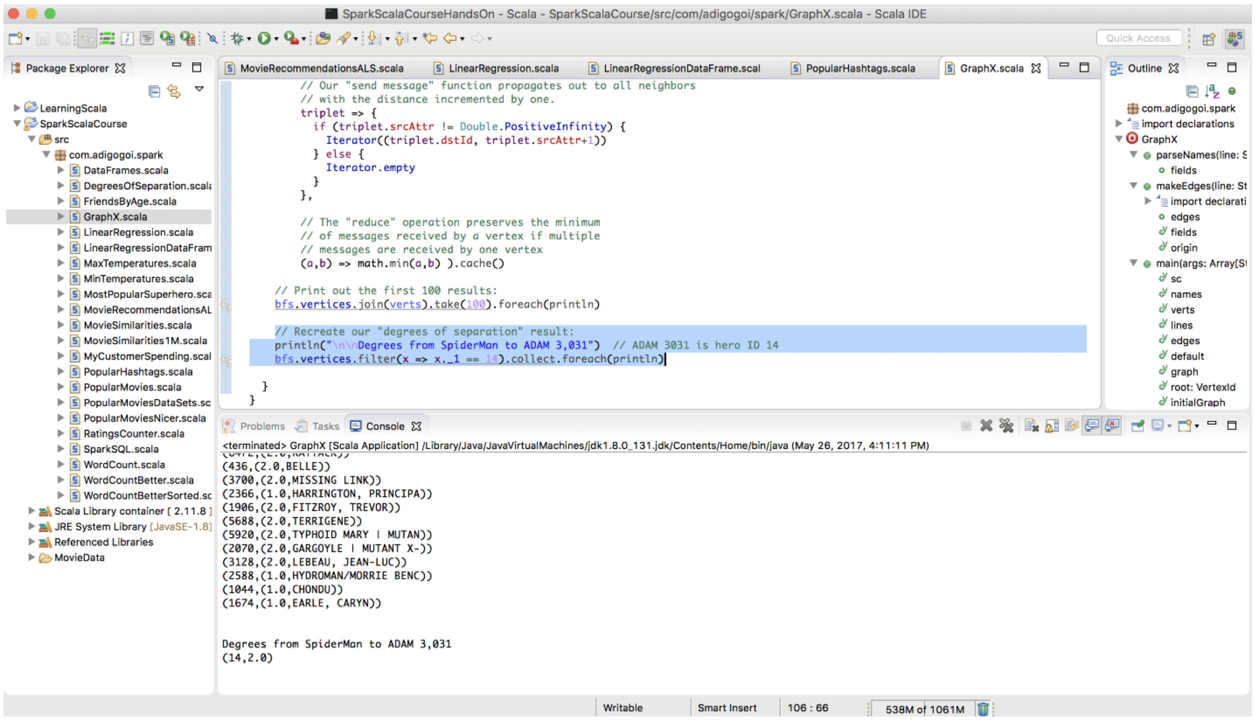


```
    // Now let's do Breadth-First Search using the Pregel API
    println("\nComputing degrees of separation from SpiderMan...")

    // Start from SpiderMan
    val root: VertexId = 5306 // SpiderMan

    // Initialize each node with a distance of infinity, unless it's our starting point
    val initialGraph = graph.mapVertices((id, _) => if (id == root) 0.0 else Double.PositiveInfinity)

    // Now the Pregel magic
    val bfs = initialGraph.pregel(Double.PositiveInfinity, 10)(
        // Our "vertex program" preserves the shortest distance
        // between an inbound message and its current value.
        // It receives the vertex ID we are operating on,
        // the attribute already stored with the vertex, and
        // the inbound message from this iteration.
        (id, attr, msg) => math.min(attr, msg),

        // Our "send message" function propagates out to all neighbors
        // with the distance incremented by one.
        triplet => {
          if (triplet.srcAttr != Double.PositiveInfinity) {
            Iterator((triplet.dstId, triplet.srcAttr+1))
          } else {
```

```
<terminated> GraphX [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 26, 2017, 4:11:11 PM)

Computing degrees of separation from SpiderMan...
(4904,(1.0,SCATTERBRAIN))
(1084,(1.0,CLOAK/TYRONE JOHNSON))
(3586,(2.0,MEACHUM, WARD))
(6400,(2.0,ZA'KEN))
(3702,(2.0,MIST))
(6308,(1.0,WOLVERINE DOPPELGANG))
(5618,(2.0,TALO))
(5354,(1.0,STACY, GWEN))
(1894,(2.0,FIRESTAR DOPPELGANGE))
(4926,(2.0,SCOTT, JAKE))
(5384,(2.0,STAR, JOAN))
(140,(2.0,ANGKOR))
(204,(2.0,ARIEL II))
(956,(1.0,CENTURY, TURNER D.))
(5928,(2.0,TYRUS KRILL))
(4992,(2.0,SHAARA))
```

I noticed that even the lesser-known heroes have just 2 to 3 degrees of separation from Spider-Man. This maybe because many of the heroes Spider-Man has co-appeared with have also co-appeared in many comics with these heroes.

3. **Finding distance between Spider-Man and Adam** – I tried to focus on just one of the heroes who is not so popular in the Marvel Universe, Adam, and decided to find out his distance from Spider-Man. Adam's ID is 14 so that made it easy to find their Degrees of Separation.



```
        // Our "send message" function propagates out to all neighbors
        // with the distance incremented by one.
        triplet => {
          if (triplet.srcAttr != Double.PositiveInfinity) {
            Iterator((triplet.dstId, triplet.srcAttr+1))
          } else {
            Iterator.empty
          }
        },

        // The "reduce" operation preserves the minimum
        // of messages received by a vertex if multiple
        // messages are received by one vertex
        (a,b) => math.min(a,b) ).cache()

    // Print out the first 100 results:
    bfs.vertices.join(verts).take(100).foreach(println)

    // Recreate our "degrees of separation" result:
    println("\n\nDegrees from SpiderMan to ADAM 3,031")  // ADAM 3031 is hero ID 14
    bfs.vertices.filter(x => x._1 == 14).collect.foreach(println)

  }
}
```

```
<terminated> GraphX [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 26, 2017, 4:11:11 PM)
(436,(2.0,BELLE))
(3700,(2.0,MISSING LINK))
(2366,(1.0,HARRINGTON, PRINCIPA))
(1906,(2.0,FITZROY, TREVOR))
(5688,(2.0,TERRIGENE))
(5920,(2.0,TYPHOID MARY | MUTAN))
(2070,(2.0,GARGOYLE | MUTANT X-))
(3128,(2.0,LEBEAU, JEAN-LUC))
(2588,(1.0,HYDROMAN/MORRIE BENC))
(1044,(1.0,CHONDU))
(1674,(1.0,EARLE, CARYN))


Degrees from SpiderMan to ADAM 3,031
(14,2.0)
```

Even Adam is just 2 degrees away from Spider-Man. These superheroes are a tight-knit bunch.

But this project did allow me to implement a network based graph data using Spark's GraphX and Pregel, so it is a success after all.