In [1]:

```
reply_network = spark.read.csv("s3://us-congress-tweets/reply_network.csv", header=
True)
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	
1	application_1572677191612_0002	pyspark	idle	Link (http://ip-1.ec2.internal:20888/proxy/application_1572677191

```
SparkSession available as 'spark'.
```

In [2]:

```
print(1)
```

```
1
```

In [3]:

```
reply_network_id_only = reply_network.select("src_id", "dst_id")
```

In [4]:

```
reply_network_id_only.show(5)
```

```
+------------------+------------------+
|            src_id|            dst_id|
+------------------+------------------+
|1047536930651611137|1047536497052844032|
|1047537000385929216|1047529220065447936|
|1047537019386093568|1047490615829827585|
|1047537318461026304|1047504406646808576|
|1047537380981465088|1047517885260750853|
+------------------+------------------+
only showing top 5 rows
```

In [5]:

```
edges = reply_network_id_only

edges = edges.selectExpr("src_id as src", "dst_id as dst")

edges.show()
```

```
+------------------+------------------+
|               src|               dst|
+------------------+------------------+
|1047536930651611137|1047536497052844032|
|1047537000385929216|1047529220065447936|
|1047537019386093568|1047490615829827585|
|1047537318461026304|1047504406646808576|
|1047537380981465088|1047517885260750853|
|1047537448295813120|1047465256023445504|
|1047537571146944512|1047504990011572229|
|1047537751049064448|1047534440757579777|
|1047538150959136772|1047529230203047937|
|1047538603008696322|1047132430098927617|
|1047538630573678592|1047508807675531264|
|1047538803005636608|1047537257727504384|
|1047539317055377410|1047538522289258499|
|1047539502946967553|1047519678749442048|
|1047539625361887234|1047516105567363073|
|1047539874662928385|1047531068738293760|
|1047540275818651649|1047503978534264834|
|1047540523312078849|1046876489113907201|
|1047540591695777792|1047539231181213696|
|1047540640588996609|1047540255409229825|
+------------------+------------------+
only showing top 20 rows
```

In [6]:

```
edges.show()
```

```
+-------------------+-------------------+
|                src|                dst|
+-------------------+-------------------+
|1047536930651611137|1047536497052844032|
|1047537000385929216|1047529220065447936|
|1047537019386093568|1047490615829827585|
|1047537318461026304|1047504406646808576|
|1047537380981465088|1047517885260750853|
|1047537448295813120|1047465256023445504|
|1047537571146944512|1047504990011572229|
|1047537751049064448|1047534440757579777|
|1047538150959136772|1047529230203047937|
|1047538603008696322|1047132430098927617|
|1047538630573678592|1047508807675531264|
|1047538803005636608|1047537257727504384|
|1047539317055377410|1047538522289258499|
|1047539502946967553|1047519678749442048|
|1047539625361887234|1047516105567363073|
|1047539874662928385|1047531068738293760|
|1047540275818651649|1047503978534264834|
|1047540523312078849|1046876489113907201|
|1047540591695777792|1047539231181213696|
|1047540640588996609|1047540255409229825|
+-------------------+-------------------+
only showing top 20 rows
```

In [7]:

```
df_1_new = edges.select('src').distinct()
df_2_new = edges.select('dst')

vertices = df_1_new.union(df_2_new).alias('id')

vertices = vertices.selectExpr("src as id")

vertices.show()
```

```
+------------------+
|                id|
+------------------+
|1096032827240517632|
|1096076620157407232|
|1096085353549713410|
|1096109760691294208|
|1096124983192027136|
|1096146262380941313|
|1096164924743544832|
|1096211185463111680|
|1096430689057193984|
|1096431283092246529|
|1096531707887521792|
|1096542032363810816|
|1096542034813304832|
|1096581982442536960|
|1096683607471591424|
|1096732094976409605|
|1096791489173176325|
|1096872220830224384|
|1096895204240568321|
|1096979084343762949|
+------------------+
only showing top 20 rows
```

In [8]:

```
vertices.distinct().count()
```

```
76510188
```

In [9]:

```
from graphframes import *
from graphframes.lib import Pregel
sc.setCheckpointDir("hdfs:///tmp/graphframes_checkpoint") # this is needed for any
 GraphFrames operation
```

In [10]:

```
# edges.show()
# edges = edges.selectExpr("src_id as src")
# edges = edges.selectExpr("dst_id as dst")


# your network construction code here
graph = GraphFrame(vertices, edges)
```

In [11]:

```
# What are the top replied to tweets? (show 20)
vertices_in = graph.inDegrees
vertices_in.orderBy("inDegree",ascending=False).show(20)
```

```
+------------------+--------+
|                id|inDegree|
+------------------+--------+
|1157787985041088513|   94351|
|1048314564826292227|   76396|
|1111289977143545856|   68172|
|1155949756792725510|   65241|
|1137060666223878144|   57764|
|1062461047892787204|   53767|
|1158036816089497601|   50059|
|1144730911889428480|   45905|
|1155949605147648006|   44810|
|1098312693436596226|   41836|
|1150408691713265665|   41825|
|1129831615952236546|   41556|
|1150859069084905472|   39020|
|1144078421670150144|   38687|
|1155132215208161281|   38572|
|1088141172638400512|   37102|
|1155469517092470784|   36961|
|1168938037071482881|   36728|
|1154161356171599877|   36552|
|1131740851909083137|   36386|
+------------------+--------+
only showing top 20 rows
```

In [12]:

```
# How many graphs in the reply network? (Hint: use connectedComponents function)
# graph.connectedComponents().show()
```

In [13]:

```

```

In [ ]:

```
```

In [14]:

```
# Spark Version
spark.version
```

'2.4.4'

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
import pyspark.sql.functions as F # needed for defining literals below and the sum
(...) function

# Now, design and execute a Pregel program that will calculate the longest paths fo
r all reply graphs in the network. Explain your design.

ranks = graph.pregel \
             .setMaxIter(5) \
             .withVertexColumn("rank",
                               F.lit(0),   # all the initial value will be 0
                               F.coalesce( Pregel.msg() + F.lit(1), F.lit(0) )
                              ) \
             .sendMsgToDst(   Pregel.src("rank")   )\
             .aggMsgs(F.sum(Pregel.msg())).run()
ranks.show()
```

In [ ]:

```
graph.connectedComponents().count()
```

In [ ]:

```
# What is the average longest path length for all reply graphs in the network?
graph.select("max(dist)").groupby().avg().show();
```

In [ ]:

```
```