IERG 4330/ESTR 4316/IEMS 5730 Spring 2022 Homework 5

Release date: Apr 20, 2022

Due date: 11:59:00 pm, May 10, 2022

No late homework will be accepted!

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

Signed (Student [illegible]) Date: 10-5-22

Name Chan Kai Tin SID 1155124983

Submission notice:

- Submit your homework via the elearning system

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

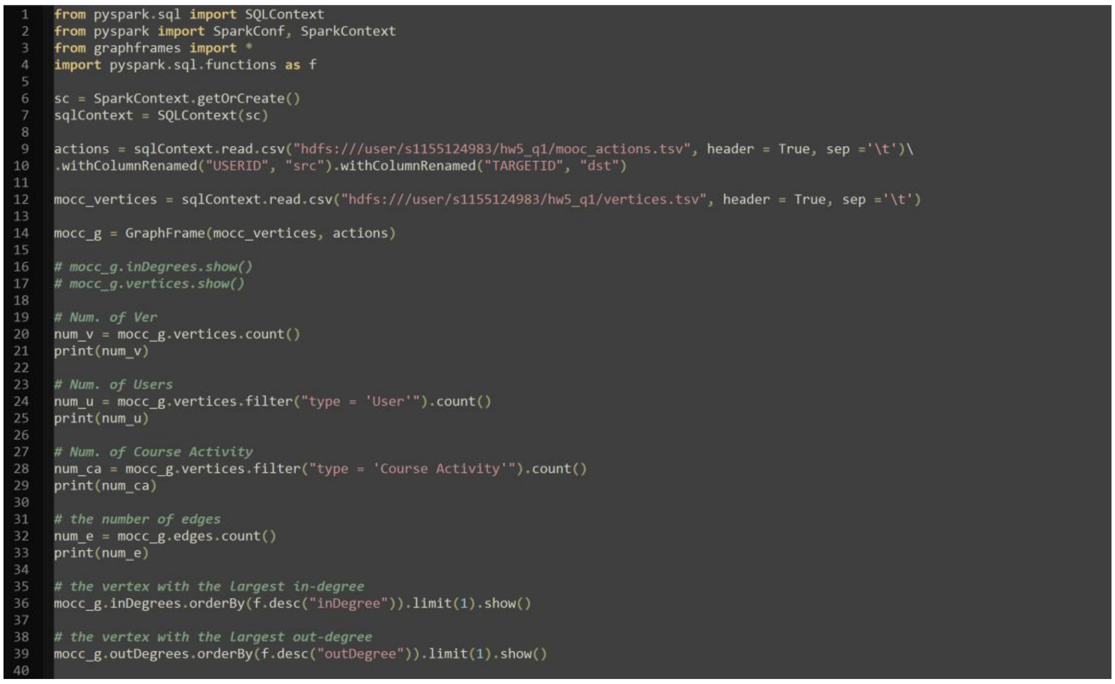
Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value and justify any assumptions you make. You will be graded not only on whether your answer is correct but also on whether you have done an intelligent analysis.

Q1.

a)

Code:



```python
from pyspark.sql import SQLContext
from pyspark import SparkConf, SparkContext
from graphframes import *
import pyspark.sql.functions as f

sc = SparkContext.getOrCreate()
sqlContext = SQLContext(sc)

actions = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/mooc_actions.tsv", header = True, sep ='\t')\
.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

mocc_vertices = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/vertices.tsv", header = True, sep ='\t')

mocc_g = GraphFrame(mocc_vertices, actions)

# mocc_g.inDegrees.show()
# mocc_g.vertices.show()

# Num. of Ver
num_v = mocc_g.vertices.count()
print(num_v)

# Num. of Users
num_u = mocc_g.vertices.filter("type = 'User'").count()
print(num_u)

# Num. of Course Activity
num_ca = mocc_g.vertices.filter("type = 'Course Activity'").count()
print(num_ca)

# the number of edges
num_e = mocc_g.edges.count()
print(num_e)

# the vertex with the largest in-degree
mocc_g.inDegrees.orderBy(f.desc("inDegree")).limit(1).show()

# the vertex with the largest out-degree
mocc_g.outDegrees.orderBy(f.desc("outDegree")).limit(1).show()

```

Output for (i) – (vi):

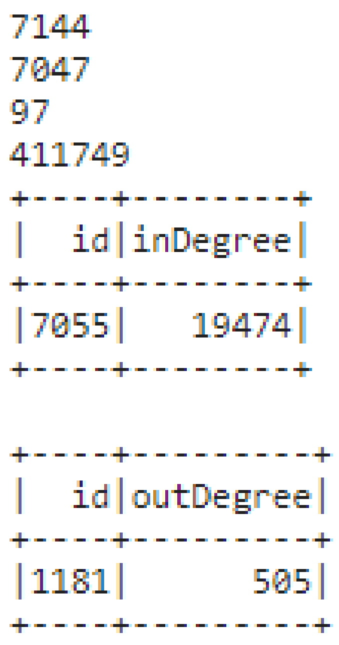
i: 7144

ii: 7047

iii: 97

iv: 411749

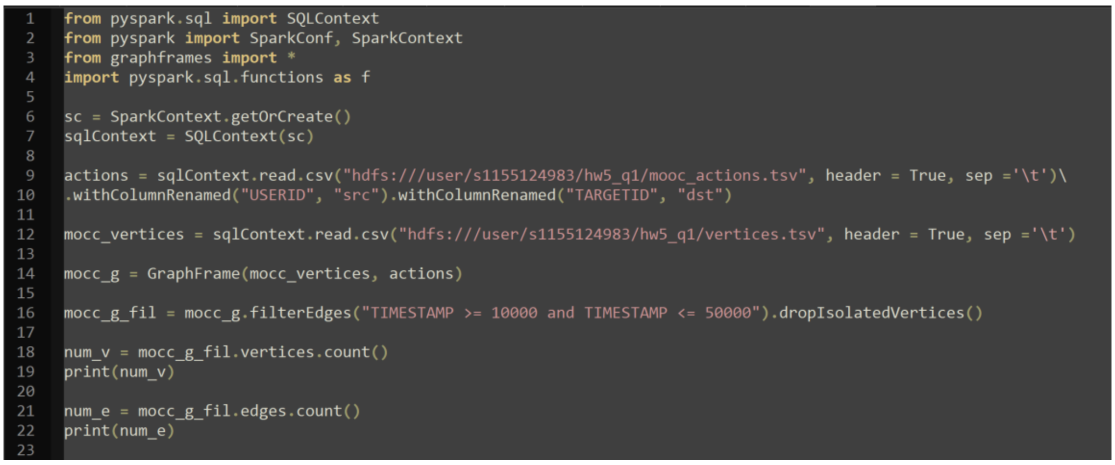
v and vi shown on the graph



```
7144
7047
97
411749
+----+--------+
|  id|inDegree|
+----+--------+
|7055|   19474|
+----+--------+

+----+---------+
|  id|outDegree|
+----+---------+
|1181|      505|
+----+---------+
```

b.)

Code:

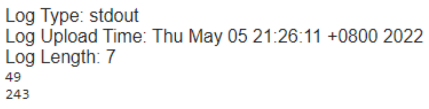


```python
from pyspark.sql import SQLContext
from pyspark import SparkConf, SparkContext
from graphframes import *
import pyspark.sql.functions as f

sc = SparkContext.getOrCreate()
sqlContext = SQLContext(sc)

actions = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/mooc_actions.tsv", header = True, sep ='\t')\
.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

mocc_vertices = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/vertices.tsv", header = True, sep ='\t')

mocc_g = GraphFrame(mocc_vertices, actions)

mocc_g_fil = mocc_g.filterEdges("TIMESTAMP >= 10000 and TIMESTAMP <= 50000").dropIsolatedVertices()

num_v = mocc_g_fil.vertices.count()
print(num_v)

num_e = mocc_g_fil.edges.count()
print(num_e)

```

Output:

Number of nodes = 49

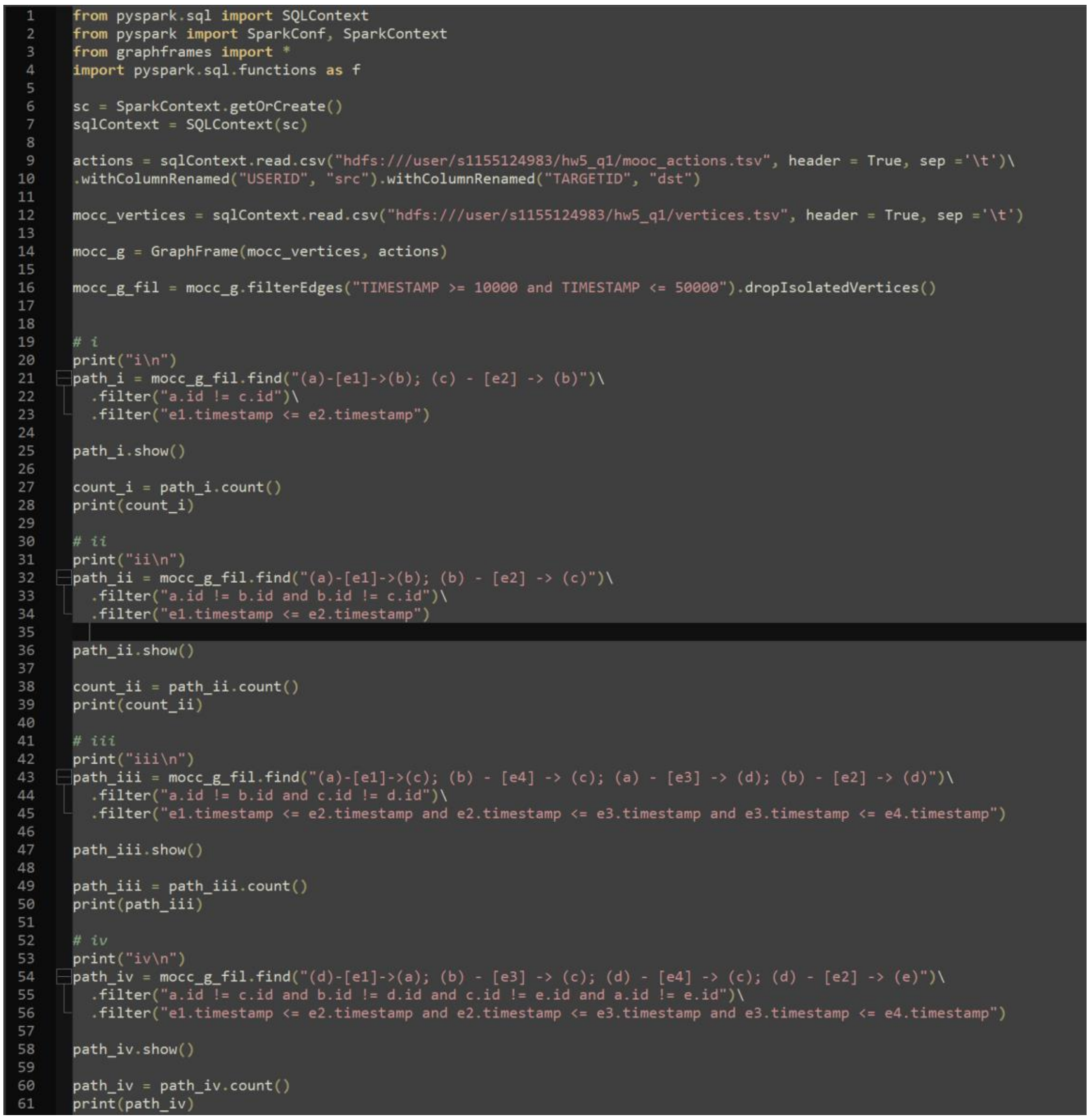
Number of edges = 243



```
Log Type: stdout
Log Upload Time: Thu May 05 21:26:11 +0800 2022
Log Length: 7
49
243
```

c.)

Code:



```python
from pyspark.sql import SQLContext
from pyspark import SparkConf, SparkContext
from graphframes import *
import pyspark.sql.functions as f

sc = SparkContext.getOrCreate()
sqlContext = SQLContext(sc)

actions = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/mooc_actions.tsv", header = True, sep ='\t')\
.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

mocc_vertices = sqlContext.read.csv("hdfs:///user/s1155124983/hw5_q1/vertices.tsv", header = True, sep ='\t')

mocc_g = GraphFrame(mocc_vertices, actions)

mocc_g_fil = mocc_g.filterEdges("TIMESTAMP >= 10000 and TIMESTAMP <= 50000").dropIsolatedVertices()


# i
print("i\n")
path_i = mocc_g_fil.find("(a)-[e1]->(b); (c) - [e2] -> (b)")\
  .filter("a.id != c.id")\
  .filter("e1.timestamp <= e2.timestamp")

path_i.show()

count_i = path_i.count()
print(count_i)

# ii
print("ii\n")
path_ii = mocc_g_fil.find("(a)-[e1]->(b); (b) - [e2] -> (c)")\
  .filter("a.id != b.id and b.id != c.id")\
  .filter("e1.timestamp <= e2.timestamp")

path_ii.show()

count_ii = path_ii.count()
print(count_ii)

# iii
print("iii\n")
path_iii = mocc_g_fil.find("(a)-[e1]->(c); (b) - [e4] -> (c); (a) - [e3] -> (d); (b) - [e2] -> (d)")\
  .filter("a.id != b.id and c.id != d.id")\
  .filter("e1.timestamp <= e2.timestamp and e2.timestamp <= e3.timestamp and e3.timestamp <= e4.timestamp")

path_iii.show()

path_iii = path_iii.count()
print(path_iii)

# iv
print("iv\n")
path_iv = mocc_g_fil.find("(d)-[e1]->(a); (b) - [e3] -> (c); (d) - [e4] -> (c); (d) - [e2] -> (e)")\
  .filter("a.id != c.id and b.id != d.id and c.id != e.id and a.id != e.id")\
  .filter("e1.timestamp <= e2.timestamp and e2.timestamp <= e3.timestamp and e3.timestamp <= e4.timestamp")

path_iv.show()

path_iv = path_iv.count()
print(path_iv)
```

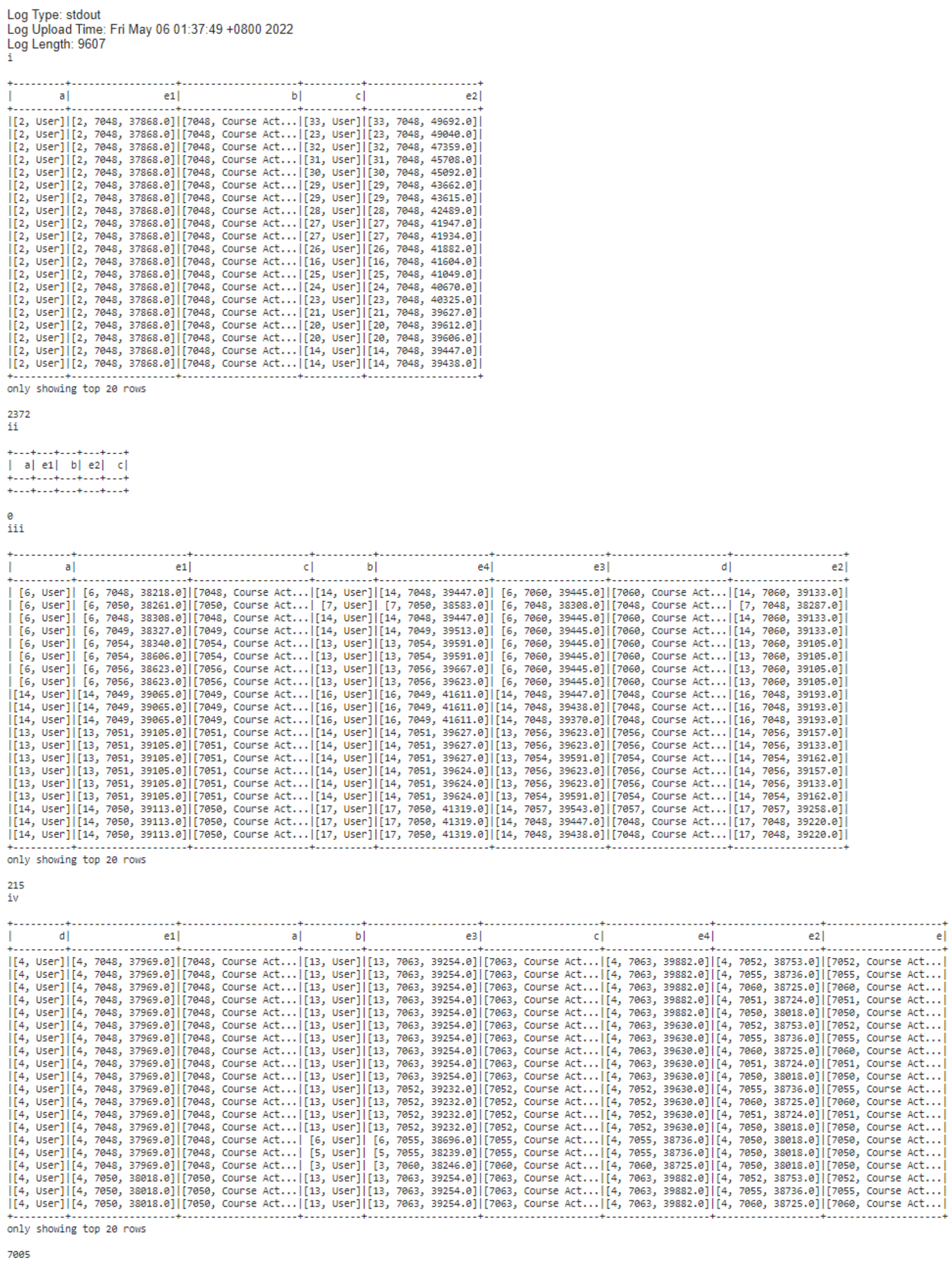
Output:

i) 2372

ii) 0

iii) 215

iv) 7005

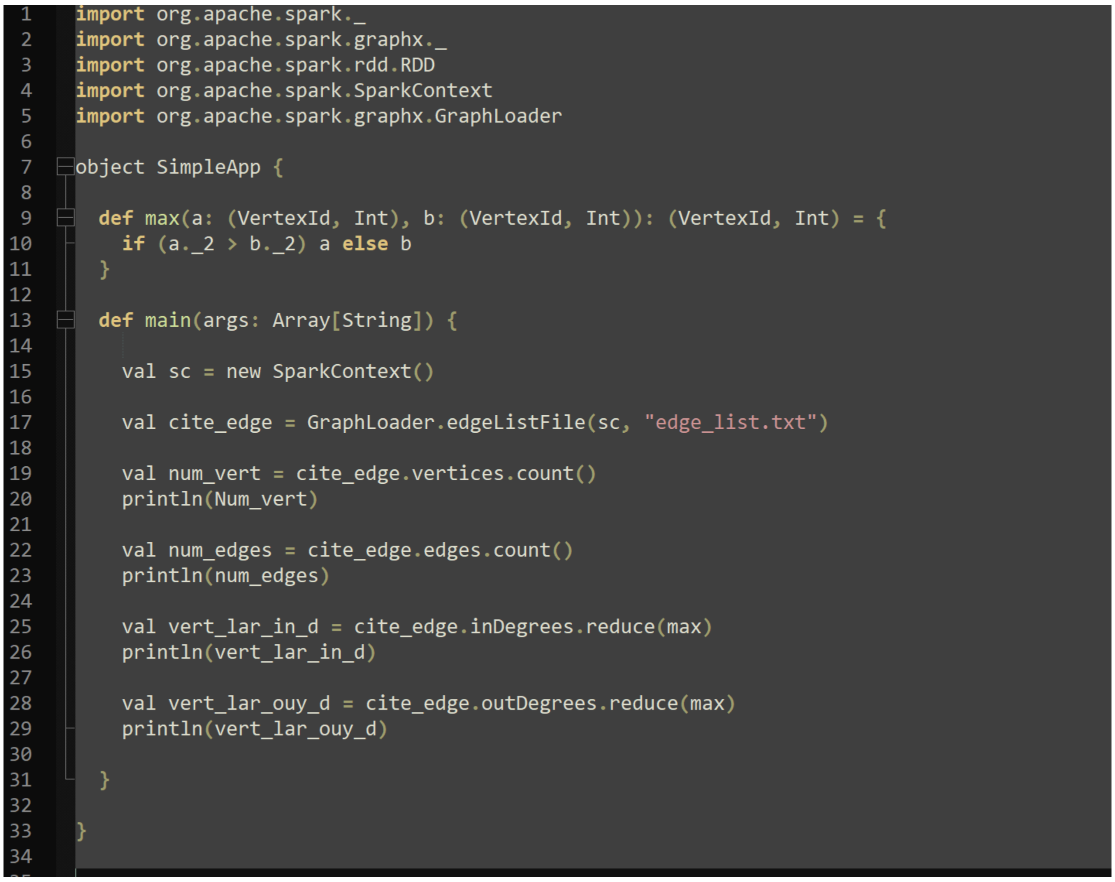


```
Log Type: stdout
Log Upload Time: Fri May 06 01:37:49 +0800 2022
Log Length: 9607
i

+---------+------------------+-------------------+----------+------------------+
|        a|                e1|                  b|         c|                e2|
+---------+------------------+-------------------+----------+------------------+
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[33, User]|[33, 7048, 49692.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[23, User]|[23, 7048, 49040.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[32, User]|[32, 7048, 47359.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[31, User]|[31, 7048, 45708.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[30, User]|[30, 7048, 45092.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[29, User]|[29, 7048, 43662.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[29, User]|[29, 7048, 43615.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[28, User]|[28, 7048, 42489.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[27, User]|[27, 7048, 41947.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[27, User]|[27, 7048, 41934.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[26, User]|[26, 7048, 41882.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[16, User]|[16, 7048, 41604.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[25, User]|[25, 7048, 41049.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[24, User]|[24, 7048, 40670.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[23, User]|[23, 7048, 40325.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[21, User]|[21, 7048, 39627.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[20, User]|[20, 7048, 39612.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[20, User]|[20, 7048, 39606.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[14, User]|[14, 7048, 39447.0]|
|[2, User]|[2, 7048, 37868.0]|[7048, Course Act...|[14, User]|[14, 7048, 39438.0]|
+---------+------------------+-------------------+----------+------------------+
only showing top 20 rows

2372
ii

+---+---+---+---+---+
|  a| e1|  b| e2|  c|
+---+---+---+---+---+
+---+---+---+---+---+

0
iii

+----------+-------------------+--------------------+----------+-------------------+-------------------+--------------------+-------------------+
|         a|                 e1|                   c|         b|                 e4|                 e3|                   d|                 e2|
+----------+-------------------+--------------------+----------+-------------------+-------------------+--------------------+-------------------+
| [6, User]| [6, 7048, 38218.0]|[7048, Course Act...|[14, User]|[14, 7048, 39447.0]| [6, 7060, 39445.0]|[7060, Course Act...|[14, 7060, 39133.0]|
| [6, User]| [6, 7050, 38261.0]|[7050, Course Act...| [7, User]| [7, 7050, 38583.0]| [6, 7048, 38308.0]|[7048, Course Act...| [7, 7048, 38287.0]|
| [6, User]| [6, 7048, 38308.0]|[7048, Course Act...|[14, User]|[14, 7048, 39447.0]| [6, 7060, 39445.0]|[7060, Course Act...|[14, 7060, 39133.0]|
| [6, User]| [6, 7049, 38327.0]|[7049, Course Act...|[14, User]|[14, 7049, 39513.0]| [6, 7060, 39445.0]|[7060, Course Act...|[14, 7060, 39133.0]|
| [6, User]| [6, 7054, 38340.0]|[7054, Course Act...|[13, User]|[13, 7054, 39591.0]| [6, 7060, 39445.0]|[7060, Course Act...|[13, 7060, 39105.0]|
| [6, User]| [6, 7054, 38606.0]|[7054, Course Act...|[13, User]|[13, 7054, 39591.0]| [6, 7060, 39445.0]|[7060, Course Act...|[13, 7060, 39105.0]|
| [6, User]| [6, 7056, 38623.0]|[7056, Course Act...|[13, User]|[13, 7056, 39667.0]| [6, 7060, 39445.0]|[7060, Course Act...|[13, 7060, 39105.0]|
| [6, User]| [6, 7056, 38623.0]|[7056, Course Act...|[13, User]|[13, 7056, 39623.0]| [6, 7060, 39445.0]|[7060, Course Act...|[13, 7060, 39105.0]|
|[14, User]|[14, 7049, 39065.0]|[7049, Course Act...|[16, User]|[16, 7049, 41611.0]|[14, 7048, 39447.0]|[7048, Course Act...|[16, 7048, 39193.0]|
|[14, User]|[14, 7049, 39065.0]|[7049, Course Act...|[16, User]|[16, 7049, 41611.0]|[14, 7048, 39438.0]|[7048, Course Act...|[16, 7048, 39193.0]|
|[14, User]|[14, 7049, 39065.0]|[7049, Course Act...|[16, User]|[16, 7049, 41611.0]|[14, 7048, 39370.0]|[7048, Course Act...|[16, 7048, 39193.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39627.0]|[13, 7056, 39623.0]|[7056, Course Act...|[14, 7056, 39157.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39627.0]|[13, 7056, 39623.0]|[7056, Course Act...|[14, 7056, 39133.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39627.0]|[13, 7054, 39591.0]|[7054, Course Act...|[14, 7054, 39162.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39624.0]|[13, 7056, 39623.0]|[7056, Course Act...|[14, 7056, 39157.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39624.0]|[13, 7056, 39623.0]|[7056, Course Act...|[14, 7056, 39133.0]|
|[13, User]|[13, 7051, 39105.0]|[7051, Course Act...|[14, User]|[14, 7051, 39624.0]|[13, 7054, 39591.0]|[7054, Course Act...|[14, 7054, 39162.0]|
|[14, User]|[14, 7050, 39113.0]|[7050, Course Act...|[17, User]|[17, 7050, 41319.0]|[14, 7057, 39543.0]|[7057, Course Act...|[17, 7057, 39258.0]|
|[14, User]|[14, 7050, 39113.0]|[7050, Course Act...|[17, User]|[17, 7050, 41319.0]|[14, 7048, 39447.0]|[7048, Course Act...|[17, 7048, 39220.0]|
|[14, User]|[14, 7050, 39113.0]|[7050, Course Act...|[17, User]|[17, 7050, 41319.0]|[14, 7048, 39438.0]|[7048, Course Act...|[17, 7048, 39220.0]|
+----------+-------------------+--------------------+----------+-------------------+-------------------+--------------------+-------------------+
only showing top 20 rows

215
iv

+---------+------------------+--------------------+----------+-------------------+--------------------+------------------+------------------+--------------------+
|        d|                e1|                   a|         b|                 e3|                   c|                e4|                e2|                   e|
+---------+------------------+--------------------+----------+-------------------+--------------------+------------------+------------------+--------------------+
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7052, 38753.0]|[7052, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7055, 38736.0]|[7055, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7060, 38725.0]|[7060, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7051, 38724.0]|[7051, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39630.0]|[4, 7052, 38753.0]|[7052, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39630.0]|[4, 7055, 38736.0]|[7055, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39630.0]|[4, 7060, 38725.0]|[7060, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39630.0]|[4, 7051, 38724.0]|[7051, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39630.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7052, 39232.0]|[7052, Course Act...|[4, 7052, 39630.0]|[4, 7055, 38736.0]|[7055, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7052, 39232.0]|[7052, Course Act...|[4, 7052, 39630.0]|[4, 7060, 38725.0]|[7060, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7052, 39232.0]|[7052, Course Act...|[4, 7052, 39630.0]|[4, 7051, 38724.0]|[7051, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...|[13, User]|[13, 7052, 39232.0]|[7052, Course Act...|[4, 7052, 39630.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...| [6, User]| [6, 7055, 38696.0]|[7055, Course Act...|[4, 7055, 38736.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...| [5, User]| [5, 7055, 38239.0]|[7055, Course Act...|[4, 7055, 38736.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7048, 37969.0]|[7048, Course Act...| [3, User]| [3, 7060, 38246.0]|[7060, Course Act...|[4, 7060, 38725.0]|[4, 7050, 38018.0]|[7050, Course Act...|
|[4, User]|[4, 7050, 38018.0]|[7050, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7052, 38753.0]|[7052, Course Act...|
|[4, User]|[4, 7050, 38018.0]|[7050, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7055, 38736.0]|[7055, Course Act...|
|[4, User]|[4, 7050, 38018.0]|[7050, Course Act...|[13, User]|[13, 7063, 39254.0]|[7063, Course Act...|[4, 7063, 39882.0]|[4, 7060, 38725.0]|[7060, Course Act...|
+---------+------------------+--------------------+----------+-------------------+--------------------+------------------+------------------+--------------------+
only showing top 20 rows

7005
```

Q2

a.)

Code:



```scala
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
import org.apache.spark.SparkContext
import org.apache.spark.graphx.GraphLoader

object SimpleApp {

  def max(a: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
    if (a._2 > b._2) a else b
  }

  def main(args: Array[String]) {

    val sc = new SparkContext()

    val cite_edge = GraphLoader.edgeListFile(sc, "edge_list.txt")

    val num_vert = cite_edge.vertices.count()
    println(Num_vert)

    val num_edges = cite_edge.edges.count()
    println(num_edges)

    val vert_lar_in_d = cite_edge.inDegrees.reduce(max)
    println(vert_lar_in_d)

    val vert_lar_ouy_d = cite_edge.outDegrees.reduce(max)
    println(vert_lar_ouy_d)

  }

}
```

Output:

The answers follow the order of variables:

num_vert = 169343

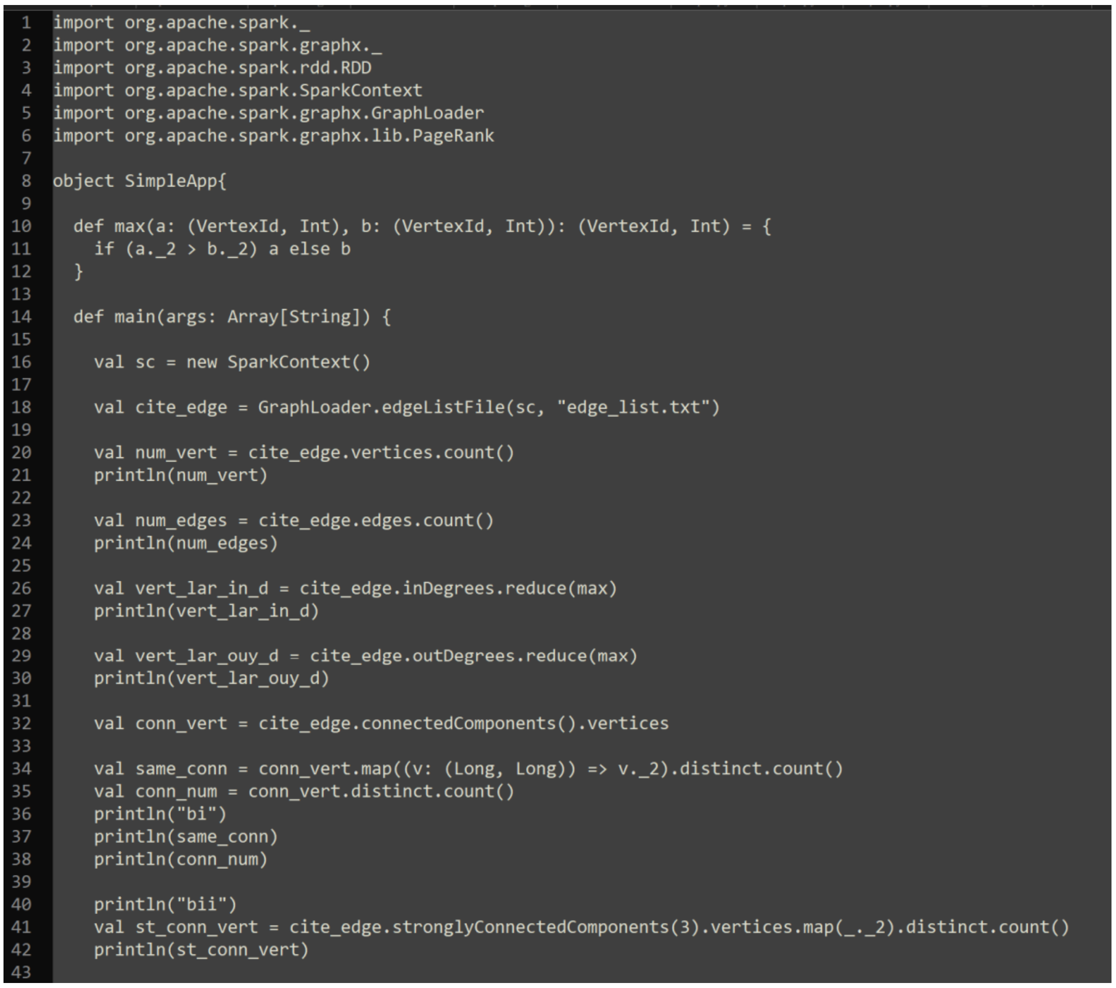
num_edges = 1166243

vert_lar_in_d = 1353

vert_lar_ouy_d = 72253

b.)

Code:



```
1  import org.apache.spark._
2  import org.apache.spark.graphx._
3  import org.apache.spark.rdd.RDD
4  import org.apache.spark.SparkContext
5  import org.apache.spark.graphx.GraphLoader
6  import org.apache.spark.graphx.lib.PageRank
7
8  object SimpleApp{
9
10   def max(a: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
11     if (a._2 > b._2) a else b
12   }
13
14   def main(args: Array[String]) {
15
16     val sc = new SparkContext()
17
18     val cite_edge = GraphLoader.edgeListFile(sc, "edge_list.txt")
19
20     val num_vert = cite_edge.vertices.count()
21     println(num_vert)
22
23     val num_edges = cite_edge.edges.count()
24     println(num_edges)
25
26     val vert_lar_in_d = cite_edge.inDegrees.reduce(max)
27     println(vert_lar_in_d)
28
29     val vert_lar_ouy_d = cite_edge.outDegrees.reduce(max)
30     println(vert_lar_ouy_d)
31
32     val conn_vert = cite_edge.connectedComponents().vertices
33
34     val same_conn = conn_vert.map((v: (Long, Long)) => v._2).distinct.count()
35     val conn_num = conn_vert.distinct.count()
36     println("bi")
37     println(same_conn)
38     println(conn_num)
39
40     println("bii")
41     val st_conn_vert = cite_edge.stronglyConnectedComponents(3).vertices.map(_._2).distinct.count()
42     println(st_conn_vert)
43
```

i.)

Connected components exist in the citation network = 1

Vertices exist in the largest connected component = 169343

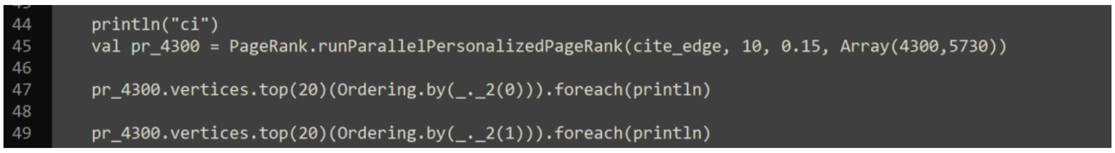
ii.)

Strongly connected components exist in the citation network = 169343

#In line 41 the stronglyConnectedComponents function is changed to 1 instead of 3 because of the driver-memory problem.

c)

Code:

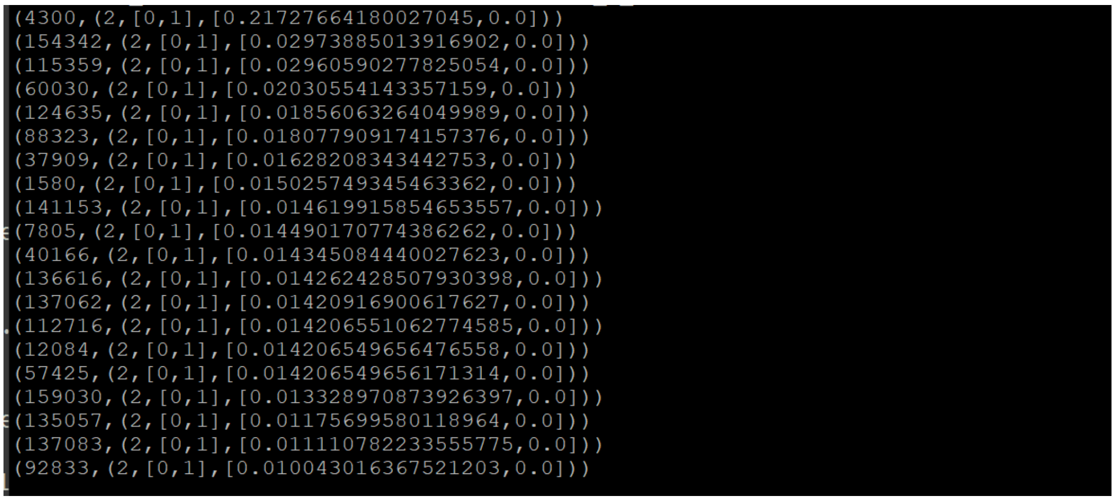


```
    println("ci")
    val pr_4300 = PageRank.runParallelPersonalizedPageRank(cite_edge, 10, 0.15, Array(4300,5730))

    pr_4300.vertices.top(20)(Ordering.by(_._2(0))).foreach(println)

    pr_4300.vertices.top(20)(Ordering.by(_._2(1))).foreach(println)
```

Output:

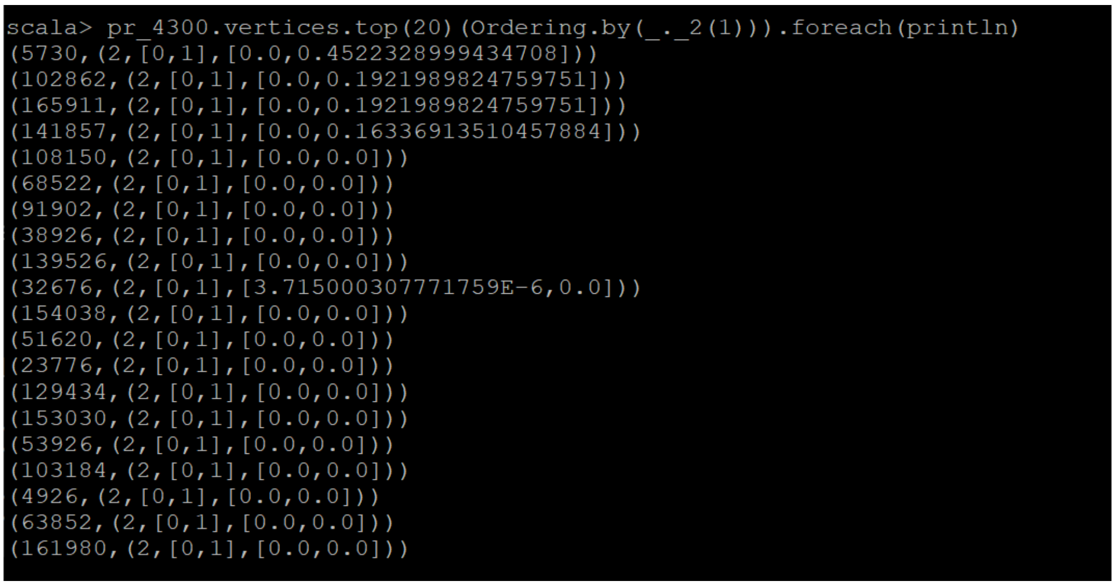
i.)

4330:



```
(4300,(2,[0,1],[0.21727664180027045,0.0]))
(154342,(2,[0,1],[0.02973885013916902,0.0]))
(115359,(2,[0,1],[0.02960590277825054,0.0]))
(60030,(2,[0,1],[0.02030554143357159,0.0]))
(124635,(2,[0,1],[0.01856063264049989,0.0]))
(88323,(2,[0,1],[0.018077909174157376,0.0]))
(37909,(2,[0,1],[0.01628208343442753,0.0]))
(1580,(2,[0,1],[0.015025749345463362,0.0]))
(141153,(2,[0,1],[0.014619915854653557,0.0]))
(7805,(2,[0,1],[0.014490170774386262,0.0]))
(40166,(2,[0,1],[0.014345084440027623,0.0]))
(136616,(2,[0,1],[0.014262428507930398,0.0]))
(137062,(2,[0,1],[0.01420916900617627,0.0]))
(112716,(2,[0,1],[0.014206551062774585,0.0]))
(12084,(2,[0,1],[0.014206549656476558,0.0]))
(57425,(2,[0,1],[0.014206549656171314,0.0]))
(159030,(2,[0,1],[0.013328970873926397,0.0]))
(135057,(2,[0,1],[0.01175699580118964,0.0]))
(137083,(2,[0,1],[0.011110782233555775,0.0]))
(92833,(2,[0,1],[0.010043016367521203,0.0]))
```

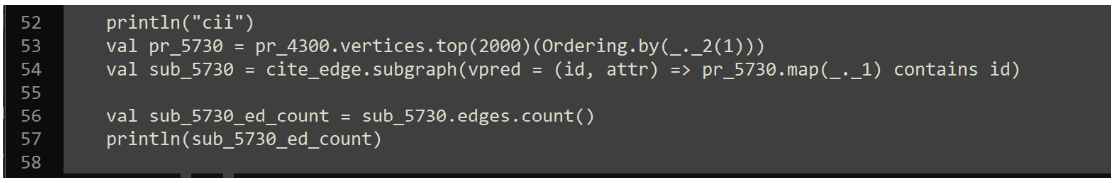
5730:



```
scala> pr_4300.vertices.top(20)(Ordering.by(_._2(1))).foreach(println)
(5730,(2,[0,1],[0.0,0.4522328999434708]))
(102862,(2,[0,1],[0.0,0.1921989824759751]))
(165911,(2,[0,1],[0.0,0.1921989824759751]))
(141857,(2,[0,1],[0.0,0.16336913510457884]))
(108150,(2,[0,1],[0.0,0.0]))
(68522,(2,[0,1],[0.0,0.0]))
(91902,(2,[0,1],[0.0,0.0]))
(38926,(2,[0,1],[0.0,0.0]))
(139526,(2,[0,1],[0.0,0.0]))
(32676,(2,[0,1],[3.715000307771759E-6,0.0]))
(154038,(2,[0,1],[0.0,0.0]))
(51620,(2,[0,1],[0.0,0.0]))
(23776,(2,[0,1],[0.0,0.0]))
(129434,(2,[0,1],[0.0,0.0]))
(153030,(2,[0,1],[0.0,0.0]))
(53926,(2,[0,1],[0.0,0.0]))
(103184,(2,[0,1],[0.0,0.0]))
(4926,(2,[0,1],[0.0,0.0]))
(63852,(2,[0,1],[0.0,0.0]))
(161980,(2,[0,1],[0.0,0.0]))
```

ii)

Code:

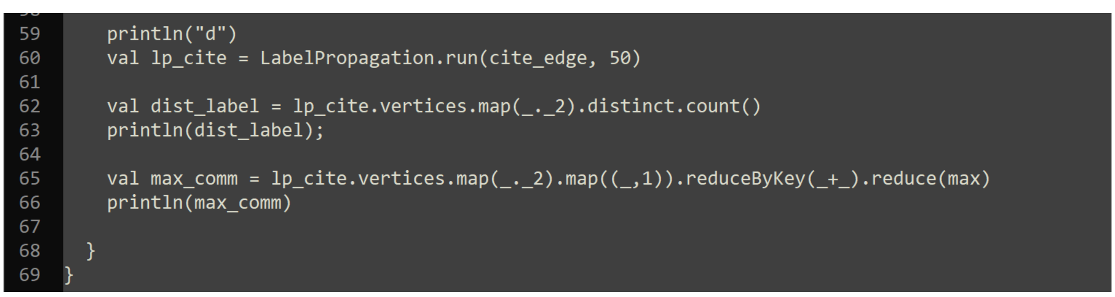


```
    println("cii")
    val pr_5730 = pr_4300.vertices.top(2000)(Ordering.by(_._2(1)))
    val sub_5730 = cite_edge.subgraph(vpred = (id, attr) => pr_5730.map(_._1) contains id)

    val sub_5730_ed_count = sub_5730.edges.count()
    println(sub_5730_ed_count)
```

number of edges = 155

d.)

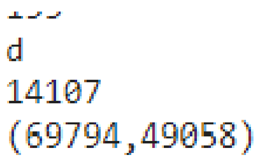
Code:



```
    println("d")
    val lp_cite = LabelPropagation.run(cite_edge, 50)

    val dist_label = lp_cite.vertices.map(_._2).distinct.count()
    println(dist_label);

    val max_comm = lp_cite.vertices.map(_._2).map((_,1)).reduceByKey(_+_).reduce(max)
    println(max_comm)

  }
}
```

Number of communities = 14107

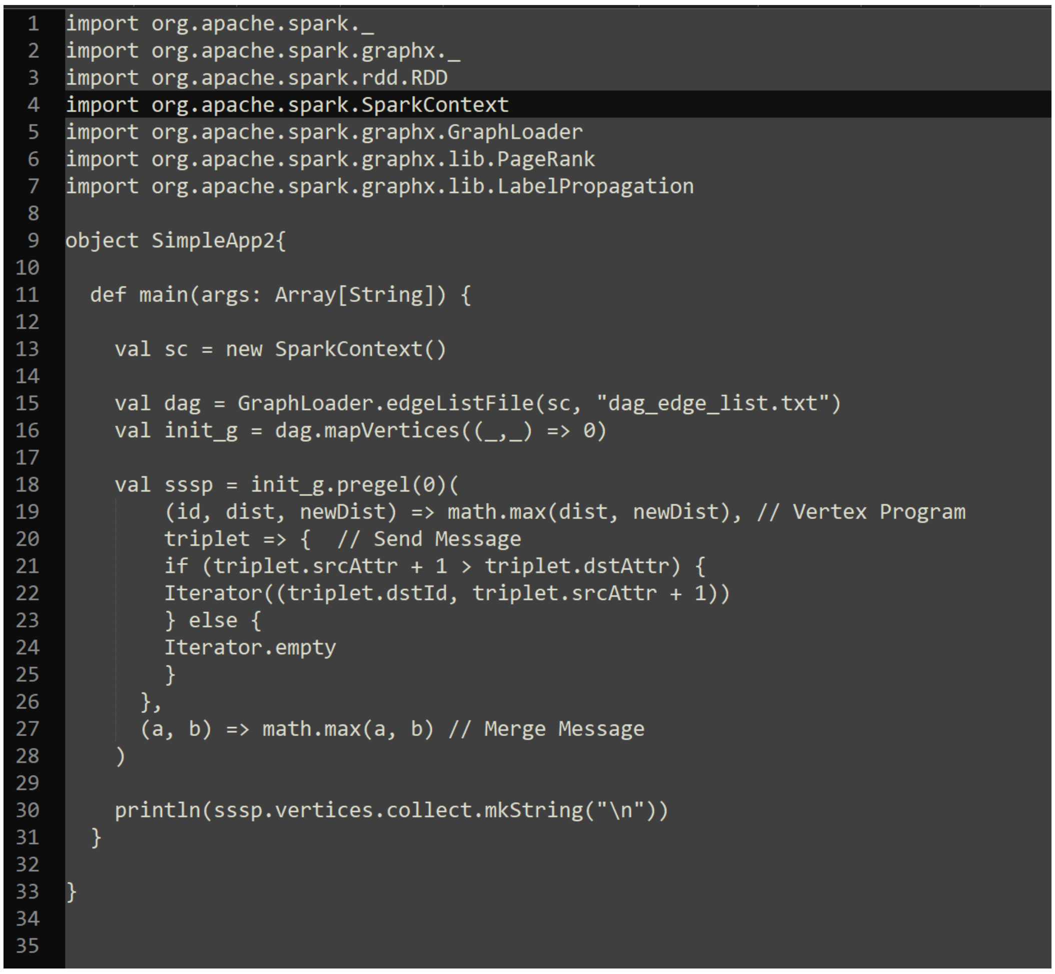
Vertices in the largest communities = 49058



```
d
14107
(69794,49058)
```

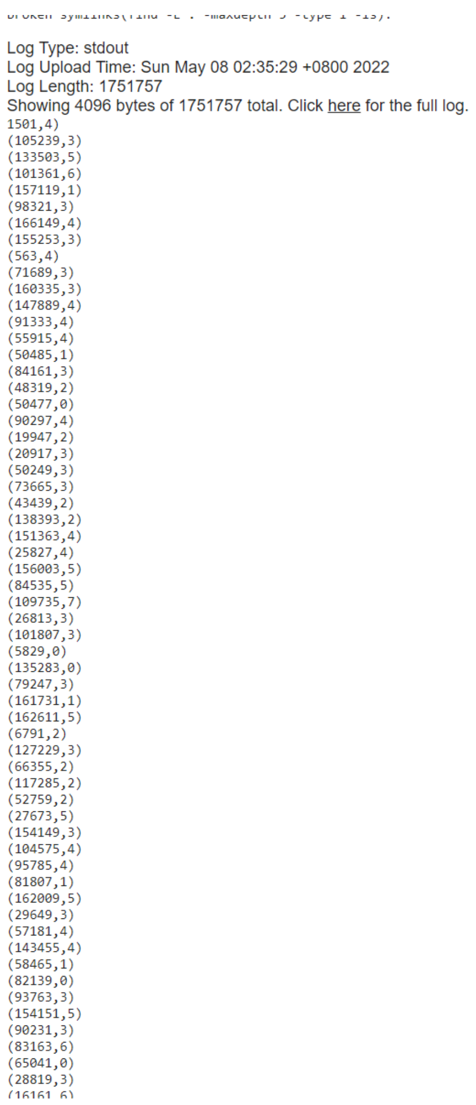
e.)

Code:



```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
import org.apache.spark.SparkContext
import org.apache.spark.graphx.GraphLoader
import org.apache.spark.graphx.lib.PageRank
import org.apache.spark.graphx.lib.LabelPropagation

object SimpleApp2{

  def main(args: Array[String]) {

    val sc = new SparkContext()

    val dag = GraphLoader.edgeListFile(sc, "dag_edge_list.txt")
    val init_g = dag.mapVertices((_,_) => 0)

    val sssp = init_g.pregel(0)(
        (id, dist, newDist) => math.max(dist, newDist), // Vertex Program
        triplet => {  // Send Message
        if (triplet.srcAttr + 1 > triplet.dstAttr) {
        Iterator((triplet.dstId, triplet.srcAttr + 1))
        } else {
        Iterator.empty
        }
      },
      (a, b) => math.max(a, b) // Merge Message
    )

    println(sssp.vertices.collect.mkString("\n"))
  }

}
```

Output:

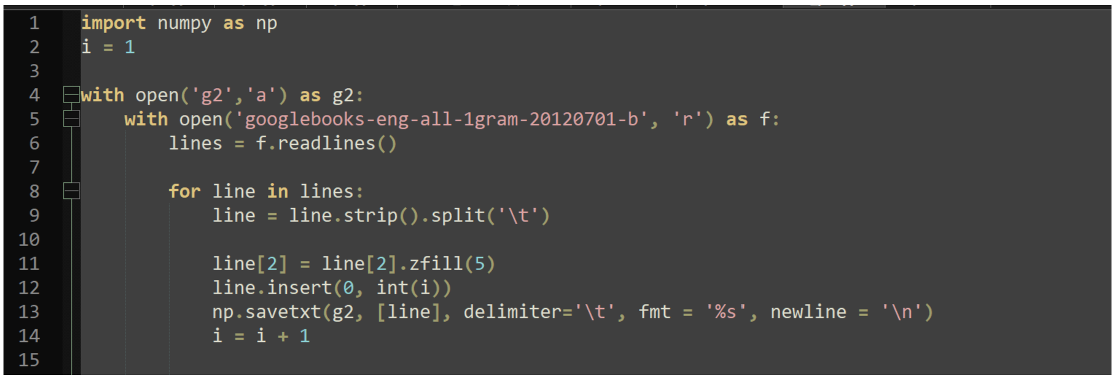


```
Broken symlinks(find -L . -maxdepth 5 -type l -ls):

Log Type: stdout
Log Upload Time: Sun May 08 02:35:29 +0800 2022
Log Length: 1751757
Showing 4096 bytes of 1751757 total. Click here for the full log.
1501,4)
(105239,3)
(133503,5)
(101361,6)
(157119,1)
(98321,3)
(166149,4)
(155253,3)
(563,4)
(71689,3)
(160335,3)
(147889,4)
(91333,4)
(55915,4)
(50485,1)
(84161,3)
(48319,2)
(50477,0)
(90297,4)
(19947,2)
(20917,3)
(50249,3)
(73665,3)
(43439,2)
(138393,2)
(151363,4)
(25827,4)
(156003,5)
(84535,5)
(109735,7)
(26813,3)
(101807,3)
(5829,0)
(135283,0)
(79247,3)
(161731,1)
(162611,5)
(6791,2)
(127229,3)
(66355,2)
(117285,2)
(52759,2)
(27673,5)
(154149,3)
(104575,4)
(95785,4)
(81807,1)
(162009,5)
(29649,3)
(57181,4)
(143455,4)
(58465,1)
(82139,0)
(93763,3)
(154151,5)
(90231,3)
(83163,6)
(65041,0)
(28819,3)
(16161,6)
```

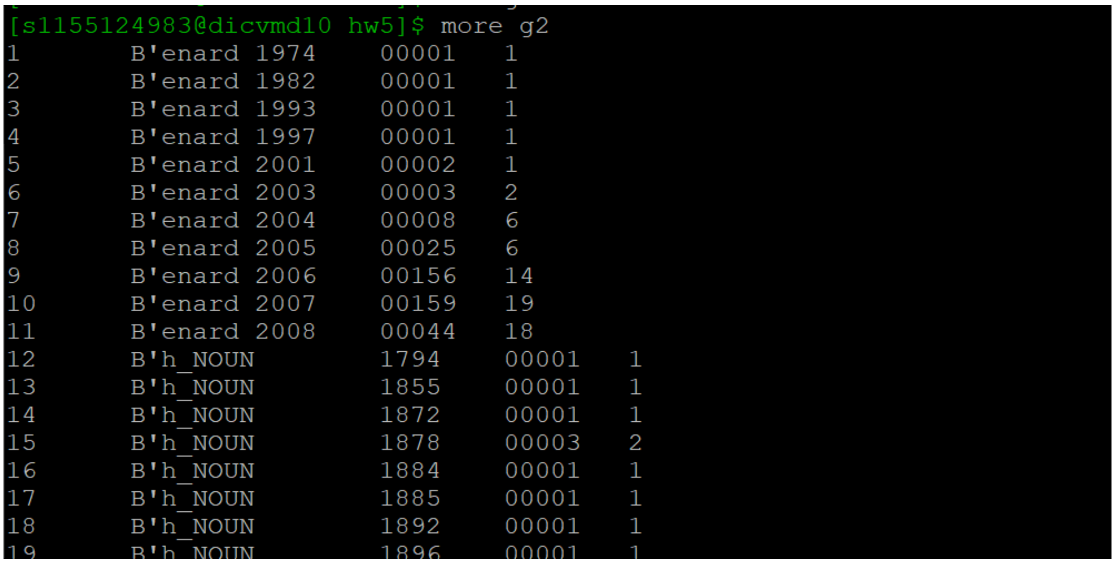
Q3.

a)

Adding row key and zero padding for occurrence count :

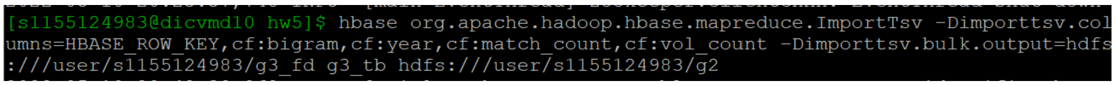


```python
import numpy as np
i = 1

with open('g2','a') as g2:
    with open('googlebooks-eng-all-1gram-20120701-b', 'r') as f:
        lines = f.readlines()

        for line in lines:
            line = line.strip().split('\t')

            line[2] = line[2].zfill(5)
            line.insert(0, int(i))
            np.savetxt(g2, [line], delimiter='\t', fmt = '%s', newline = '\n')
            i = i + 1

```



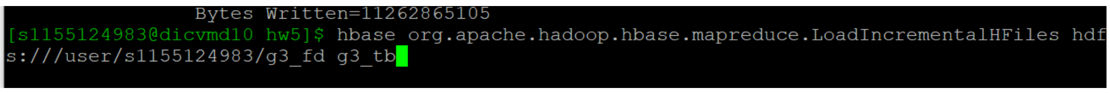
```
[s1155124983@dicvmd10 hw5]$ more g2
1       B'enard 1974    00001   1
2       B'enard 1982    00001   1
3       B'enard 1993    00001   1
4       B'enard 1997    00001   1
5       B'enard 2001    00002   1
6       B'enard 2003    00003   2
7       B'enard 2004    00008   6
8       B'enard 2005    00025   6
9       B'enard 2006    00156   14
10      B'enard 2007    00159   19
11      B'enard 2008    00044   18
12      B'h_NOUN        1794    00001   1
13      B'h_NOUN        1855    00001   1
14      B'h_NOUN        1872    00001   1
15      B'h_NOUN        1878    00003   2
16      B'h_NOUN        1884    00001   1
17      B'h_NOUN        1885    00001   1
18      B'h_NOUN        1892    00001   1
19      B'h_NOUN        1896    00001   1
```

ImportTsv:



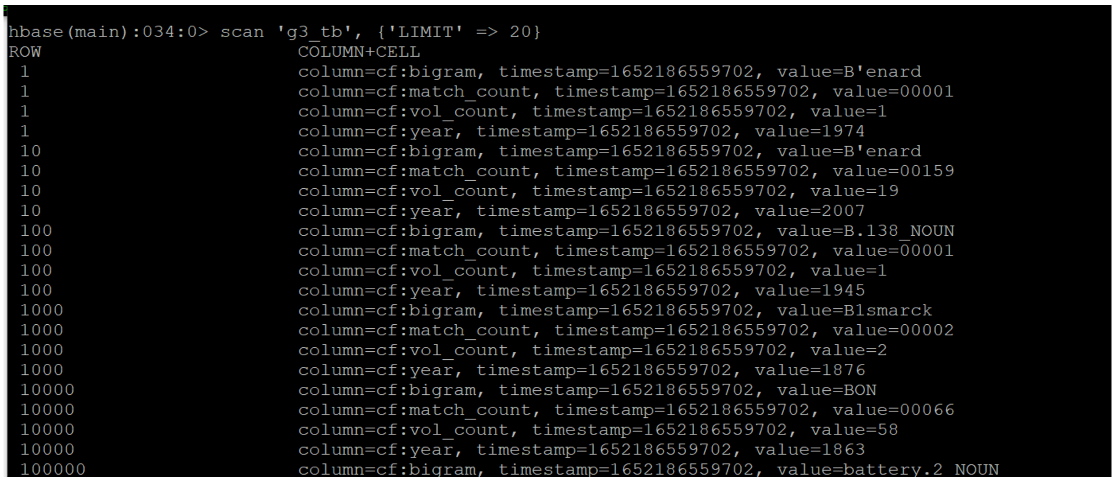
```
[s1155124983@dicvmd10 hw5]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.col
umns=HBASE_ROW_KEY,cf:bigram,cf:year,cf:match_count,cf:vol_count -Dimporttsv.bulk.output=hdfs
:///user/s1155124983/g3_fd g3_tb hdfs:///user/s1155124983/g2
```

Complete bulk load:



```
                Bytes Written=11262865105
[s1155124983@dicvmd10 hw5]$ hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles hdf
s:///user/s1155124983/g3_fd g3_tb
```

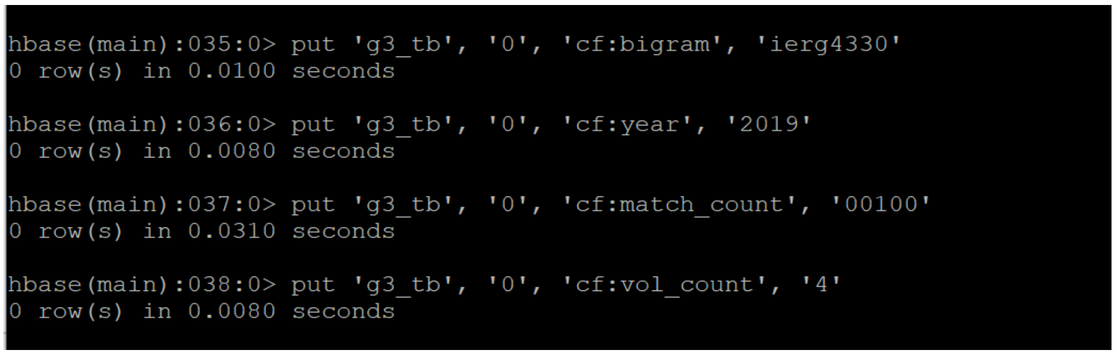
Showing the table:



```
hbase(main):034:0> scan 'g3_tb', {'LIMIT' => 20}
ROW                         COLUMN+CELL
 1                          column=cf:bigram, timestamp=1652186559702, value=B'enard
 1                          column=cf:match_count, timestamp=1652186559702, value=00001
 1                          column=cf:vol_count, timestamp=1652186559702, value=1
 1                          column=cf:year, timestamp=1652186559702, value=1974
 10                         column=cf:bigram, timestamp=1652186559702, value=B'enard
 10                         column=cf:match_count, timestamp=1652186559702, value=00159
 10                         column=cf:vol_count, timestamp=1652186559702, value=19
 10                         column=cf:year, timestamp=1652186559702, value=2007
 100                        column=cf:bigram, timestamp=1652186559702, value=B.138_NOUN
 100                        column=cf:match_count, timestamp=1652186559702, value=00001
 100                        column=cf:vol_count, timestamp=1652186559702, value=1
 100                        column=cf:year, timestamp=1652186559702, value=1945
 1000                       column=cf:bigram, timestamp=1652186559702, value=B1smarck
 1000                       column=cf:match_count, timestamp=1652186559702, value=00002
 1000                       column=cf:vol_count, timestamp=1652186559702, value=2
 1000                       column=cf:year, timestamp=1652186559702, value=1876
 10000                      column=cf:bigram, timestamp=1652186559702, value=BON
 10000                      column=cf:match_count, timestamp=1652186559702, value=00066
 10000                      column=cf:vol_count, timestamp=1652186559702, value=58
 10000                      column=cf:year, timestamp=1652186559702, value=1863
 100000                     column=cf:bigram, timestamp=1652186559702, value=battery.2_NOUN
```

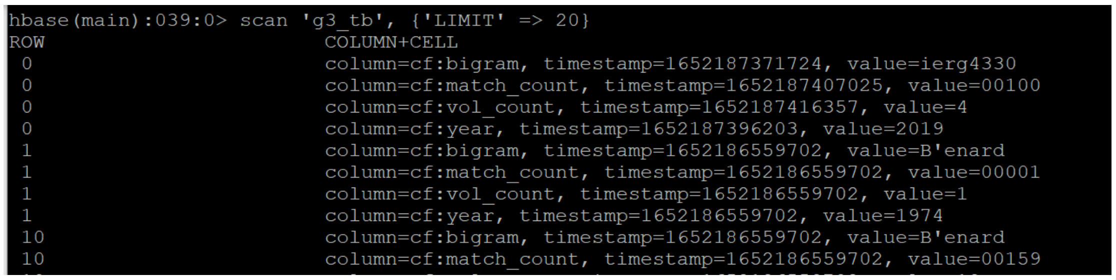
b.)

1. Insert ierg4330 2019 100 4



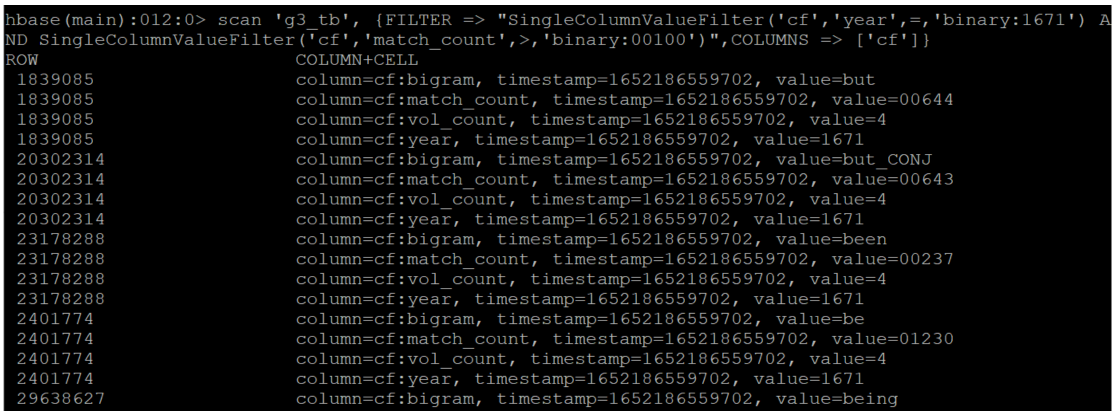
```
hbase(main):035:0> put 'g3_tb', '0', 'cf:bigram', 'ierg4330'
0 row(s) in 0.0100 seconds

hbase(main):036:0> put 'g3_tb', '0', 'cf:year', '2019'
0 row(s) in 0.0080 seconds

hbase(main):037:0> put 'g3_tb', '0', 'cf:match_count', '00100'
0 row(s) in 0.0310 seconds

hbase(main):038:0> put 'g3_tb', '0', 'cf:vol_count', '4'
0 row(s) in 0.0080 seconds
```

Updated table:



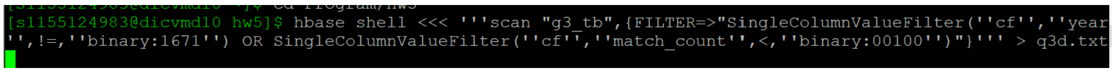
```
hbase(main):039:0> scan 'g3_tb', {'LIMIT' => 20}
ROW                          COLUMN+CELL
 0                           column=cf:bigram, timestamp=1652187371724, value=ierg4330
 0                           column=cf:match_count, timestamp=1652187407025, value=00100
 0                           column=cf:vol_count, timestamp=1652187416357, value=4
 0                           column=cf:year, timestamp=1652187396203, value=2019
 1                           column=cf:bigram, timestamp=1652186559702, value=B'enard
 1                           column=cf:match_count, timestamp=1652186559702, value=00001
 1                           column=cf:vol_count, timestamp=1652186559702, value=1
 1                           column=cf:year, timestamp=1652186559702, value=1974
 10                          column=cf:bigram, timestamp=1652186559702, value=B'enard
 10                          column=cf:match_count, timestamp=1652186559702, value=00159
```

2. Filter

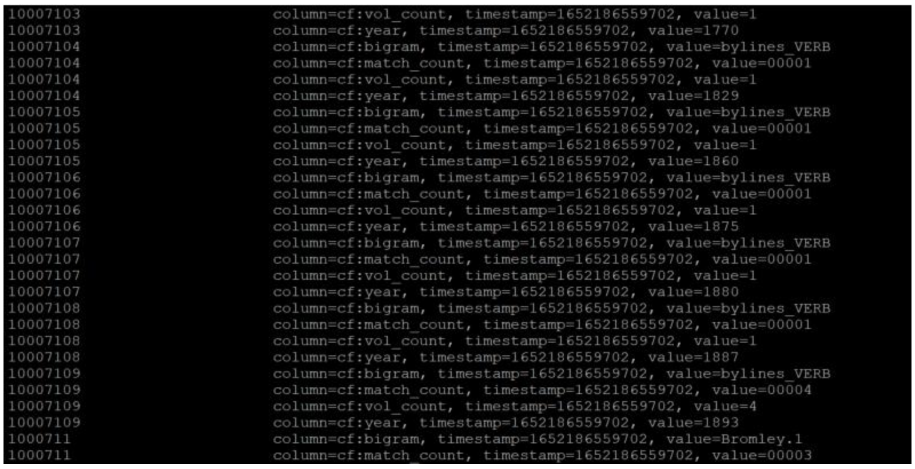


```
hbase(main):012:0> scan 'g3_tb', {FILTER => "SingleColumnValueFilter('cf','year',=,'binary:1671') A
ND SingleColumnValueFilter('cf','match_count',>,'binary:00100')",COLUMNS => ['cf']}
ROW                        COLUMN+CELL
 1839085                   column=cf:bigram, timestamp=1652186559702, value=but
 1839085                   column=cf:match_count, timestamp=1652186559702, value=00644
 1839085                   column=cf:vol_count, timestamp=1652186559702, value=4
 1839085                   column=cf:year, timestamp=1652186559702, value=1671
 20302314                  column=cf:bigram, timestamp=1652186559702, value=but_CONJ
 20302314                  column=cf:match_count, timestamp=1652186559702, value=00643
 20302314                  column=cf:vol_count, timestamp=1652186559702, value=4
 20302314                  column=cf:year, timestamp=1652186559702, value=1671
 23178288                  column=cf:bigram, timestamp=1652186559702, value=been
 23178288                  column=cf:match_count, timestamp=1652186559702, value=00237
 23178288                  column=cf:vol_count, timestamp=1652186559702, value=4
 23178288                  column=cf:year, timestamp=1652186559702, value=1671
 2401774                   column=cf:bigram, timestamp=1652186559702, value=be
 2401774                   column=cf:match_count, timestamp=1652186559702, value=01230
 2401774                   column=cf:vol_count, timestamp=1652186559702, value=4
 2401774                   column=cf:year, timestamp=1652186559702, value=1671
 29638627                  column=cf:bigram, timestamp=1652186559702, value=being
```

3. Deleted the records in part 2



```
[s1155124983@dicvmd10 hw5]$ hbase shell <<< '''scan "g3_tb",{FILTER=>"SingleColumnValueFilter(''cf'',''year
'',!=,''binary:1671'') OR SingleColumnValueFilter(''cf'',''match_count'',<,''binary:00100'')"}''' > q3d.txt
```



```
10007103                        column=cf:vol_count, timestamp=1652186559702, value=1
10007103                        column=cf:year, timestamp=1652186559702, value=1770
10007104                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007104                        column=cf:match_count, timestamp=1652186559702, value=00001
10007104                        column=cf:vol_count, timestamp=1652186559702, value=1
10007104                        column=cf:year, timestamp=1652186559702, value=1829
10007105                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007105                        column=cf:match_count, timestamp=1652186559702, value=00001
10007105                        column=cf:vol_count, timestamp=1652186559702, value=1
10007105                        column=cf:year, timestamp=1652186559702, value=1860
10007106                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007106                        column=cf:match_count, timestamp=1652186559702, value=00001
10007106                        column=cf:vol_count, timestamp=1652186559702, value=1
10007106                        column=cf:year, timestamp=1652186559702, value=1875
10007107                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007107                        column=cf:match_count, timestamp=1652186559702, value=00001
10007107                        column=cf:vol_count, timestamp=1652186559702, value=1
10007107                        column=cf:year, timestamp=1652186559702, value=1880
10007108                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007108                        column=cf:match_count, timestamp=1652186559702, value=00001
10007108                        column=cf:vol_count, timestamp=1652186559702, value=1
10007108                        column=cf:year, timestamp=1652186559702, value=1887
10007109                        column=cf:bigram, timestamp=1652186559702, value=bylines_VERB
10007109                        column=cf:match_count, timestamp=1652186559702, value=00004
10007109                        column=cf:vol_count, timestamp=1652186559702, value=4
10007109                        column=cf:year, timestamp=1652186559702, value=1893
1000711                         column=cf:bigram, timestamp=1652186559702, value=Bromley.1
1000711                         column=cf:match_count, timestamp=1652186559702, value=00003
```