Q1

First I downloaded the required .tsv using wget and use the following command to put it to hdfs:

wget https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/mooc_actions.tsv

wget https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/vertices.tsv

hdfs dfs -put ./vertices.tsv ./input

hdfs dfs -put ./vertices.tsv ./input

1a)

Code:

q1a.py > ...

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

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

# Load data
vertices = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/vertices.tsv')
edges = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/mooc_actions.tsv')


# Rename columns to match GraphFrames requirements
edges = edges.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

# Create GraphFrame
graph = GraphFrame(vertices, edges)

# Total number of vertices
total_vertices = graph.vertices.count()
print("Total number of vertices: {}").format(total_vertices)

# Total number of edges
total_edges = graph.edges.count()
print("Total number of edges: {}").format(total_edges)

# Number of Users
num_users = graph.vertices.filter("type = 'User'").count()
print("Number of users: {}").format(num_users)

# Number of Course Activities
num_activities = graph.vertices.filter("type = 'Course Activity'").count()
print("Number of course activities: {}").format(num_activities)

# Degree information
in_degrees = graph.inDegrees
out_degrees = graph.outDegrees
max_in_degree = in_degrees.orderBy("inDegree", ascending=False).first()
max_out_degree = out_degrees.orderBy("outDegree", ascending=False).first()

print("Vertex with the largest in-degree: {}, In-degree: {}").format(max_in_degree['id'], max_in_degree['inDegree'])
print("Vertex with the largest out-degree: {}, Out-degree: {}").format(max_out_degree['id'], max_out_degree['outDegree'])
```

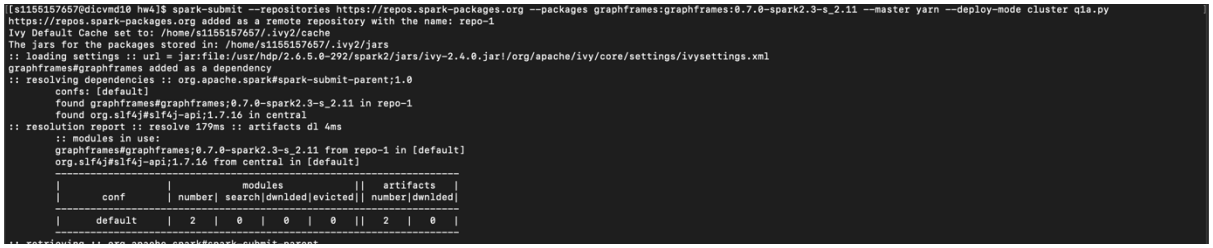
Explanation:

First get the files and put it to hdfs, then rename the column name to “src” and “dst” to fit the requirement of GraphFrame. Then extract the useful information required

Submit the job:

spark-submit --repositories https://repos.spark-packages.org --packages graphframes:graphframes:0.7.0-spark2.3-s_2.11 --master yarn --deploy-mode cluster q1a.py

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3710



Output:

```
Total number of vertices: 7144
Total number of edges: 411749
Number of users: 7047
Number of course activities: 97
Vertex with the largest in-degree: 7055, In-degree: 19474
Vertex with the largest out-degree: 1181, Out-degree: 505
```

I. 7144
II. 411749
III. 7047
IV. 97
V. 7055
VI. 1181

1b)

Code:

```python
from pyspark.sql import SQLContext
from pyspark import SparkContext
from graphframes import *

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

# Load data
vertices = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/vertic
edges = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/mooc_acti

# Rename columns to match GraphFrames requirements
edges = edges.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

# filter
filtered_edges = edges.filter("timestamp >= 10000 AND timestamp <= 50000")

# Create GraphFrame with filtered edges
graph = GraphFrame(vertices, filtered_edges)

# drop according to the question
subgraph = graph.dropIsolatedVertices()

num_vertices_subgraph = subgraph.vertices.count()
num_edges_subgraph = subgraph.edges.count()

print("Number of nodes in the subgraph: {}").format(num_vertices_subgraph)
print("Number of edges in the subgraph: {}").format(num_edges_subgraph)
```

Explanation:

Filter the edge first and drop all the isolated vertices after the filter.

Submit the job:

spark-submit --repositories https://repos.spark-packages.org --packages graphframes:graphframes:0.7.0-spark2.3-s_2.11 --master yarn --deploy-mode cluster q1b.py

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3727

```
[s1155157657@dicvmd10 hw4]$ spark-submit --repositories https://repos.spark-packages.org --packages graphframes:graphframes:0.7.0-spark2.3-s_2.11 --master yarn --deploy-mode cluster q1b.py
https://repos.spark-packages.org added as a remote repository with the name: repo-1
Ivy Default Cache set to: /home/s1155157657/.ivy2/cache
The jars for the packages stored in: /home/s1155157657/.ivy2/jars
:: loading settings :: url = jar:file:/usr/hdp/2.6.5.0-292/spark2/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
graphframes#graphframes added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
        confs: [default]
        found graphframes#graphframes;0.7.0-spark2.3-s_2.11 in repo-1
        found org.slf4j#slf4j-api;1.7.16 in central
:: resolution report :: resolve 173ms :: artifacts dl 3ms
        :: modules in use:
        graphframes#graphframes;0.7.0-spark2.3-s_2.11 from repo-1 in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   2   |   0   |   0   |   0   ||   2   |   0   |
        ---------------------------------------------------------------------
```

Output:

```
Number of nodes in the subgraph: 49
Number of edges in the subgraph: 243
```

1c)

Code:

```python
from pyspark.sql import SQLContext
from pyspark import SparkContext
from graphframes import *

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

# Load data
vertices = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/vertic
edges = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true', delimiter='\t').load('./input/mooc_acti

# Rename columns to match GraphFrames requirements
edges = edges.withColumnRenamed("USERID", "src").withColumnRenamed("TARGETID", "dst")

# filter
filtered_edges = edges.filter("timestamp >= 10000 AND timestamp <= 50000")

# Create GraphFrame with filtered edges
graph = GraphFrame(vertices, filtered_edges)

# drop according to the question
subgraph = graph.dropIsolatedVertices()

# Find the motif
results1 = subgraph.find("(A)-[AB]->(B); (C)-[CB]->(B)")
results2 = subgraph.find("(A)-[AB]->(B); (B)-[BC]->(C)")
results3 = subgraph.find("(A)-[AC]->(C); (B)-[BC]->(C); (A)-[AD]->(D); (B)-[BD]->(D)")
results4 = subgraph.find("(A)-[AC]->(C); (A)-[AD]->(D); (B)-[BD]->(D); (B)-[BE]->(E)")
```

```python

# Filter the DISTINCT vertices
result1_count = results1.filter("A.id != B.id AND B.id != C.id AND A.id != C.id").count()
result2_count = results2.filter("A.id != B.id AND B.id != C.id AND A.id != C.id").count()
result3_count = results3.filter(
    "A.id != B.id AND A.id != C.id AND A.id != D.id AND " +
    "B.id != C.id AND B.id != D.id AND C.id != D.id"
).count()
result4_count = results4.filter(
    "A.id != B.id AND A.id != C.id AND A.id != D.id AND A.id != E.id AND " +
    "B.id != C.id AND B.id != D.id AND B.id != E.id AND " +
    "C.id != D.id AND C.id != E.id AND D.id != E.id"
).count()

print("Number of occurrences for (i): {}").format(result1_count)
print("Number of occurrences for (ii): {}").format(result2_count)
print("Number of occurrences for (iii): {}").format(result3_count)
print("Number of occurrences for (iv): {}").format(result4_count)
```

Explanation:

Construct the relationship for the nodes, then filter all the node to prevent it pointing to itself and avoid duplicated node in a graph. Then use count() function to get the occurrence for each graph.

Output:

```
Number of occurrences for (i): 4744
Number of occurrences for (ii): 0
Number of occurrences for (iii): 33660
Number of occurrences for (iv): 374920
```

Q2)

First download the dataset using wget and put it to hdfs.

wget https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/edge_list.txt

hdfs dfs -put ./edge_list.txt ./input

2a)

Code:

q2a.scala

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

object q2a {
  def main(args: Array[String]) {
    val sc = new SparkContext()

    // load edge_list.txt
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // count vertices
    val numVertices = graph.vertices.count
    println(s"Number of vertices: $numVertices")

    // count edges
    val numEdges = graph.edges.count
    println(s"Number of edges: $numEdges")

    // find vertex with largest in-degree
    val maxInDegree = graph.inDegrees.reduce((a, b) => if (a._2 > b._2) a else b)
    println(s"Vertex with the largest in-degree: ${maxInDegree._1} with in-degree of ${maxInDegree._2}")

    // find vertex with largest out-degree
    val maxOutDegree = graph.outDegrees.reduce((a, b) => if (a._2 > b._2) a else b)
    println(s"Vertex with the largest out-degree: ${maxOutDegree._1} with out-degree of ${maxOutDegree._2}")

    sc.stop()
  }
}
```

Build file:

q2a_build.sbt

```
name := "q2a"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

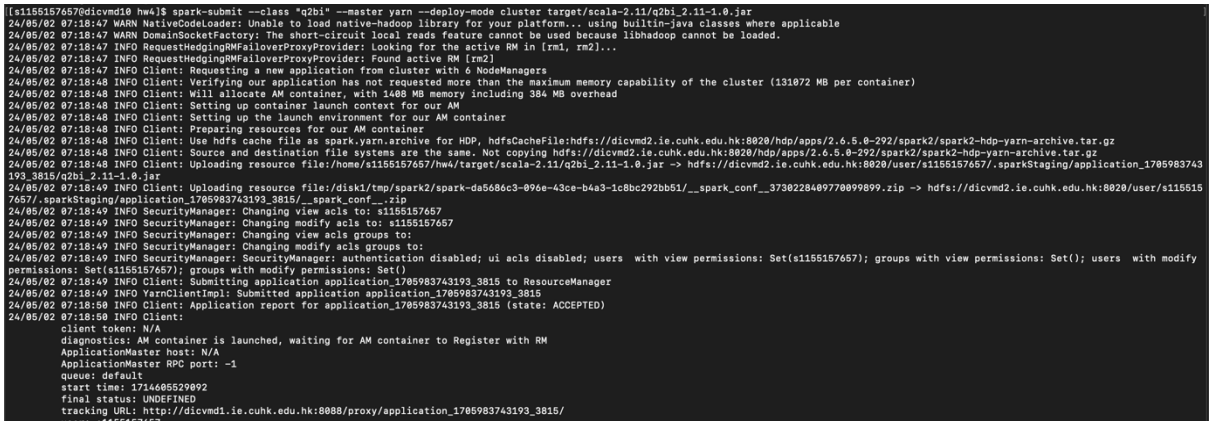
Explanation:

Just use different functions for finding number of edges, vertex with largest in-degree and largest out-degree.

Submit job:

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3815

spark-submit --class "q2a" --master yarn --deploy-mode cluster target/scala-2.11/q2a_2.11-1.0.jar



Output:

```
Number of vertices: 169343
Number of edges: 1166243
Vertex with the largest in-degree: 1353 with in-degree of 13155
Vertex with the largest out-degree: 72253 with out-degree of 436
```

2bi)

Code:

q2bi.scala

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

object q2bi {
  def main(args: Array[String]) {
    val sc = new SparkContext()

    // load edge_list.txt
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // get connected components
    val cc = graph.connectedComponents()

    // get the size of each connected component
    val componentCounts = cc.vertices.map(_._2).countByValue()

    // count the number of components and find the size of the largest component
    val numberOfComponents = componentCounts.size
    val largestComponentSize = componentCounts.values.max

    println(s"Number of connected components: $numberOfComponents")
    println(s"Number of vertices in the largest component: $largestComponentSize")

    sc.stop()
  }
}
```

build:

q2bi_build.sbt

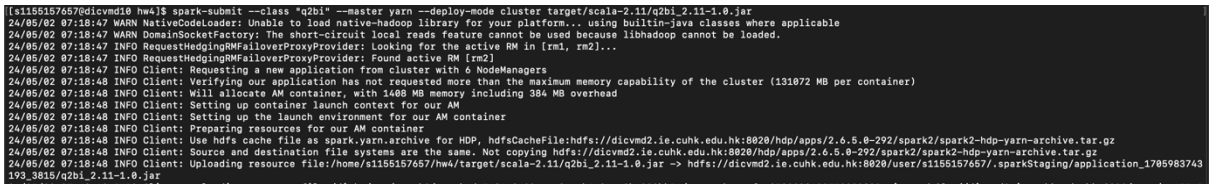
```
name := "q2bi"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

Explanation:

use countByValue() and find the maximum value and the size.

Submit job:

spark-submit --class "q2bi" --master yarn --deploy-mode cluster target/scala-2.11/q2bi_2.11-1.0.jar



```
[s1155157657@dicvmd10 hw4]$ spark-submit --class "q2bi" --master yarn --deploy-mode cluster target/scala-2.11/q2bi_2.11-1.0.jar
24/05/02 07:18:47 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/05/02 07:18:47 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
24/05/02 07:18:47 INFO RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
24/05/02 07:18:47 INFO RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
24/05/02 07:18:47 INFO Client: Requesting a new application from cluster with 6 NodeManagers
24/05/02 07:18:48 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (131072 MB per container)
24/05/02 07:18:48 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
24/05/02 07:18:48 INFO Client: Setting up container launch context for our AM
24/05/02 07:18:48 INFO Client: Setting up the launch environment for our AM container
24/05/02 07:18:48 INFO Client: Preparing resources for our AM container
24/05/02 07:18:48 INFO Client: Use hdfs cache file as spark.yarn.archive for HDP, hdfsCacheFile:hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 07:18:48 INFO Client: Source and destination file systems are the same. Not copying hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 07:18:48 INFO Client: Uploading resource file:/home/s1155157657/hw4/target/scala-2.11/q2bi_2.11-1.0.jar -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_3815/q2bi_2.11-1.0.jar
```

Output:

```
Number of connected components: 1
Number of vertices in the largest component: 169343
```

2bii)

Code:

q2bii.scala

```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

object q2bii {
  def main(args: Array[String]) {
    val sc = new SparkContext()

    // Load the graph from edge list
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // Compute the strongly connected components
    val scc = graph.stronglyConnectedComponents(1)

    // Get the size of each strongly connected component
    val componentCounts = scc.vertices.map(_._2).countByValue()

    // Count the number of components and find the size of the largest component
    val numberOfComponents = componentCounts.size
    val largestComponentSize = componentCounts.values.max

    println(s"Number of strongly connected components: $numberOfComponents")
    println(s"Number of vertices in the largest strongly connected component: $largestComponentSize")

    sc.stop()
  }
}
```

q2bii_build.sbt:

q2bii_build.sbt

```
name := "q2bii"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

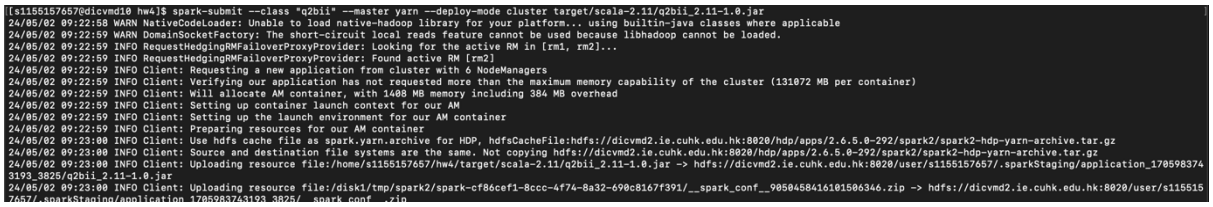
Explanation:

Basically the same as 2bi just change the function to StronglyConnectedComponents().

Submit job:

spark-submit --class "q2bii" --master yarn --deploy-mode cluster target/scala-2.11/q2bii_2.11-1.0.jar

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3825



```
[s1155157657@dicvmd10 hw4]$ spark-submit --class "q2bii" --master yarn --deploy-mode cluster target/scala-2.11/q2bii_2.11-1.0.jar
24/05/02 09:22:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/05/02 09:22:59 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
24/05/02 09:22:59 INFO RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
24/05/02 09:22:59 INFO RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
24/05/02 09:22:59 INFO Client: Requesting a new application from cluster with 6 NodeManagers
24/05/02 09:22:59 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (131072 MB per container)
24/05/02 09:22:59 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
24/05/02 09:22:59 INFO Client: Setting up container launch context for our AM
24/05/02 09:22:59 INFO Client: Setting up the launch environment for our AM container
24/05/02 09:22:59 INFO Client: Preparing resources for our AM container
24/05/02 09:23:00 INFO Client: Use hdfs cache file as spark.yarn.archive for HDP, hdfsCacheFile:hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 09:23:00 INFO Client: Source and destination file systems are the same. Not copying hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 09:23:00 INFO Client: Uploading resource file:/home/s1155157657/hw4/target/scala-2.11/q2bii_2.11-1.0.jar -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_3825/q2bii_2.11-1.0.jar
24/05/02 09:23:00 INFO Client: Uploading resource file:/disk1/tmp/spark2/spark-cf86cef1-8ccc-4f74-8a32-690c8167f391/__spark_conf__9050458416101506346.zip -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983743193_3825/__spark_conf__.zip
```

Output:

```
Number of strongly connected components: 169343
Number of vertices in the largest strongly connected component: 1
```

2ci)

Code:

q2c.scala

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

object q2c {
  def main(args: Array[String]) {
    val sc = new SparkContext(new SparkConf())

    // load the graph from edge list
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // define the target vertices
    val targetVertices = Array(4330, 5730)

    targetVertices.foreach { target =>
      // personalized PageRank
      val prGraph = graph.personalizedPageRank(target, tol = 0.0001, resetProb = 0.15)

      // top 20 scores for each target
      val top20 = prGraph.vertices
        .filter { case (id, _) => id != target }
        .sortBy(_._2, ascending = false)
        .take(20)

      println(s"Top 20 nodes with highest PageRank for vertex $target:")
      top20.foreach { case (id, rank) =>
        println(s"Vertex $id has PageRank: $rank")
      }
      println("")
    }

    sc.stop()
  }
}
```

q2c_builder:

q2c_build.sbt

```
name := "q2c"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

Explanation:

First we define the two id that we would like to do personalizedPageRank(). Then for each we compute the pagerank and filter all the point-to relations (without itself), filter, sort and take top 20 vertices.

Submit job:

spark-submit --class "q2c" --master yarn --deploy-mode cluster target/scala-2.11/q2c_2.11-1.0.jar

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3830

```
[s1155157657@dicvmd10 q2c]$ spark-submit --class "q2c" --master yarn --deploy-mode cluster target/scala-2.11/q2c_2.11-1.0.jar
24/05/02 10:27:26 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/05/02 10:27:27 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
24/05/02 10:27:27 INFO RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
24/05/02 10:27:27 INFO RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
24/05/02 10:27:27 INFO Client: Requesting a new application from cluster with 6 NodeManagers
24/05/02 10:27:27 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (131072 MB per container)
24/05/02 10:27:27 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
24/05/02 10:27:27 INFO Client: Setting up container launch context for our AM
24/05/02 10:27:27 INFO Client: Setting up the launch environment for our AM container
24/05/02 10:27:27 INFO Client: Preparing resources for our AM container
24/05/02 10:27:28 INFO Client: Use hdfs cache file as spark.yarn.archive for HDP, hdfsCacheFile:hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 10:27:28 INFO Client: Source and destination file systems are the same. Not copying hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 10:27:28 INFO Client: Uploading resource file:/home/s1155157657/hw4/q2c/target/scala-2.11/q2c_2.11-1.0.jar -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/.sparkStaging/application_1705983
743193_3830/q2c_2.11-1.0.jar
24/05/02 10:27:28 INFO Client: Uploading resource file:/disk1/tmp/spark2/spark-9c95d7a4-a56d-4066-86ac-cb7f6b865944/__spark_conf__5289411884092470773.zip -> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s115515
```

Output:

```
Top 20 nodes with highest PageRank for vertex 4330:
Vertex 153677 has PageRank: 0.06507955005454329
Vertex 16921 has PageRank: 0.04421509106214325
Vertex 123933 has PageRank: 0.042350785121722495
Vertex 162330 has PageRank: 0.03950727084141603
Vertex 63804 has PageRank: 0.03067444841376113
Vertex 19645 has PageRank: 0.02932353159302809
Vertex 113389 has PageRank: 0.02883692060880084
Vertex 26614 has PageRank: 0.027910872617384428
Vertex 111306 has PageRank: 0.027707921630210023
Vertex 133211 has PageRank: 0.027707921630210023
Vertex 71148 has PageRank: 0.019233777870669645
Vertex 69228 has PageRank: 0.01664258062749259
Vertex 81746 has PageRank: 0.015547240762744939
Vertex 112464 has PageRank: 0.015475530550481814
Vertex 62510 has PageRank: 0.01434231453990431
Vertex 42645 has PageRank: 0.014077850757206273
Vertex 119218 has PageRank: 0.013818407806099244
Vertex 96770 has PageRank: 0.012674889078343226
Vertex 77078 has PageRank: 0.012190723804344814
Vertex 116080 has PageRank: 0.011800377285354909

Top 20 nodes with highest PageRank for vertex 5730:
Vertex 102862 has PageRank: 0.19219898247597514
Vertex 165911 has PageRank: 0.19219898247597514
Vertex 141857 has PageRank: 0.16336913510457887
Vertex 138859 has PageRank: 0.0
Vertex 113843 has PageRank: 0.0
Vertex 102831 has PageRank: 0.0
Vertex 103301 has PageRank: 0.0
Vertex 110163 has PageRank: 0.0
Vertex 19021 has PageRank: 0.0
Vertex 111517 has PageRank: 0.0
Vertex 31037 has PageRank: 0.0
Vertex 101257 has PageRank: 0.0
Vertex 34207 has PageRank: 0.0
Vertex 165737 has PageRank: 0.0
Vertex 135935 has PageRank: 0.0
Vertex 138469 has PageRank: 0.0
Vertex 138533 has PageRank: 0.0
Vertex 156041 has PageRank: 0.0
Vertex 53499 has PageRank: 0.0
Vertex 155943 has PageRank: 0.0
```

2cii)

Code:

q2cii.scala

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

object q2cii {
  def main(args: Array[String]) {
    val sc = new SparkContext()

    // load the graph from an edge list file
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // personalized PageRank for 5730
    val targetVertex = 5730
    val personalizedPR = graph.personalizedPageRank(targetVertex, tol = 0.0001, resetProb = 0.15)

    // get the top 2000 nodes with the highest PageRank scores, excluding the target vertex itself
    val top2000Nodes = personalizedPR.vertices
      .filter { case (id, _) => id != targetVertex }
      .sortBy(_._2, ascending = false)
      .map(_._1)
      .take(2000)

    val top2000NodesSet = sc.broadcast(top2000Nodes.toSet)

    val top2000NodesSet_value = top2000NodesSet.value

    // construct a subgraph using these 2000 nodes
    val subgraph = graph.subgraph(vpred = (id, _) => top2000NodesSet_value.contains(id))

    // Count the edges in the subgraph
    val edgeCount = subgraph.edges.count()

    println(s"The subgraph contains $edgeCount edges.")

    sc.stop()
  }
}
```

q2cii_build.sbt:

q2cii_build.sbt

```
name := "q2cii"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

Explanation:

After calculating the top 2000 relevant node, then we make it into a set then make a subgraph if the id is in the set and count the edges involved.

Submit job:

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3850

spark-submit --class "q2cii" --master yarn --deploy-mode cluster target/scala-2.11/q2cii_2.11-1.0.jar

```
[s1155157657@dicvmd10 q2cii]$ spark-submit --class "q2cii" --master yarn --deploy-mode cluster target/scala-2.11/q2cii_2.11-1.0.jar
24/05/02 12:42:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/05/02 12:42:41 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
24/05/02 12:42:41 INFO RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
24/05/02 12:42:41 INFO RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
24/05/02 12:42:41 INFO Client: Requesting a new application from cluster with 6 NodeManagers
24/05/02 12:42:41 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (131072 MB per container)
24/05/02 12:42:41 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
24/05/02 12:42:41 INFO Client: Setting up container launch context for our AM
24/05/02 12:42:41 INFO Client: Setting up the launch environment for our AM container
24/05/02 12:42:41 INFO Client: Preparing resources for our AM container
24/05/02 12:42:42 INFO Client: Use hdfs cache file as spark.yarn.archive for HDP, hdfsCacheFile:hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
24/05/02 12:42:42 INFO Client: Source and destination file systems are the same. Not copying hdfs://dicvmd2.ie.cuhk.edu.hk:8020/hdp/apps/2.6.5.0-292/spark2/spark2-hdp-yarn-archive.tar.gz
```

Output:

```
Log Length: 55
The subgraph contains 165 edges.
```

2d)

Code:

q2d.scala

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

object q2d {
  def main(args: Array[String]): Unit = {
    val sc = new SparkContext()

    // load the graph from an edge list file
    val graph = GraphLoader.edgeListFile(sc, "./input/edge_list.txt")

    // run Label Propagation Algorithm to find communities
    val result = lib.LabelPropagation.run(graph, maxSteps = 50)

    // extract communities
    val communities = result.vertices.map(_._2)

    // count the communities
    val distinctCommunities = communities.distinct().count()

    // Find the size of the largest community
    val largestCommunitySize = communities
      .map(label => (label, 1))
      .reduceByKey(_ + _)
      .map(_._2)
      .max()

    println(s"Total number of distinct communities: $distinctCommunities")
    println(s"Size of the largest community: $largestCommunitySize")

    sc.stop()
  }
}
```

q2d_build.sbt

```
name := "q2cii"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.0"
libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.0"
```

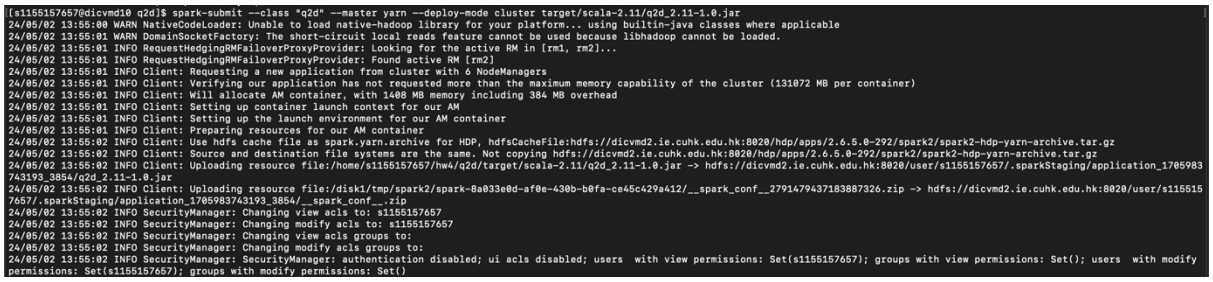
Explanation:

By calling the library we have communities, .Then we can map it to (label,1) and reduce by label, in that case we have a count. Choose the max() for largest community

Submit job:

spark-submit --class "q2d" --master yarn --deploy-mode cluster target/scala-2.11/q2d_2.11-1.0.jar

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3854



Output:

```
Total number of distinct communities: 14107
Size of the largest community: 49058
```

Q3)

3a)

Get from google:

wget http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-b.gz

Put it to hdfs:

hdfs dfs -put googlebooks-eng-all-1gram-20120701-b ./input

Create Hbase table:

echo "create 'google_books', 'data', {SPLITS => ['m', 't']}" | hbase shell

```
[s1155157657@dicvmd10 q3]$ echo "create 'google_books', 'data', {SPLITS => ['m', 't']}" | hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.5.0-292, r897822d4dd5956ca186974c10382e9094683fa29, Fri May 11 08:00:59 UTC 2018

create 'google_books', 'data', {SPLITS => ['m', 't']}
0 row(s) in 1.3870 seconds

Hbase::Table - google_books
```

Create HFile:

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3920

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv \

-Dimporttsv.columns=HBASE_ROW_KEY,data:year,data:match_count,data:volume_count \

-Dimporttsv.separator=$'\t'\

-Dimporttsv.bulk.output=./output/google_book \

google_books \

hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/input/googlebooks-eng-all-1gram-20120701-b

```
[s1155157657@dicvmd10 q3]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv \
>       -Dimporttsv.columns=HBASE_ROW_KEY,data:year,data:match_count,data:volume_count \
>       -Dimporttsv.separator=$'\t'\
>       -Dimporttsv.bulk.output=./output/google_book \
>       google_books \
> hdfs://dicvmd2.ie.cuhk.edu.hk:8020/user/s1155157657/input/googlebooks-eng-all-1gram-20120701-b
2024-05-02 22:10:23,944 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0
x6a6afff2 connecting to ZooKeeper ensemble=dicvmd1.ie.cuhk.edu.hk:2181,dicvmd2.ie.cuhk.edu.hk:2181,di
cvmd3.ie.cuhk.edu.hk:2181,dicvmd7.ie.cuhk.edu.hk:2181
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-
292--1, built on 05/11/2018 07:15 GMT
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:host.name=dicvmd10.ie.cu
hk.edu.hk
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_112
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corpo
ration
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:java.home=/usr/jdk64/jdk
1.8.0_112/jre
2024-05-02 22:10:23,950 INFO  [main] zookeeper.ZooKeeper: Client environment:java.class.path=/usr/hdp
/2.6.5.0-292/hbase/conf:/usr/jdk64/jdk1.8.0_112/lib/tools.jar:/usr/hdp/2.6.5.0-292/hbase:/usr/hdp/2.6
```

Load it into HBase table:

/usr/hdp/2.6.5.0-292/hbase/bin/hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles ./output/google_book google_books

http://dicvmd1.ie.cuhk.edu.hk:8088/cluster/app/application_1705983743193_3921

```
[s1155157657@dicvmd10 hw4]$ /usr/hdp/2.6.5.0-292/hbase/bin/hbase org.apache.hadoop.hbase.mapreduce.Lo
adIncrementalHFiles ./output/google_book google_books
2024-05-02 22:33:31,798 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0
x57d7f8ca connecting to ZooKeeper ensemble=dicvmd1.ie.cuhk.edu.hk:2181,dicvmd2.ie.cuhk.edu.hk:2181,di
cvmd3.ie.cuhk.edu.hk:2181,dicvmd7.ie.cuhk.edu.hk:2181
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-
292--1, built on 05/11/2018 07:15 GMT
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:host.name=dicvmd10.ie.cu
hk.edu.hk
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_112
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corpo
ration
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:java.home=/usr/jdk64/jdk
1.8.0_112/jre
2024-05-02 22:33:31,803 INFO  [main] zookeeper.ZooKeeper: Client environment:java.class.path=/usr/hdp
/2.6.5.0-292/hbase/conf:/usr/jdk64/jdk1.8.0_112/lib/tools.jar:/usr/hdp/2.6.5.0-292/hbase:/usr/hdp/2.6
.5.0-292/hbase/lib/activation-1.1.jar:/usr/hdp/2.6.5.0-292/hbase/lib/aopalliance-1.0.jar:/usr/hdp/2.6
.5.0-292/hbase/lib/apacheds-i18n-2.0.0-M15.jar:/usr/hdp/2.6.5.0-292/hbase/lib/apacheds-kerberos-codec
```

b)

I use shell in this question

b1)

use hbase shell, and the following 3 command:

put 'google_books', 'ierg4330', 'data:year', '2019'

put 'google_books', 'ierg4330', 'data:match_count', '100'

put 'google_books', 'ierg4330', 'data:volume_count', '4'

```
[hbase(main):001:0> put 'google_books', 'ierg4330', 'data:year', '2019'
0 row(s) in 0.1650 seconds

[hbase(main):002:0> put 'google_books', 'ierg4330', 'data:match_count', '100'
0 row(s) in 0.0100 seconds

[hbase(main):003:0> put 'google_books', 'ierg4330', 'data:volume_count', '4'
0 row(s) in 0.6450 seconds
```

b2)

scan 'google_books', {COLUMNS => ['data:year', 'data:match_count'], FILTER => "(SingleColumnValueFilter('data','year',=,'binary:1671') AND SingleColumnValueFilter('data','match_count',>, 'binary:100'))"}

This command uses two SingleColumnValueFilter for getting year column equals 1671 and match_count column has a value greater than 100.

```
[hbase(main):004:0> scan 'google_books', {COLUMNS => ['data:year', 'data:match_count'], FILTER => "(Si
ngleColumnValueFilter('data','year',=,'binary:1671') AND SingleColumnValueFilter('data','match_count'
,>, 'binary:100'))"}
ROW                     COLUMN+CELL
 beggar                 column=data:match_count, timestamp=1714659023665, value=2
 beggar                 column=data:year, timestamp=1714659023665, value=1671
1 row(s) in 1.2380 seconds
```

b3)

deleteall 'google_books', ‘beggar’

After getting the ‘beggar’ row in b2, delete it from the table. And verify it is deleted.

```
[hbase(main):005:0> deleteall 'google_books', 'beggar'
0 row(s) in 0.0180 seconds

[hbase(main):006:0> scan 'google_books', {COLUMNS => ['data:year', 'data:match_count'], FILTER => "(Si
ngleColumnValueFilter('data','year',=,'binary:1671') AND SingleColumnValueFilter('data','match_count'
,>, 'binary:100'))"}
ROW                     COLUMN+CELL
0 row(s) in 1.1610 seconds

hbase(main):007:0>
```