ID2221 Data Intensive Computing

Lab2 - Spark Streaming and GraphX

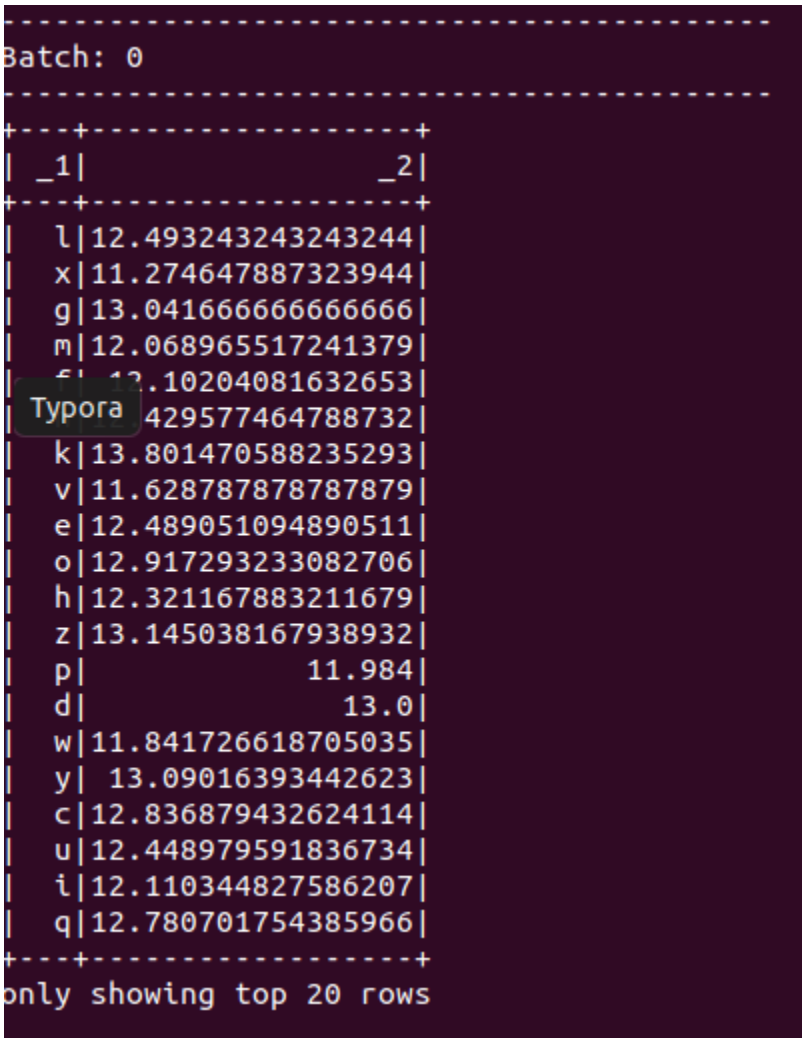
Group 1: Yizhan Wu (yizhanw@kth.se), Yage Hao (yage@kth.se)

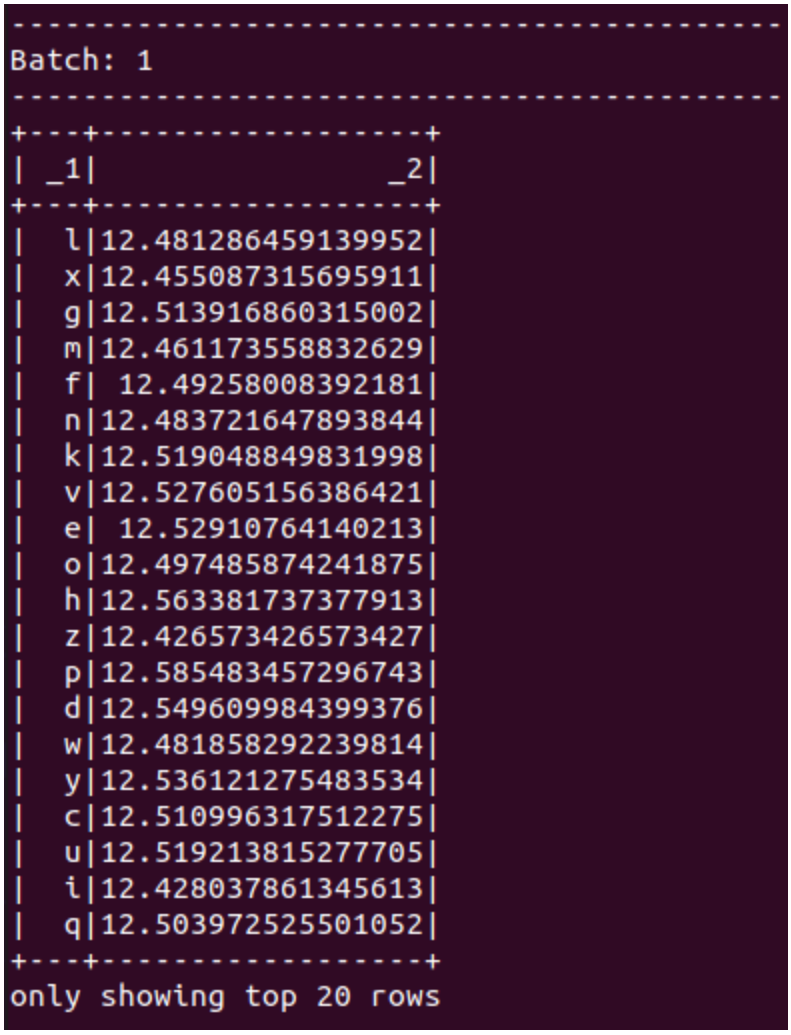
Task 1 Results:

```
21/10/09 18:23:28 INFO DAGScheduler: Job 86 finished: print at KafkaSpark.scala:
47, took 0.468199 s
-------------------------------------------
Time: 1633796608000 ms
-------------------------------------------
(r,12.235109717868339)
(b,12.049562682215743)
(z,12.29391891891892)
(l,12.229681978798586)
(x,12.508426966292134)
(p,12.68867924528302)
(t,12.090592334494774)
(x,12.515406162464986)
(v,12.516556291390728)
(v,12.528052805280527)
...
```

```
21/10/09 18:23:29 INFO DAGScheduler: Job 87 finished: print at KafkaSpark.scala:
47, took 0.165465 s
-------------------------------------------
Time: 1633796609000 ms
-------------------------------------------
(x,12.611180124223603)
(f,12.003663003663004)
(p,12.723700887198985)
(h,12.855844155844157)
(x,12.601736972704714)
(r,12.490322580645161)
(d,11.900962861072902)
(n,12.674623115577889)
(v,12.567785234899329)
(l,12.245205479452055)
...
```

Task 2 Results:





```
-------------------------------------------
Batch: 1
-------------------------------------------
+---+------------------+
| _1|                _2|
+---+------------------+
|  l|12.481286459139952|
|  x|12.455087315695911|
|  g|12.513916860315002|
|  m|12.461173558832629|
|  f| 12.49258008392181|
|  n|12.483721647893844|
|  k|12.519048849831998|
|  v|12.527605156386421|
|  e| 12.52910764140213|
|  o|12.497485874241875|
|  h|12.563381737377913|
|  z|12.426573426573427|
|  p|12.585483457296743|
|  d|12.549609984399376|
|  w|12.481858292239814|
|  y|12.536121275483534|
|  c|12.510996317512275|
|  u|12.519213815277705|
|  i|12.428037861345613|
|  q|12.503972525501052|
+---+------------------+
only showing top 20 rows
```

Task 3:

In this task you will work with GraphX to process graph-based data. To do this assignment, write a code in GraphX to build the following graph and provide answer the questions. This graph shows a small social network with users and their ages modeled as vertices and likes modeled as directed edges.

Array_vertex = Array((1,(Alice,28)), (2,(Bob,27)), (3,(Charlie,65)), (4,(David,42)), (5,(Ed,55)), (6,(Fran,50)), (7,(Alex,55)))
Array_edge = Array(Edge(2,1,7), Edge(2,4,2), Edge(3,2,4), Edge(3,6,3), Edge(4,1,1), Edge(5,2,2), Edge(5,3,8), Edge(5,6,3), Edge(7,5,3), Edge(7,6,4))

vertices = ParallelCollectionRDD[0] at parallelize at <console>:53
edges = ParallelCollectionRDD[1] at parallelize at <console>:54
graph = org.apache.spark.graphx.impl.GraphImpl@7dee1ce

1.Display the names of the users that are at least 30 years old.

```
David
Fran
Charlie
Alex
Ed
```

2.Display who likes who.

```
Bob likes Alice.
Ed likes Bob.
Ed likes Charlie.
Ed likes Fran.
Alex likes Ed.
Alex likes Fran.
Bob likes David.
Charlie likes Bob.
Charlie likes Fran.
David likes Alice.
```

3.If someone likes someone else more than 5 times than that relationship is getting pretty serious, so now display the lovers

```
Bob likes Alice.
Ed likes Charlie.
```

4. Print the number of people who like each user (e.g., Alice is liked by 2 people).

```
David is liked by 1 people
Fran is liked by 3 people
Bob is liked by 2 people
Alice is liked by 2 people
Charlie is liked by 1 people
Alex is liked by 0 people
Ed is liked by 1 people
```

5.Print the names of the users who are liked by the same number of people they like (e.g., Bob and David)

```
Fran
Alice
```

6. Find the oldest follower of each user (hint: use the aggregateMessages).

```
Bob is the oldest follower of David.
Charlie is the oldest follower of Fran.
Charlie is the oldest follower of Bob.
David is the oldest follower of Alice.
Ed is the oldest follower of Charlie.
Alex does not have any followers.
Alex is the oldest follower of Ed.
```