

---

# **Lab 02: MapReduce Programming**

CSC14118 Introduction to Big Data 20KHMT1

All in

2023-04-11

# Contents

<b>Task progression</b>	<b>2</b>
<b>1 Problem 1: Word Count</b>	<b>3</b>
<b>2 Problem 2: Word Size Word Count</b>	<b>5</b>
<b>3 Problem 3: Weather Data</b>	<b>7</b>
<b>4 Problem 4: Patent Program</b>	<b>9</b>
<b>5 Problem 5: MaxTemp Program</b>	<b>11</b>
<b>6 Problem 6: Average Salary Program</b>	<b>14</b>
<b>7 Problem 7: DelIdentify Healthcare Program</b>	<b>17</b>
<b>8 Problem 8: Music Track Program</b>	<b>19</b>
<b>9 Problem 9: Call Data Record</b>	<b>21</b>
<b>10 Problem 10: Connected Components</b>	<b>24</b>

- Author: [All in] group
- Date: 11/04/2023
- Subtitle: CSC14118 Introduction to Big Data 20KHMT1
- Language: English

## **Task progression**

No. of task	% completed
1 - 9	100%
10	30%

# 1 Problem 1: Word Count

**Input:** text file ‘word\_count.txt’

**Output:** the word count of particular words in the input file: the result as key-value pairs where the key represents the distinct words and the value denotes the number of times the word appears.

**Idea:** For each word, the mapper function extracts the token and emits a key-value pair where the key is the word and value is set to 1. The reducer reads through the values associated with each key, sums up these values to find the total count for the given word, and emits the final key-value pair representing the word and its respective count.

**Code: Mapper map() function:**

- Use `toString()` to convert it to a string format that `StringTokenizer` can work with.
- Each key is a distinct word extracted from the input data, and the value is an integer representation of the number 1 (one), write down (key-value) pairs to context.

```
StringTokenizer str = new StringTokenizer(value.toString());
while (str.hasMoreTokens()) {
    word.set(str.nextToken());
    context.write(word, one);
}
```

*reducer() function:* - initializing an integer variable called sum to 0, in order to keep track of the total number of times the given word appears across all the input files. - Iterating over all the values associated with a given key: For each iteration, this line adds the integer value of the current `val` object to the `sum` variable. This effectively aggregates the frequency counts assigned to each instance of the word, giving a total count for the entire input data set.

```
int sum = 0;
for (IntWritable val : values) {
    sum += val.get();
}
result.set(sum); // thiết lập giá trị của result thành sum
context.write(key, result);
```

## Run program

### Run:

```

id20120030@thienan:~/Desktop/Lab02/WordCount
8065 Jps
4277 SecondaryNameNode
4058 ResourceManager
4554 ResourceManager
3914 NameNode
4068 DataNode
4056 NodeManager
2023-04-11 01:37:47,192 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ls: output: No such file or directory
id20120030@thienan:~/Desktop/Lab02/WordCount$ hdfs dfs -ls /Lab02/WordCount/Output
2023-04-11 01:38:15,483 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ls: found 2 items
-rw-r--r-- 3 id20120030 supergroup 0 2023-04-11 01:28 /Lab02/WordCount/Output/_SUCCESS
-rw-r--r-- 3 id20120030 supergroup 113 2023-04-11 01:28 /Lab02/WordCount/Output/part-r-00000
2023-04-11 01:39:15,798 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
id20120030@thienan:~/Desktop/Lab02/WordCount$ jps
4277 SecondaryNameNode
4058 ResourceManager
3914 NameNode
40619 Jps
4068 DataNode
4056 NodeManager
2023-04-11 01:40:15,486 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Deleted /Lab02/WordCount/Output
id20120030@thienan:~/Desktop/Lab02/WordCount$ hdfs dfs -rm -r -f /Lab02/WordCount/Output
2023-04-11 01:40:21,114 INFO util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Deleted /Lab02/WordCount/Input/input.txt
id20120030@thienan:~/Desktop/Lab02/WordCount$ hadoop fs -put '/home/id20120030/Desktop/Lab02/WordCount/input_data/wordcount.txt' '/Lab02/WordCount/Input'
2023-04-11 01:47:34,576 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2023-04-11 01:48:17,469 INFO util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2023-04-11 01:48:18,267 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2023-04-11 01:49:18,838 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2023-04-11 01:49:18,890 INFO mapreduce.jobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/id20120030.staging/job_1681140538736_0004
2023-04-11 01:49:19,134 INFO mapreduce.Job: Job Id: job_1681140538736_0004
2023-04-11 01:49:19,322 INFO mapreduce.JobsSubmissionuler: number of splits:1
2023-04-11 01:49:19,625 INFO mapreduce.JobsSubmissionuler: Submitting tokens for Job: job_1681140538736_0004
2023-04-11 01:49:19,626 INFO mapreduce.JobsSubmissionuler: Executing with tokens: []
2023-04-11 01:49:19,627 INFO mapreduce.JobsSubmissionuler: Ignoring configuration entry types.xm
2023-04-11 01:49:19,674 INFO resource.Resourceutils: Unable to find 'resource-types.xml'.
2023-04-11 01:49:20,358 INFO impl.YarnClientImpl: Submitted application application_1681140538736_0004
2023-04-11 01:49:20,454 INFO mapreduce.Job: The url to track the job: http://thienan:8088/proxy/application_1681140538736_0004/
2023-04-11 01:49:27,659 INFO mapreduce.Job: Job job_1681140538736_0004 running in uber mode : false
2023-04-11 01:49:27,662 INFO mapreduce.Job: map 0% reduce 0%
2023-04-11 01:49:33,769 INFO mapreduce.Job: map 100% reduce 0%
2023-04-11 01:49:33,770 INFO mapreduce.Job: map 100% reduce 100%
2023-04-11 01:49:41,863 INFO mapreduce.Job: Job job_1681140538736_0004 completed successfully
2023-04-11 01:49:42,028 INFO mapreduce.Job: Counters: 59
File System Counters
FILE: Number of bytes read=1755

```

### Result:

```

id20120030@thienan:~/Desktop/Lab02/WordCount
File Output Format Counters
Bytes Written=184
id20120030@thienan:~/Desktop/Lab02/WordCount$ hdfs dfs -ls /Lab02/WordCount/Output
2023-04-11 01:49:05,911 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ls: found 2 items
-rw-r--r-- 3 id20120030 supergroup 0 2023-04-11 01:48 /Lab02/WordCount/Output/_SUCCESS
-rw-r--r-- 3 id20120030 supergroup 1184 2023-04-11 01:48 /Lab02/WordCount/Output/part-r-00000
id20120030@thienan:~/Desktop/Lab02/WordCount$ hadoop dfs -cat /Lab02/WordCount/Output/
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement 'hdfs dfs' instead.
2023-04-11 01:49:36,854 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
I
Infinite 1
Nobody 1
This 1
We 1
When 1
Whether 1
Worry 1
Years 1
Youth 2
a 11
adventure 1
aeritals 2
a 8
appetite 1
appetite, 1
are 4
as 3
at 2
back 1
beauty, 1
being's 1
c 4
bows 1
but 2
by 2
catch 1
center 1
cheeks, 1
cheer 1
child-like 1
courage 2
covered 1
cynicism 1
dear 1
deserting 1
die 1
down, 1
down't 1
dose, 1
ease. 1
eighty. 1
emotions; 1

```

## 2 Problem 2: Word Size Word Count

**Input:** a text file name “alphabets” contains a large amount of words from a book.

**Output:** list of word size and the number of occurrences for each word size. In short, how many words has size  $x$ , for  $x = 1, \dots, \text{inf}$

**Idea:** this is just some derivation from trivial Word Count program. But here for each word in the text, we have to count the lenght of that word. The key is the word size, and the value is 1 for each word has that size. Using MapReduce mechanism, we then merge the result of each word size to get the final result.

**Code:**

**Mapper** The map implementation is `WordSizeWordCountMapper()`. It receives each line of the text, as type `Text`. After that it emits a pair, with key is an integer show the word size - type `IntWritable` - and value is 1. It means that, the map function seperate words by whitespace, then for each word, it counts the number of letters in that word. At the end, it just found 1 word has some specific size, and it emits them a pair key-value described above.

But the word can have some special character as a result of sentence or paragraph. Example you., ha!, so, have ., ! and , attached to them, but normally, that can not be counted as some valid letter forms a word. To solve this problem in the map function, for each word we only keep a-zA-Z\_ letter and remove all others before we find the size of a word by using Regex<sup>1</sup>.

```
word.set( itr.nextToken().replaceAll("[^\\w]", "") );
```

**Reducer** The reduce implementation is `WordSizeWordCountReducer()`. It receives a pair `<size_of_word, [list_of_1]>` from the map functions. It means it takes one word size and list of 1s respect to how many words have that size. Now the reduce function only need to traverse the list of values and aggregate them. Finally, for each word size, the reduce emits a pair, with key is word size, and with value is the number of occurrences, with type `<IntWritable, IntWritable>`.

---

<sup>1</sup>A. Batkin, “Converting a sentence string to a string array of words in Java,” StackOverflow, 2014. <https://stackoverflow.com/a/4674887/11985028>.

```

int sum = 0;
for (IntWritable val : values) {
    sum += val.get();
}
result.set(sum);

```

20120090@thehoang: ~/BigDataLabs/Lab\_1/Lab\_2

File Edit View Search Terminal Help

2023-04-11 00:01:56,541 INFO mapred.LocalJobRunner: reduce task executor complete.

2023-04-11 00:01:56,611 INFO mapreduce.Job: Job job\_local608123517\_0001 running in uber mode : false

2023-04-11 00:01:56,612 INFO mapreduce.Job: map 100% reduce 100%

2023-04-11 00:01:56,613 INFO mapreduce.Job: Job job\_local608123517\_0001 completed successfully

2023-04-11 00:01:56,619 INFO mapreduce.Job: Counters: 30

File System Counters

- FILE: Number of bytes read=3113920
- FILE: Number of bytes written=1284434
- FILE: Number of read operations=0
- FILE: Number of large read operations=0
- FILE: Number of write operations=0

Map-Reduce Framework

- Map input records=33215
- Map output records=268117
- Map output bytes=2144936
- Map output materialized bytes=376
- Input split bytes=128
- Combine input records=268117
- Combine output records=37
- Reduce input groups=37
- Reduce shuffle bytes=376
- Reduce input records=37
- Reduce output records=37
- Spilled Records=74
- Shuffled Maps =1
- Failed Shuffles=0
- Merged Map outputs=1
- GC time elapsed (ms)=25
- Total committed heap usage (bytes)=404750336

Shuffle Errors

- BAD\_ID=0
- CONNECTION=0
- IO\_ERROR=0
- WRONG\_LENGTH=0
- WRONG\_MAP=0
- WRONG\_REDUCE=0

File Input Format Counters

- Bytes Read=1553116

File Output Format Counters

- Bytes Written=242

20120090@thehoang: ~/BigDataLabs/Lab\_1/Lab\_2\$ \_

20120090@thehoang: ~/BigDataLabs/Lab\_1/Lab\_2/output

File Edit View Search Terminal Help

GNU nano 6.2 part-r-00000

```

0      226
1     10739
2     42436
3     59034
4     47355
5     32884
6     23947
7     20053
8     12448
9     8178
10    5001
11    2903
12    1531
13    752
14    322
15    141
16    63
17    37
18    21
19    6
20    6
21    9
22    3
23    2
24    1
25    2
26    4
27    3
28    1
30    1
33    1
34    2
36    1
39    1
53    1
59    1
71    1

```

[ Read 37 lines ]

^G Help ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-[ To Bracket  
 ^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy M-Q Where Was

### 3 Problem 3: Weather Data

**Input:** data list including date, minimum temperature, maximum temperature and other information

**Output:** print “hot day” for days with the highest temperature greater than 40 degrees, and “cold days” for days with the lowest temperature less than 10 degrees.

**Idea:** extract information like date, minimum temperature, maximum temperature. Check for the day has the highest temperature > 40 degrees, the lowest temperature < 10 degrees, then save the date information.

**Code:**

*Mapper map() function*

- Use the substring() command to get the date and temperature information
- Convert the temperature from string to real number, check the min and max temperature according to the problem requirements and write the result to the context variable.
- Use 2 if functions to check in case a day has minimum temperature less than 10 and maximum temperature greater than 40.

```
Text date = new Text();
Text word = new Text();

String line = value.toString();
String d = line.substring(14, 22);
float max = Float.parseFloat(line.substring(104, 108).trim());
float min = Float.parseFloat(line.substring(112, 116).trim());

if (max > 40.0) {
    date.set(d);
    word.set("Hot day");
    context.write(date, word);
}

if (min < 10.0) {
    date.set(d);
    word.set("Cold day");
    context.write(date, word);
}
```

**Reducer** *reduce()* function

- Record the output results, if there is a case in the same day that the lowest and highest temperatures are satisfied, then combine “cold day” and “hot day” into 1 result

```
Text result = new Text();
String res = "";
for (Text val : values) {
    res = res + val + " ";
}
result.set(res);
context.write(key, result);
```

## 4 Problem 4: Patent Program

**Input:** Each patent has sub-patent ids associated with it.

**Output:** The number of sub-patent associated with each patent.

**Idea:** Just look like WordCount program.

**Map:** Split lines, get 2 tokens and write them to context.

```
public void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line, " ");
    while (tokenizer.hasMoreTokens()) {
        String pat = tokenizer.nextToken();
        k.set(pat);
        String sub = tokenizer.nextToken();
        v.set(sub);
        context.write(k, v);
    }
}
```

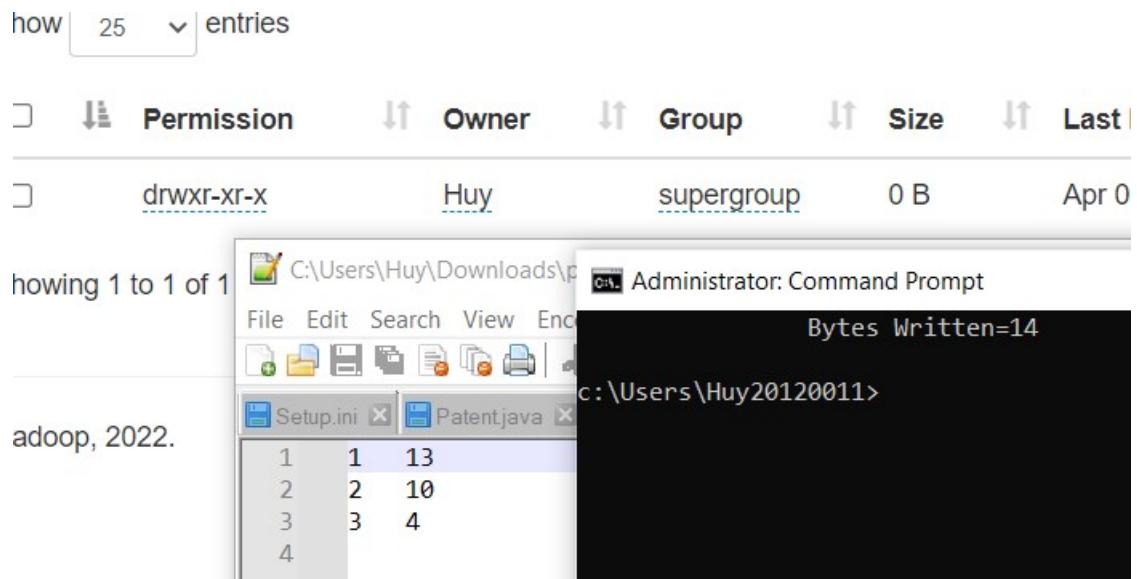
**Reduce:** Count the number of sub-patent associated with each patent.

```
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
    InterruptedException {
    int sum = 0;
    for (Text x : values) {
        sum++;
    }
    context.write(key, new IntWritable(sum));
}
```

**Run:**

```
c:\Users\Huy20120011>hadoop jar D:\Homework\Problem4\pt.jar Patent /input/patent.txt /output/patent.txt
2023-04-05 19:13:11,394 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-04-05 19:13:11,652 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement Job.getConfiguration(JobContext) to use Hadoop configuration from main().
```

**Result:**



## 5 Problem 5: MaxTemp Program

**Input:** a subset of temperature records written in text file ‘temperature.txt’. Each line is a year and a temperature record associated with it. Each year has multiple temperatures listed.

**Output:** find the maximum temperature during a year. Each line is a year and the maximum temperature record associated with it.

**Idea:** From the LabRequirement.pdf: tokenizes the input data and outputs key-value pairs where the key is the year and the value is the corresponding temperature. The reducer takes these key-value pairs and aggregates them for each year, finding the maximum temperature for each year as it iterates through the values associated with each key.

**Code: Mapper map() function:**

- Use `toString()` to convert input to a string format that `StringTokenizer` can work with.
- Breaking each record according to the delimiter whitespace
- For each iteration: set the value for `year` from the first token, and set the value for `temp` from the second one. After that, we convert `temp` (string type) into `temper` (integer type).
- Each key is a distinct year extracted from the input data, and the value is an integer representation of temperature, write down (key-value) pairs to context.

```
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line, " ");

while (tokenizer.hasMoreTokens()) {
    String year = tokenizer.nextToken();
    word.set(year);

    String temp = tokenizer.nextToken().trim();
    int temper = Integer.parseInt(temp);
    context.write(word, new IntWritable(temper));
}
```

**reducer() function:** - initializing an integer variable called `maxTemp` to 0, in order to keep track of the maximum temperature of each year appears across all the input files. - Iterating over all the values associated with a given key: For each iteration, we define a local variable ‘temperature’ of type

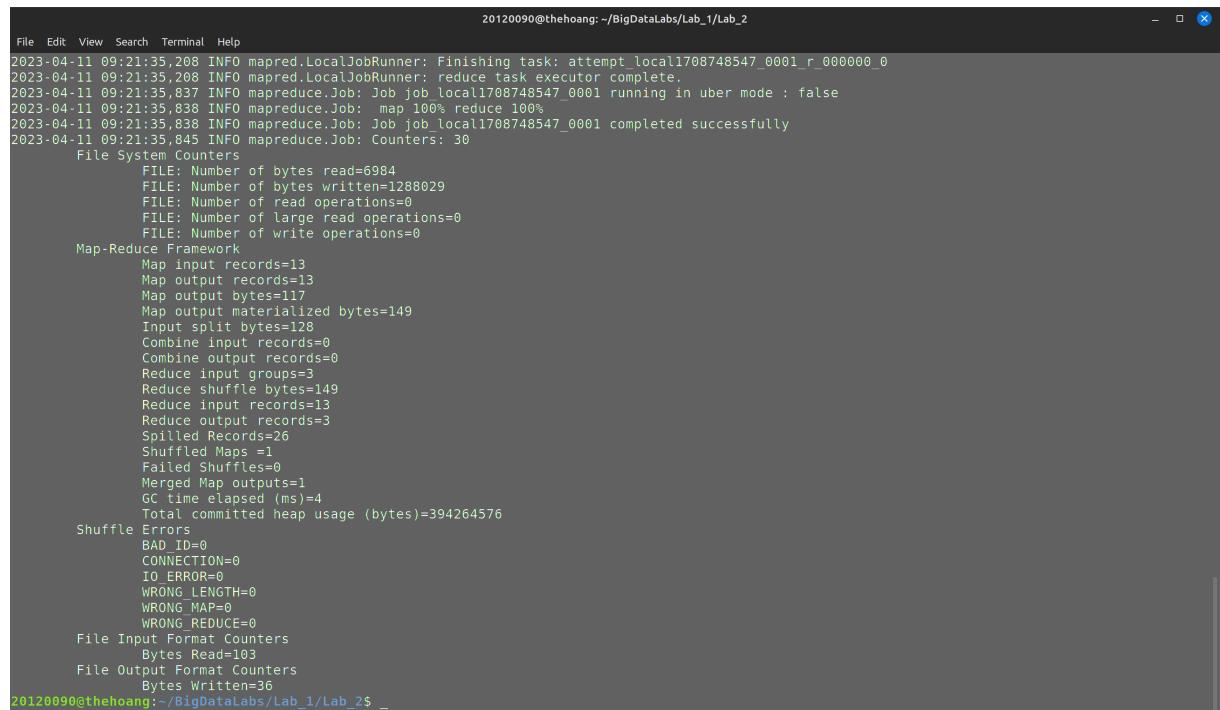
int which is taking all the temperature that belong to that year. Then we compare each element in temperature with maxtemp to update maxtemp with the new value greater than it.

```
int maxtemp = 0;
for(IntWritable it : values) {
    int temperature = it.get();
    if(maxtemp < temperature) {
        maxtemp = temperature;
    }
}

context.write(key, new IntWritable(maxtemp));
```

## Run program

### Run:



The screenshot shows a terminal window titled '20120090@thehoang: ~/BigDataLabs/Lab\_1/Lab\_2'. The window displays the output of a MapReduce job. The logs show the job starting, tasks being completed, and finally the job finishing successfully. Below the logs, detailed system and map-reduce framework counters are listed, such as file operations, map and reduce records, and shuffle errors. The terminal window has a dark theme with light-colored text.

```
File Edit View Search Terminal Help
20120090@thehoang: ~/BigDataLabs/Lab_1/Lab_2
2023-04-11 09:21:35,208 INFO mapred.LocalJobRunner: Finishing task: attempt local1708748547_0001_r_000000_0
2023-04-11 09:21:35,208 INFO mapred.LocalJobRunner: reduce task executor complete.
2023-04-11 09:21:35,837 INFO mapreduce.Job: Job job_local1708748547_0001 running in uber mode : false
2023-04-11 09:21:35,838 INFO mapreduce.Job: map 100% reduce 100%
2023-04-11 09:21:35,838 INFO mapreduce.Job: Job job_local1708748547_0001 completed successfully
2023-04-11 09:21:35,845 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=6984
    FILE: Number of bytes written=1288029
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  Map-Reduce Framework
    Map input records=13
    Map output records=13
    Map output bytes=117
    Map output materialized bytes=149
    Input split bytes=128
    Combine input records=0
    Combine output records=0
    Reduce input groups=3
    Reduce shuffle bytes=149
    Reduce input records=13
    Reduce output records=3
    Spilled Records=26
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=4
    Total committed heap usage (bytes)=394264576
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=103
  File Output Format Counters
    Bytes Written=36
20120090@thehoang: ~/BigDataLabs/Lab_1/Lab_2$ _
```

### Result:

20120090@thehoang: ~/BigDataLabs/Lab\_1/Lab\_2/output

GNU nano 6.2

part-r-00000

```
1900 36
1901 48
1902 49
```

[ Read 3 lines ]

^G Help ^O Write Out ^W Where Is ^K Cut ^C Location M-U Undo  
^X Exit ^R Read File ^A Replace ^U Paste ^T Execute ^J Justify ^Y Go To Line M-E Redo  
M-A Set Mark M-6 Copy M-] To Bracket ^Q Where Was

## 6 Problem 6: Average Salary Program

**Input:** the input file is taken from <sup>1</sup>. The file is written in csv format, with using "" as escaped character for , in case of long string. There are 13 fields. In 13 fields, there are 3 empty fields. Based on id field, there are 148654 distinct employees. For this task, we take into account 2 fields: JobTitle and TotalPayBenefits. We will count for each distinct JobTitle, what is the average total pay benefits the employees in that job title can receive. Based on data, there are 2159 distinct JobTitle.

**Output:** for each JobTitle, employees in that field receive how much total pay benefits (salary) in average.

**Idea:** for each JobTitle, we only need to take all total pay benefits of all people working in that job title, and create a pair key-value. After that, for each JobTitle, we calculate the average values.

**Code:**

**Mapper** The map implementation is AverageSalaryMapper(). The map function takes each line as input with Text type. It then uses separate the data as csv format by this statements, using Regex (we need to take care the case there is , in the job title, so we used Regex). <sup>2</sup>.

```
String[] columns = valueStr.split(",(?=([^\\\"]*\"[^\\\"]*\"))*[^\\\"]*$)", -1);
```

The map function then just emit the pair <Text, FloatWritable> takes from separated data. The key is the JobTitle and the value is TotalPayBenefits of an employee. Here the value of salary is float number, so we use FloatWritable for that data.

But because the first line of file contains the header title of the data, we need to use try...catch... to prevent the case the text are converted to float number.

```
try {
    context.write( new Text(columns[2]), new FloatWritable(Float.parseFloat(columns[8])) );
} catch (Exception e) {
```

<sup>1</sup>A. Batkin, “Converting a sentence string to a string array of words in Java,” StackOverflow, 2014. <https://stackoverflow.com/a/4674887/11985028>.

<sup>2</sup>Baeldung, “Ignoring Commas in Quotes When Splitting a Comma-separated String,” 2021. <https://www.baeldung.com/java-split-string-commas>.

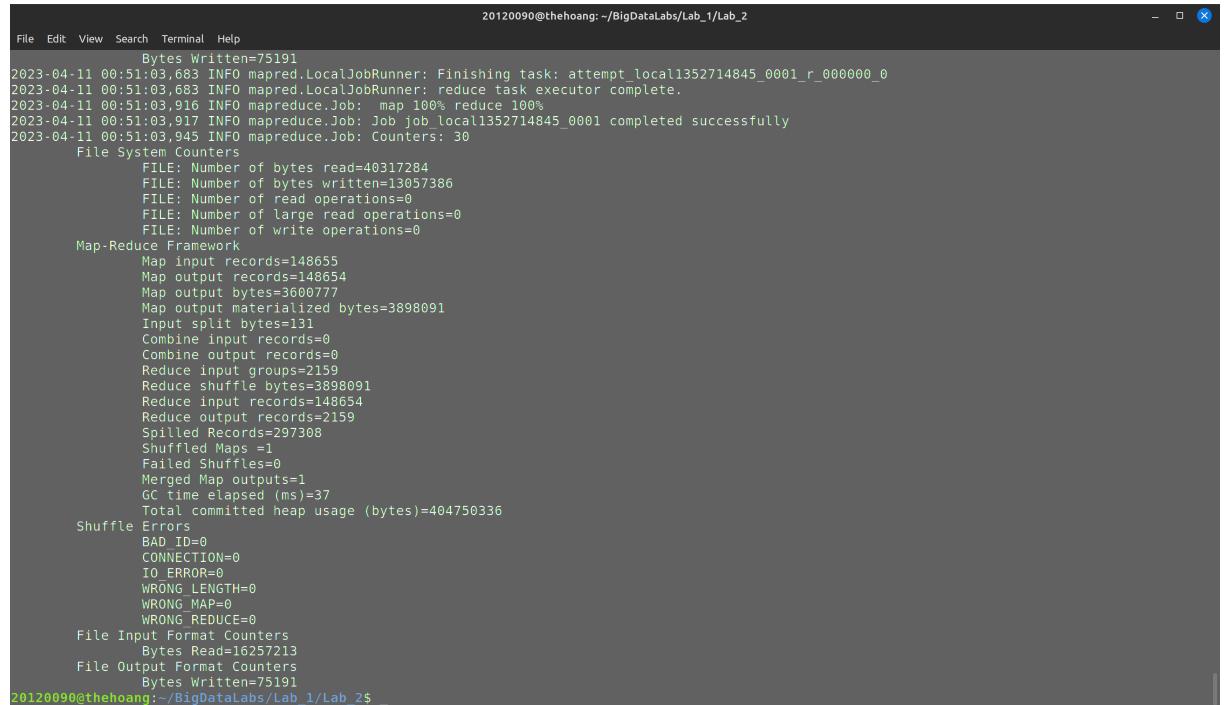
```
    return;
}
```

**Reducer** The reduce implementation is `AverageSalaryReducer()`. It receives pairs `<JobTitle, [list_of_salary_values]>`. It means it takes a job title and list of salary values of all people working in that position. Here we only need to traverse the list of values, then add that value to the current computed average salary value. The detailed implementation is shown below:

```
float average = 0;
int cnt = 0;

for (FloatWritable val : values) {
    ++cnt;
    average = average + (val.get() - average)/cnt;
}
```

It emits the results as `<JobTitle, Average_value_computed_of_that_jobtitle>` as type `<Text, FloatWritable>`.



```
File Edit View Search Terminal Help
Bytes Written=75191
2023-04-11 00:51:03,683 INFO mapred.LocalJobRunner: Finishing task: attempt_local1352714845_0001_r_000000_0
2023-04-11 00:51:03,683 INFO mapred.LocalJobRunner: reduce task executor complete.
2023-04-11 00:51:03,916 INFO mapreduce.Job: map 100% reduce 100%
2023-04-11 00:51:03,917 INFO mapreduce.Job: Job job_local1352714845_0001 completed successfully
2023-04-11 00:51:03,945 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=40317284
    FILE: Number of bytes written=13057386
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  Map-Reduce Framework
    Map input records=148655
    Map output records=148654
    Map output bytes=3600777
    Map output materialized bytes=3898091
    Input split bytes=131
    Combine input records=0
    Combine output records=0
    Reduce input groups=2159
    Reduce shuffle bytes=3898091
    Reduce input records=148654
    Reduce output records=2159
    Spilled Records=297368
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=37
    Total committed heap usage (bytes)=404750336
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=16257213
  File Output Format Counters
    Bytes Written=75191
2023-04-11 00:51:03,945 INFO mapreduce.Job: Counters: 30
```

```

20120090@thehoang: ~/BigDataLabs/Lab_1/Lab_2/output
File Edit View Search Terminal Help part-r-00000
GNU nano 6.2
"ACPO,JuvP, Juv Prob (SFERS)" 80266.37
"ADMINISTRATOR, SFGH MEDICAL CENTER" 257124.44
"AIRPORT ASSISTANT DEPUTY DIRECTOR, BUSINESS ADMINI" 1927.5
"AIRPORT ASSISTANT DEPUTY DIRECTOR, OPERATIONS" 15420.0
"APPRENTICE STATIONARY ENGINEER, SEWAGE PLANT" 71108.445
"APPRENTICE STATIONARY ENGINEER,WATER TREATMENT PLN" 59027.805
"AREA SUPERVISOR, PARKS, SQUARES AND FACILITIES" 88242.32
"ASSISTANT CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)" 203427.84
"ASSISTANT CLERK, BOARD OF SUPERVISORS" 73480.414
"ASSISTANT DEPUTY DIRECTOR, PORT" 132242.5
"ASSISTANT DIRECTOR, JUVENILE HALL" 20717.65
"ASSISTANT DIRECTOR, LOG CABIN RANCH" 81589.92
"ASSISTANT RENTAL MANAGER, PORT" 81015.79
"ASSISTANT TO THE DIRECTOR, PUBLIC AFFAIRS RS" 95283.35
"ASSOCIATE MUSEUM CONSERVATOR, ASIAN ART MUSEUM" 72844.4
"ASST. CHIEF, BUREAU OF CLAIMS INVEST. & ADMIN" 135166.98
"ATTORNEY, TAX COLLECTOR" 87465.53
"Adm, SFGH Medical Center" 347079.7
"Administrator, DPH" 331564.03
"AprrntcStatnry Eng, Sew Plant" 92028.266
"AprrntcStatnry Eng,WtrTreatPlnt" 106728.805
"Area Spr Parks, Squares & Fac" 137362.42
"Assistant Director, Probate" 110758.41
"Assoc Musm Cnsrvt, AAM" 115269.695
"Ass't Chf Prob Ofc, Juv Prob" 231537.98
"Ass't Chf, Bur Clm Invest&Admin" 197832.8
"Ass't Clk, Board of Supervisors" 127252.234
"Ass't Dir, Log Cabin Rnch" 131029.29
"Ass't Director, Juvenile Hall" 74674.125
"Ass't.Dep.Dir., Port" 191966.03
"Attorney, Tax Collector" 208386.16
"BATTALION CHIEF, (FIRE DEPARTMENT)" 216655.53
"BOARD COMMISSION MEMBER, $200 PER MEETING" 9571.429
"BOARD/COMMISSION MEMBER, GROUP II" 296.5116
"BOARD/COMMISSION MEMBER, GROUP III" 638.78784
"BOARD/COMMISSION MEMBER, GROUP V" 1195.9049
"Battalion Chief, Fire Suppress" 306954.53
"Battlion Chief, Fire Suppress" 268663.44
^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File    ^L Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo      M-6 Copy       ^Q Where Was

```

## 7 Problem 7: DeIdentify Healthcare Program

**Input:** the input file is described in the given document <sup>1</sup>. We only create the dataset includes 6 records, which are same as in the above document.

**Output:** we specify the fields we want to encrypt. Here we only keep the 1st, 7th and 9th fields. The others will be encrypted by the SHA-256 encryption algorithm. The final datas need to have the same format as original file.

**Idea:** we only use one map function to encrypt required fields. The results from this function will also be the final result. We need to choose the field for encryption carefully while keep other fields untouched. The whole idea is taken from<sup>2</sup>.

**Code:**

**Mapper** The map implementation is DeIdentifyHealthcareMapper. The map function takes each line as Text type, separate words by , delimiter. There is no special things here, we check if current field need to be encrypted. If it is, we call toHexString() and getSHA() function to encrypt data as SHA-256 implementation.

We need to take care that, if we have not any result word yet, we must not add , to the result word to keep the original data format.

```
if (newStr.length() > 0) newStr += ",";
...
if (newStr.length() > 0) newStr += ",";
```

The map function after encrypted a line of data will emit the pair result as <null, encrypted\_text>. Here, we do not need any key so we set it null, the value is the encrypted text.

---

<sup>1</sup>S. Balasubramanian, “Hadoop-MapReduce Lab.” p. 27, 2016.

<sup>2</sup>S. Balasubramanian, “Hadoop-MapReduce Lab.” p. 27, 2016.

```
2020090@thehoang:~/BigDataLabs/Lab_1/Lab_2
```

File Edit View Search Terminal Help

```
2023-04-11 01:12:35,488 INFO mapred.LocalJobRunner: Finishing task: attempt local1019703260_0001_r_000000
2023-04-11 01:12:35,489 INFO mapred.LocalJobRunner: reduce task executor complete.
2023-04-11 01:12:35,902 INFO mapreduce.Job: Job job_local1019703260_0001 running in uber mode : false
2023-04-11 01:12:35,903 INFO mapreduce.Job: map 100% reduce 100%
2023-04-11 01:12:35,904 INFO mapreduce.Job: Job job_local1019703260_0001 completed successfully
2023-04-11 01:12:35,912 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=12574
    FILE: Number of bytes written=1297124
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  Map-Reduce Framework
    Map input records=6
    Map output records=6
    Map output bytes=2418
    Map output materialized bytes=2448
    Input split bytes=128
    Combine input records=0
    Combine output records=0
    Reduce input groups=1
    Reduce shuffle bytes=2448
    Reduce input records=6
    Reduce output records=6
    Spilled Records=12
    Shuffled Maps =1
    Failed Shuffless=0
    Merged Map outputs=1
    GC time elapsed (ms)=22
    Total committed heap usage (bytes)=394264576
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=396
  File Output Format Counters
    Bytes Written=2434
2020090@thehoang:~/BigDataLabs/Lab_1/Lab_2$
```

```
2020090@thehoang:~/BigDataLabs/Lab_1/Lab_2/output
```

File Edit View Search Terminal Help

```
GNU nano 6.2          part-r-00000
```

```
11116,6de05335d3b3fc38b86507ab67a35e408e6c1bdb0b0ee8f6c06cf1878f8db4,c8fb40aa89c32b47268092d2478a061ec91e34b21c8b072123e21c338bdः,87a8e469e>
11115,e8c330e955d0f7ef3bea25c50019ee357d10a8ffeb27d20e1679c220f9158aa7ff,f1749336339a5126f57e35a8f468bb732ff5b7ae1bfff0fae754094b3c56642,87a8e469e>
11114,680e5c27445df32dd13016dcda0e7d8cb6e0cb022fa2fb1e7db6de3565d6d0d5,5b0569f7f358f54a6b66aca4c36d4edb5db58c9200811287448a3dc4895c42f,87a8e469e>
11113,9c73a1a2461934a8c0dc0da9fecc91de965e8c7f8667dfeafbdde73e4d2604,c03b5b84e9347ab129170ba1a1717ea4947308880ce92d6e55cc3e466fc58,87a8e469e>
11112,7874b5a9c649d/a46/b1f208498f865/83c9a2078/264ab1c9ebdd2f2a83852,8fb6cbfb9529b/91ec25084916ea96f0d932f150311bc519e42d8ae9008034e,87a8e469e>
11111,015c0eb43bc3715e8295a91fc029deaffe3ca383904e12b4f00021f85773d2942375278cb8621a22c58755b0f774ba0c4d5b9e4ff39e8f7e624f3e,87a8e469e>
```

^G Help ^W Write Out ^W Where Is ^K Cut ^U Paste [ Read 6 lines ] ^T Execute ^C Location M-U Undo M-A Set Mark M-B Copy M-Q To Bracket M-Q Where Was

## 8 Problem 8: Music Track Program

**Input:** UserId, TrackId, Shared, Radio, Skip

**Output:** TrackID, the number of unique users

**Idea:** I reference to the solution and github.

<https://gist.github.com/deshpandetanmay/70277a43dc9332819c93>

**Map:**

Split lines, if it is a valid record then write to context

```
public static class UniqueListenerMapper extends
    Mapper<Object, Text, IntWritable, IntWritable> {

    IntWritable trackId = new IntWritable();
    IntWritable userId = new IntWritable();

    public void map(Object key, Text value,
                    Mapper<Object, Text, IntWritable, IntWritable>.Context context)
        throws IOException, InterruptedException {

        String[] parts = value.toString().split("[|]");
        trackId.set(Integer.parseInt(parts[LastFMConstants.TRACK_ID]));
        userId.set(Integer.parseInt(parts[LastFMConstants.USER_ID]));

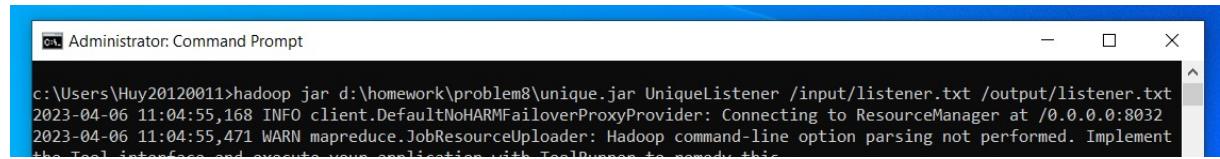
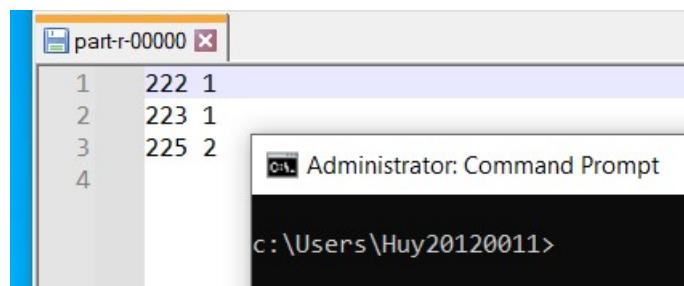
        if (parts.length == 5) {
            context.write(trackId, userId);
        } else {
            // add counter for invalid records
            context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);
        }
    }
}
```

**Reduce:** Count the number of unique users

```
public static class UniqueListenerReducer extends
    Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
```

```
public void reduce(
    IntWritable trackId,
    Iterable<IntWritable> userIds,
    Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)
    throws IOException, InterruptedException {

    Set<Integer> userIdSet = new HashSet<Integer>();
    for (IntWritable userId : userIds) {
        userIdSet.add(userId.get());
    }
    IntWritable size = new IntWritable(userIdSet.size());
    context.write(trackId, size);
}
```

**Run:****Result:**

## 9 Problem 9: Call Data Record

**Input:** list of data, each line includes caller phone number, recipient phone number, call start time, call end time, STD flag, each information separated by vertical bar ( | ) Consider following format  
FromPhoneNumber|ToPhoneNumber|CallStartTime|CallEndTime|STDFlag

**Output:** all phone numbers who are making more than 60 minutes of STD calls.

**Idea:** extract caller phone number, call start and end time from data. Calculate the total call time of each phone number, save all the phone numbers have total time more than 60 minutes.

**Code: Mapper map() function:**

- Use split() command to cut string, save information into an array
- For the phone number has STD call, use get\_time() function to get time in milliseconds of call start time and call end time, call time in millisec is saved to dur, in minutes is saved to duration

```
Text phone_num = new Text();
LongWritable duration = new LongWritable();

String[] word = value.toString().split("[|]");

if (word[4].equalsIgnoreCase("1")) {
    phone_num.set(word[0]);
    String call_end = word[3];
    String call_start = word[2];
    long dur = get_time(call_end) - get_time(call_start); //dur: time in millisecond
    duration.set(dur / (1000 * 60)); //duration: time in minute
    context.write(phone_num, duration);
}
```

**get\_time() function:** - Use SimpleDateFormat library to convert string data to Date data then use getTime() command to get time (in millisec)

```
private long get_time(String str) {
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date date = null;
    try {
        date = formatter.parse(str);
    } catch (ParseException e) {
```

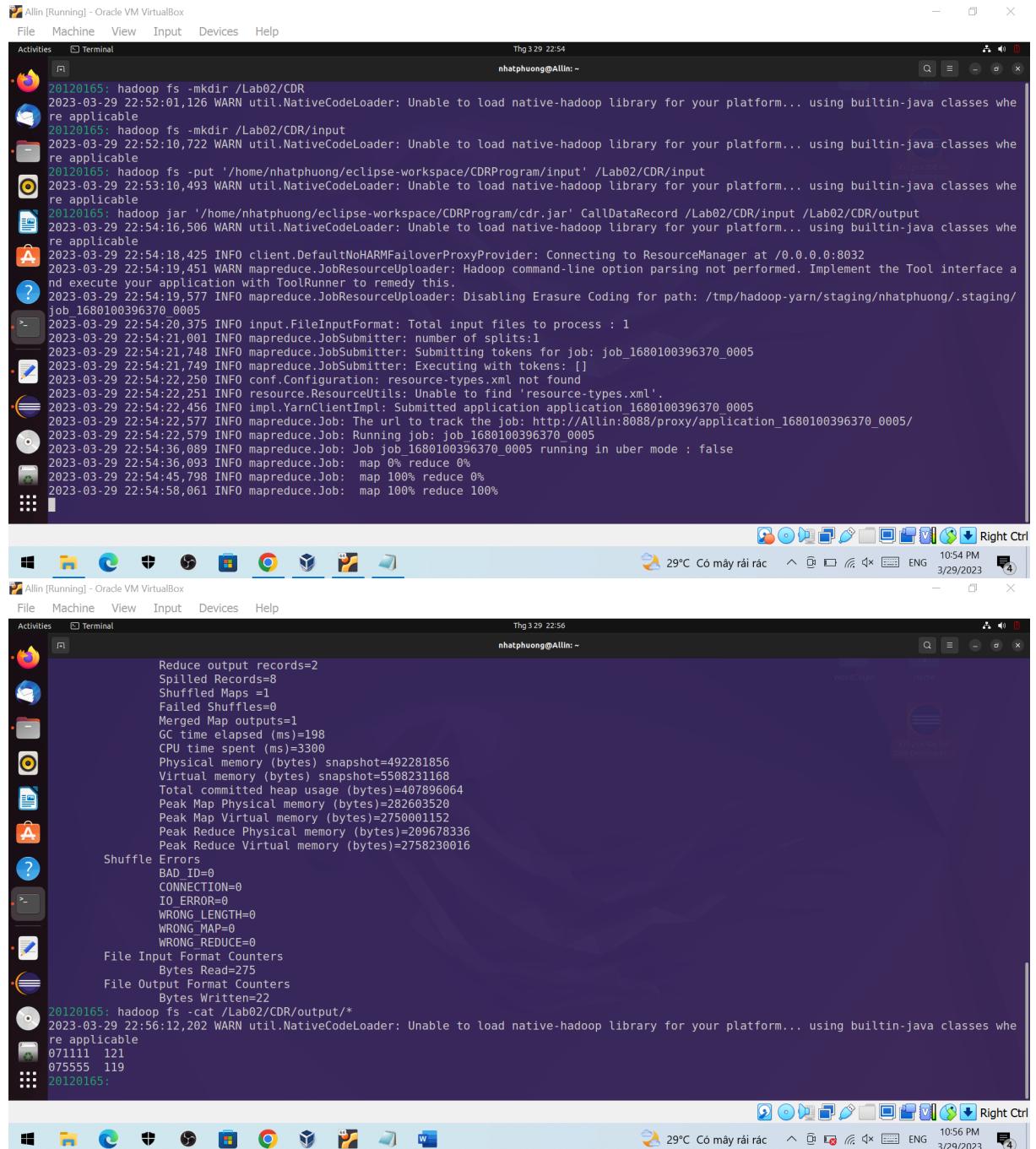
```
        e.printStackTrace();
    }
    return date.getTime();
}
```

*reducer() function:* - Calculate the total call time of 1 phone number, if total time is more than 60 minutes, print the result.

```
LongWritable result = new LongWritable();
long sum = 0;
for (LongWritable val : values) {
    sum += val.get();
}
result.set(sum);
if (sum >= 60) {
    context.write(key, result);
}
```

### Run program

071111 082982 2023-03-29 10:00:00 2023-03-29 12:01:00 1
033333 076234 2023-03-29 10:00:00 2023-03-29 10:28:00 1
094444 087659 2023-03-29 10:00:00 2023-03-29 13:00:00 0
075555 094444 2023-03-29 10:00:00 2023-03-29 10:30:00 1
075555 037290 2023-03-29 12:50:50 2023-03-29 13:00:00 1
039899 037290 2023-03-29 22:33:10 2023-03-29 22:40:00 1
036929 086273 2023-03-29 11:00:00 2023-03-29 12:29:00 0
072942 093723 2023-03-29 20:10:02 2023-03-29 20:30:00 0
092740 072942 2023-03-29 05:40:02 2023-03-29 05:45:00 0



The image shows a Linux desktop environment with two terminal windows running on an Oracle VM VirtualBox machine named "Allin".

**Terminal Window 1 (Top):**

```

20120165: hadoop fs -mkdir /Lab02/CDR
2023-03-29 22:52:01,126 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
20120165: hadoop fs -mkdir /Lab02/CDR/input
2023-03-29 22:52:10,722 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
20120165: hadoop fs -put '/home/nhatphuong/eclipse-workspace/CDRProgram/input' /Lab02/CDR/input
2023-03-29 22:53:10,493 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
20120165: hadoop jar '/home/nhatphuong/eclipse-workspace/CDRProgram/cdr.jar' CallDataRecord /Lab02/CDR/input /Lab02/CDR/output
2023-03-29 22:54:16,506 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
2023-03-29 22:54:18,425 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-03-29 22:54:19,451 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2023-03-29 22:54:19,577 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/nhatphuong/.staging/
job_1680100396370_0005
2023-03-29 22:54:20,375 INFO input.FileInputFormat: Total input files to process : 1
2023-03-29 22:54:21,001 INFO mapreduce.JobSubmitter: number of splits:1
2023-03-29 22:54:21,748 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1680100396370_0005
2023-03-29 22:54:21,749 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-03-29 22:54:22,250 INFO conf.Configuration: resource-types.xml not found
2023-03-29 22:54:22,251 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-03-29 22:54:22,456 INFO impl.YarnClientImpl: Submitted application application_1680100396370_0005
2023-03-29 22:54:22,577 INFO mapreduce.Job: The url to track the job: http://Allin:8088/proxy/application_1680100396370_0005/
2023-03-29 22:54:22,579 INFO mapreduce.Job: Running job: job_1680100396370_0005
2023-03-29 22:54:36,089 INFO mapreduce.Job: Job job_1680100396370_0005 running in uber mode : false
2023-03-29 22:54:36,093 INFO mapreduce.Job: map 0% reduce 0%
2023-03-29 22:54:45,798 INFO mapreduce.Job: map 100% reduce 0%
2023-03-29 22:54:58,061 INFO mapreduce.Job: map 100% reduce 100%

```

**Terminal Window 2 (Bottom):**

```

Reduce output records=2
Spilled Records=8
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=198
CPU time spent (ms)=3300
Physical memory (bytes) snapshot=492281856
Virtual memory (bytes) snapshot=5508231168
Total committed heap usage (bytes)=497996064
Peak Map Physical memory (bytes)=282603520
Peak Map Virtual memory (bytes)=2750001152
Peak Reduce Physical memory (bytes)=209678336
Peak Reduce Virtual memory (bytes)=2758230016
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=275
File Output Format Counters
Bytes Written=22
20120165: hadoop fs -cat /Lab02/CDR/output/*
2023-03-29 22:56:12,202 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
071111 121
075555 119
20120165:

```

## 10 Problem 10: Connected Components

We use the idea from <sup>1</sup>.

---

<sup>1</sup>V. Rastogi, A. Machanavajjhala, L. Chitnis, and A. Das Sarma, “Finding Connected Components on Map-reduce in Logarithmic Rounds,” Mar. 2012, [Online]. Available: <http://arxiv.org/abs/1203.5387>.