Roll No. 33313

Program:

```
package PackageDemo;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class LoginTimeMinMax {
   public static void main(String[] args) throws Exception {
      Configuration c = new Configuration();
      String[] files = new GenericOptionsParser(c, args).getRemainingArgs();
      Path input = new Path(files[0]);
      Path output = new Path(files[1]);
      Job j = Job.getInstance(c, "logInOutTime");
      j.setJarByClass(LoginTimeMinMax.class);
      j.setMapperClass(MapForWordCount.class);
      j.setReducerClass(ReduceForWordCount.class);
      j.setOutputKeyClass(Text.class);
      j.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(j, input);
      FileOutputFormat.setOutputPath(j, output);
      System.exit(j.waitForCompletion(true) ? 0 : 1);
   }

   public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable> {
      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String alllines = value.toString();
         String[] lines = alllines.split("\n");
         for (String line : lines) {
            String[] word = line.split(",");
            if (word.length > 7) {
               String login = word[5];
               String logout = word[7];
               String ip = word[1];
               String[] loginPart = login.split(" ");
               String[] logoutPart = logout.split(" ");
               if (loginPart.length > 1 && logoutPart.length > 1) {
                  String[] loginTimePart = loginPart[1].split(":");
                  String[] logoutTimePart = logoutPart[1].split(":");
                  int loginHour = Integer.parseInt(loginTimePart[0]);
```

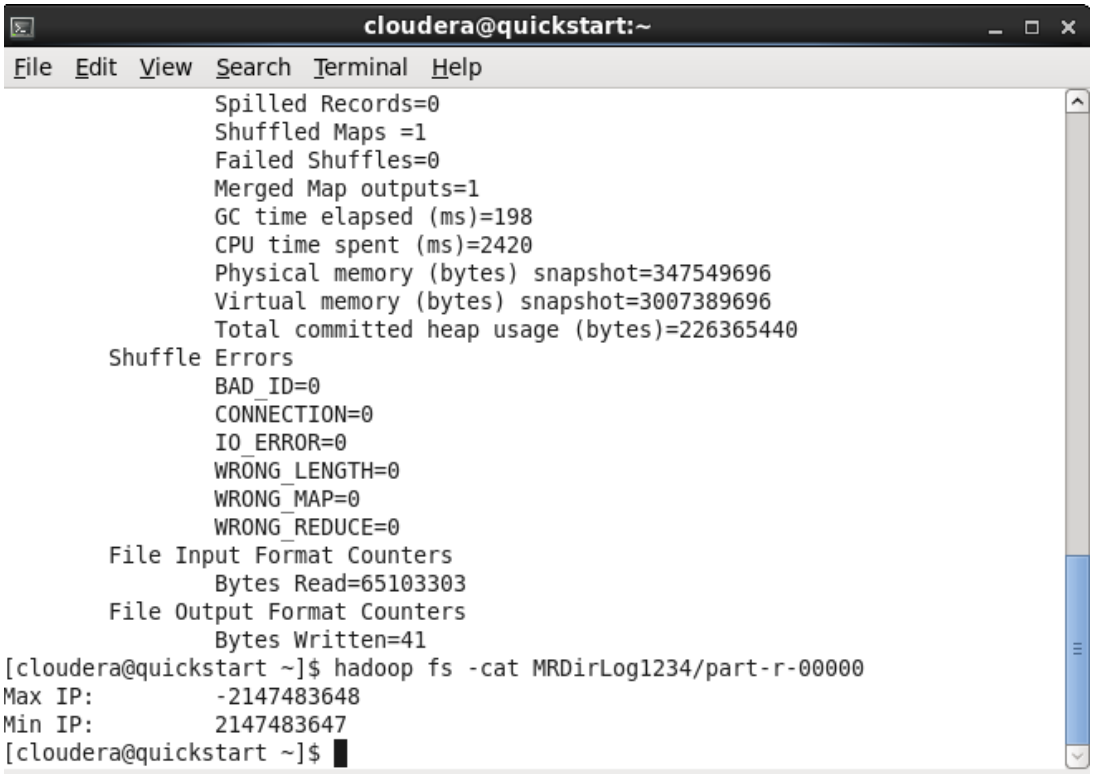
```
                    int loginMinute = Integer.parseInt(loginTimePart[1]);
                    int logoutHour = Integer.parseInt(logoutTimePart[0]);
                    int logoutMinute = Integer.parseInt(logoutTimePart[1]);
                    int loginTotalMinutes = loginHour * 60 + loginMinute;
                    int logoutTotalMinutes = logoutHour * 60 + logoutMinute;
                    int differenceInSeconds = (logoutTotalMinutes - loginTotalMinutes) * 60;
                    Text outputKey = new Text(ip.toUpperCase().trim());
                    IntWritable outputValue = new IntWritable(differenceInSeconds);
                    context.write(outputKey, outputValue);
                }
            }
        }
    }

    public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
        private int maxTime = Integer.MIN_VALUE;
        private int minTime = Integer.MAX_VALUE;
        private Text maxIp = new Text();
        private Text minIp = new Text();

        public void reduce(Text word, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
            int localMax = Integer.MIN_VALUE;
            int localMin = Integer.MAX_VALUE;
            for (IntWritable value : values) {
                int loginTime = value.get();
                if (loginTime > localMax) {
                    localMax = loginTime;
                }
                if (loginTime < localMin) {
                    localMin = loginTime;
                }
            }
            if (localMax > maxTime) {
                maxTime = localMax;
                maxIp.set(word);
            }
            if (localMin < minTime) {
                minTime = localMin;
                minIp.set(word);
            }
        }

        protected void cleanup(Context context) throws IOException, InterruptedException {
            context.write(new Text("Max IP: " + maxIp.toString()), new IntWritable(maxTime));
            context.write(new Text("Min IP: " + minIp.toString()), new IntWritable(minTime));
        }
    }
}
```

Output:


cloudera@quickstart:~
File Edit View Search Terminal Help
Spilled Records=0
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=198
CPU time spent (ms)=2420
Physical memory (bytes) snapshot=347549696
Virtual memory (bytes) snapshot=3007389696
Total committed heap usage (bytes)=226365440
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=65103303
File Output Format Counters
Bytes Written=41
[cloudera@quickstart ~]$ hadoop fs -cat MRDirLog1234/part-r-00000
Max IP: -2147483648
Min IP: 2147483647
[cloudera@quickstart ~]$