# Big Data and Large Scale Computing

Lab Report -02

July 21, 2025

**Name : Rohan Baghel**
**Student ID: 202116011**

## Question 1

To modification the "WordCount.java" program and run the same input file as in 1st lab using hadoop libraries and observe the output.

**Answer :**

We have done some modification in "WordCount.java" program to get the output format as.

- Tokenizing strings based on \t\n\r\f , . : ; ? ! [ ] '
- All the word are in small letters
- Counting and storing only those words which occurs more than 4 times in the document.

<u>Modification:</u>

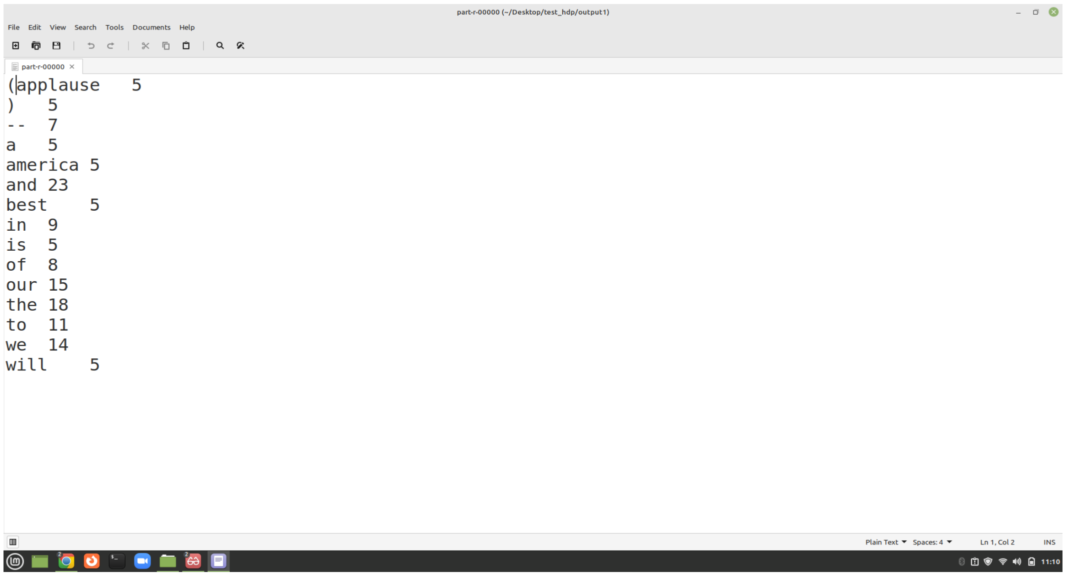
1. In place of StringTokenizer itr = new StringTokenizer(line); use
   StringTokenizer itr = new StringTokenizer(line,"\t\n\r\f , . : ; ? ! [ ] ' ");
2. In place of word.set(itr.nextToken()); use
   word.set(itr.nextToken().toLowerCase());
3. While storing data add a condition
   if (sum > 4) output.collect(key, new IntWritable(sum)); it will sore the words which occurs more than 4 times.

**File Name: WordCount.java**

This java WordCount.java program is modified version on the the lab1 WordCount;java program.

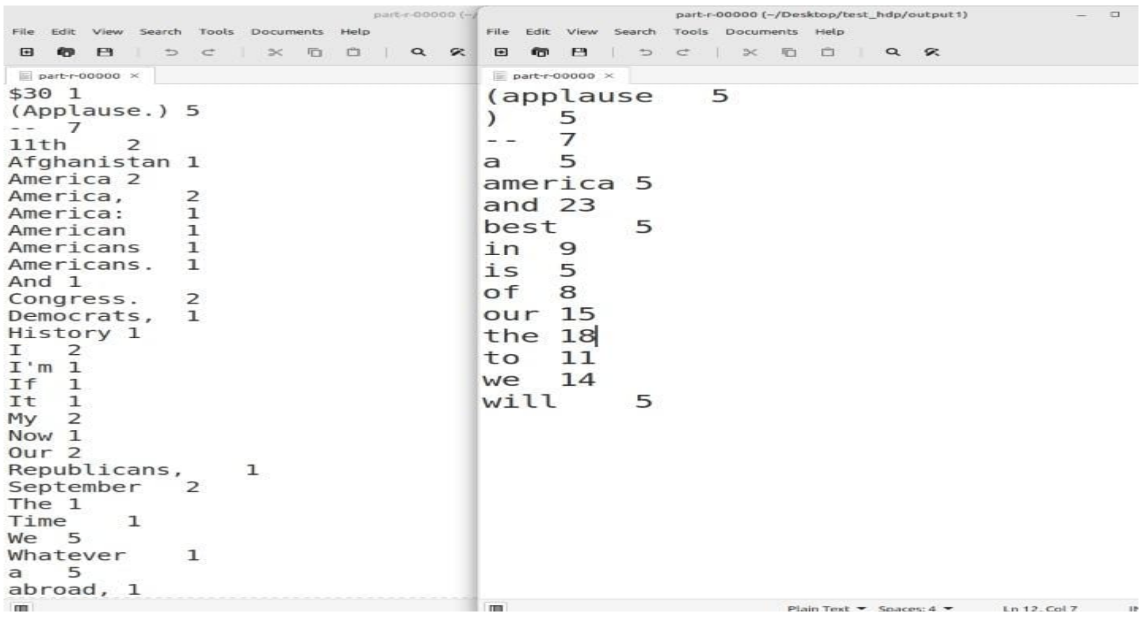
```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;

public class WordCount {

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

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString()," \t\n\r\f...
   ,.:;?![]'");
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken().toLowerCase());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      if(sum>4) context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

## Output


part-r-00000 (~/Desktop/test_hdp/output1)
File Edit View Search Tools Documents Help
part-r-00000
Plain Text
Spaces: 4
Ln 1, Col 2
INS
11:10

```
(applause	5
)	5
--	7
a	5
america	5
and	23
best	5
in	9
is	5
of	8
our	15
the	18
to	11
we	14
will	5
```

Figure 1: Output file


part-r-00000 (~/Desktop/test_hdp/output1)
File Edit View Search Tools Documents Help
part-r-00000
part-r-00000
Plain Text
Spaces: 4
Ln 12, Col 7

```
$30	1
(Applause.)	5
--	7
11th	2
Afghanistan	1
America	2
America,	2
America:	1
American	1
Americans	1
Americans.	1
And	1
Congress.	2
Democrats,	1
History	1
I	2
I'm	1
If	1
It	1
My	2
Now	1
Our	2
Republicans,	1
September	2
The	1
Time	1
We	5
Whatever	1
a	5
abroad,	1
```

```
(applause	5
)	5
--	7
a	5
america	5
and	23
best	5
in	9
is	5
of	8
our	15
the	18
to	11
we	14
will	5
```

Figure 2: Difference in both output files

## Question 2

Working on patent data set and observe the output
Dataset Download link :-
`https://www.nber.org/research/data/us-patents`

**Answer :**

**Finding Citing Patents**
The data set is of size 264.1 MB
Name : cite75_99.txt

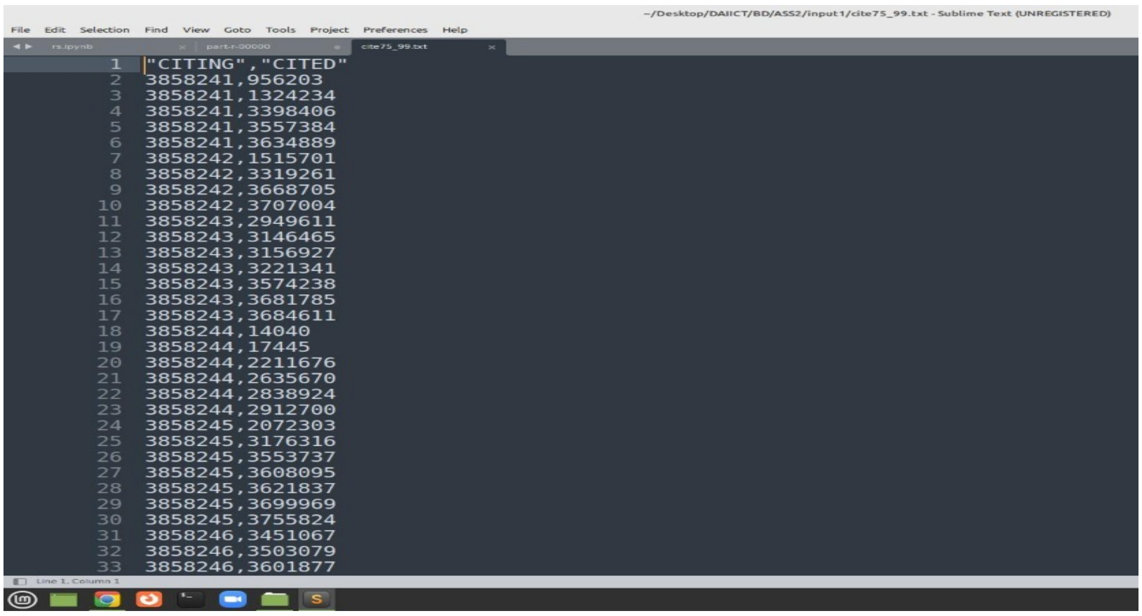
These data comprise detail information on almost 3 million U.S. patents granted between January 1963 and December 1999, all citations made to these patents between 1975 and 1999 (over 16 million), and a reasonably broad match of patents to Compustat (the data set of all firms traded in the U.S. stock market).

**File Name: FindCitingPatents.java**

```java

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
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.Tool;
import org.apache.hadoop.util.ToolRunner;

public class FindCitingPatents extends Configured implements Tool {

    public static enum Counters {
        TOTAL_CITATIONS,
        TOTAL_PATENTS
    }

    // Map inputs: (citing patent, cited patent)
    // Map outputs: (cited patent, citing patent)
    public static class MapClass extends Mapper<LongWritable, Text, Text, Text> ...
   {

        private Text citing = new Text();
        private Text cited = new Text();

        @Override
        protected void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

            String[] split = value.toString().split(",");
            citing.set(split[0]);
            cited.set(split[1]);

            context.write(cited, citing);
            context.getCounter(Counters.TOTAL_CITATIONS).increment(1L);
        }
    }

    // Reduce inputs: (cited patent, list(citing patent))
    // Reduce outputs: (cited patent, CSV of citing patents)
    public static class Reduce extends Reducer<Text, Text, Text, Text> {

        private Text citing = new Text();

        @Override
        protected void reduce(Text key, Iterable<Text> values, Context context)
                throws IOException, InterruptedException {

            StringBuilder builder = new StringBuilder();
            for (Text value : values) {
                if (builder.length() > 0) {
                    builder.append(",");
                }
                builder.append(value.toString());
            }

            citing.set(builder.toString());
            context.write(key, citing);
            context.getCounter(Counters.TOTAL_PATENTS).increment(1L);
        }
    }

    public int run(String[] args) throws Exception {
        Configuration conf = getConf();

        Job job = new Job(conf, FindCitingPatents.class.getSimpleName());
        job.setJarByClass(FindCitingPatents.class);
        job.setMapperClass(MapClass.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int result = ToolRunner.run(new Configuration(), new FindCitingPatents()...
    , args);
        System.exit(result);
    }

}
```

This program will take the patent citation data and invert it. For each patent, we we will find and group the patents that cite it.

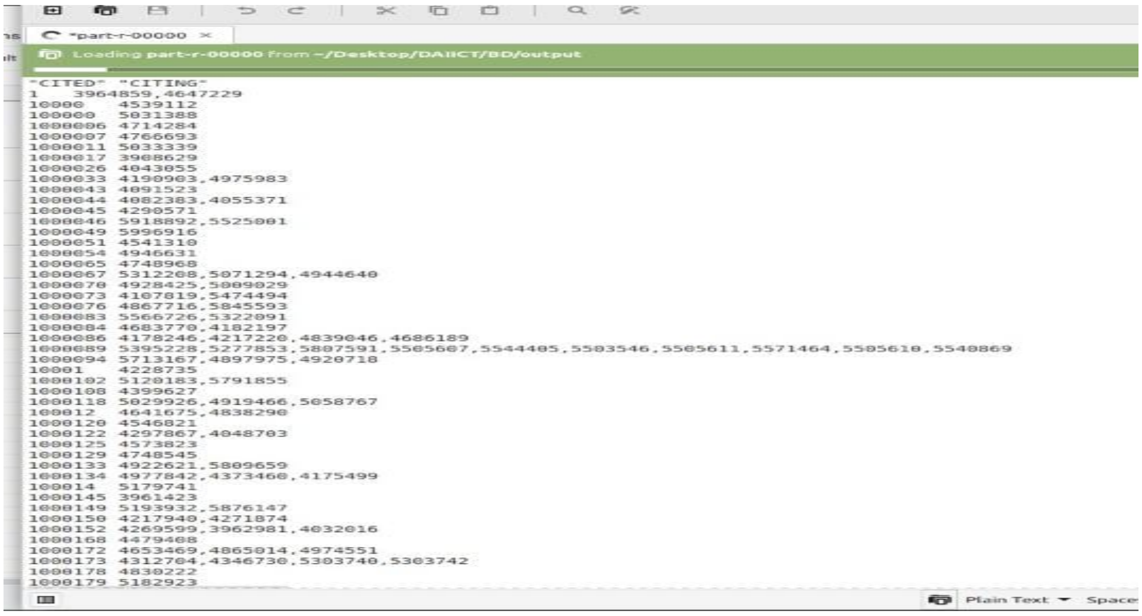
## Input



```
"CITING","CITED"
3858241,956203
3858241,1324234
3858241,3398406
3858241,3557384
3858241,3634889
3858242,1515701
3858242,3319261
3858242,3668705
3858242,3707004
3858243,2949611
3858243,3146465
3858243,3156927
3858243,3221341
3858243,3574238
3858243,3681785
3858243,3684611
3858244,14040
3858244,17445
3858244,2211676
3858244,2635670
3858244,2838924
3858244,2912700
3858245,2072303
3858245,3176316
3858245,3553737
3858245,3608095
3858245,3621837
3858245,3699969
3858245,3755824
3858246,3451067
3858246,3503079
3858246,3601877
```

Figure 3: Input

## Output



```
"CITED" "CITING"
1	3964859,4647229
10000	4539112
100000	5031388
1000006 4714284
1000007 4766693
1000011 5033339
1000017 3908629
1000026 4043055
1000033 4190903,4975983
1000043 4091523
1000044 4082383,4055371
1000045 4290571
1000046 5918892,5525001
1000049 5996916
1000051 4541310
1000054 4946631
1000065 4748968
1000067 5312208,5071294,4944640
1000070 4928425,5009029
1000073 4107819,5474494
1000076 4867716,5845593
1000083 5566726,5322091
1000084 4683770,4182197
1000086 4178246,4217220,4839046,4686189
1000089 5395228,5277853,5807591,5505607,5544405,5503546,5505611,5571464,5505610,5540869
1000094 5713167,4897975,4920718
10001	4228735
1000102 5120183,5791855
1000108 4399627
1000118 5029926,4919466,5058767
100012	4641675,4838290
1000120 4546821
1000122 4297867,4048703
1000125 4573823
1000129 4748545
1000133 4922621,5809659
1000134 4977842,4373460,4175499
100014	5179741
1000145 3961423
1000149 5193932,5876147
1000150 4217940,4271874
1000152 4269599,3962981,4032016
1000168 4479408
1000172 4653469,4865014,4974551
1000173 4312704,4346730,5303740,5303742
1000178 4830222
1000179 5182923
```

Figure 4: citation data

The output is same the given in the Book

## Counting Citations of patent

Take input as used in previous input file

As the patent citation data set only covers patents issued between 1975 and 1999.
To build a new program just need some modification in previous program name "Find-CitingPatents.java".
in this java program modifiction required is in Reducer class.

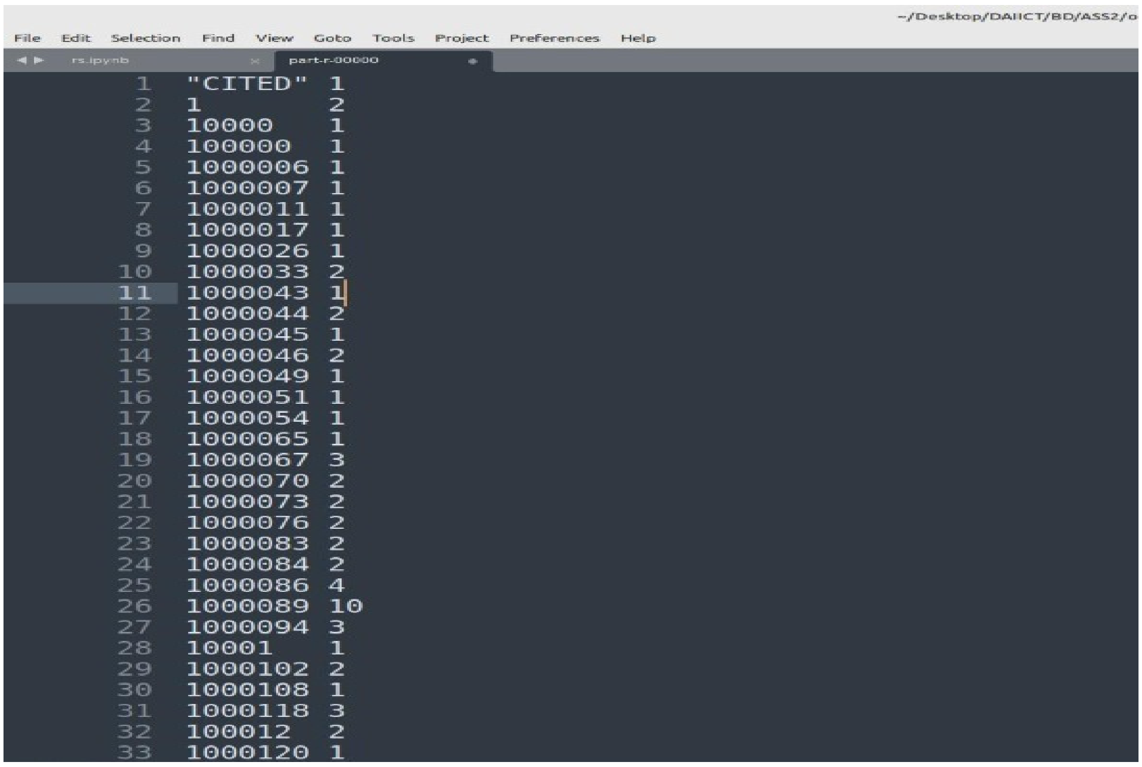
**File Name: FindCitingPatentsCount.java**

```java
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.Tool;
import org.apache.hadoop.util.ToolRunner;

public class FindCitingPatentsCount extends Configured implements Tool {

    public static enum Counters {
        TOTAL_CITATIONS,
        TOTAL_PATENTS
    }

    // Map inputs: (citing patent, cited patent)
    // Map outputs: (cited patent, citing patent)
    public static class MapClass extends Mapper<LongWritable, Text, Text, Text> ...
   {

        private Text citing = new Text();
        private Text cited = new Text();

        @Override
        protected void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

            String[] split = value.toString().split(",");
            citing.set(split[0]);
            cited.set(split[1]);

            context.write(cited, citing);
            context.getCounter(Counters.TOTAL_CITATIONS).increment(1L);
        }
    }

    // Reduce inputs: (cited patent, list(citing patent))
    // Reduce outputs: (cited patent, count of citing patents)
    public static class Reduce extends Reducer<Text, Text, Text, IntWritable> {

        private IntWritable citingCount = new IntWritable();

        @SuppressWarnings({ "UnusedDeclaration" })
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context context)
                throws IOException, InterruptedException {

            int count = 0;
            for (Text value : values) {
                count++;
            }

            citingCount.set(count);
            context.write(key, citingCount);
            context.getCounter(Counters.TOTAL_PATENTS).increment(1L);
        }
    }

    public int run(String[] args) throws Exception {
        Configuration conf = getConf();

        Job job = new Job(conf, FindCitingPatentsCount.class.getSimpleName());
        job.setJarByClass(FindCitingPatentsCount.class);
        job.setMapperClass(MapClass.class);
        job.setReducerClass(Reduce.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int result = ToolRunner.run(new Configuration(), new ...
    FindCitingPatentsCount(), args);
        System.exit(result);
    }

}
```

## Output

In each record, a patent number is associated with the number of citations it has received.

## Output File


~/Desktop/DAIICT/BD/ASS2/o
File Edit Selection Find View Goto Tools Project Preferences Help
rs.ipynb
part-r-00000

```
"CITED" 1
1       2
10000   1
100000  1
1000006 1
1000007 1
1000011 1
1000017 1
1000026 1
1000033 2
1000043 1
1000044 2
1000045 1
1000046 2
1000049 1
1000051 1
1000054 1
1000065 1
1000067 3
1000070 2
1000073 2
1000076 2
1000083 2
1000084 2
1000086 4
1000089 10
1000094 3
10001   1
1000102 2
1000108 1
1000118 3
100012  2
1000120 1
```

Figure 5: count patent cited

## Citation Histogram

In this program we are taking input as output of the previous(patent citation count data) program.

Use the program given below to get the axis to plot histogram.

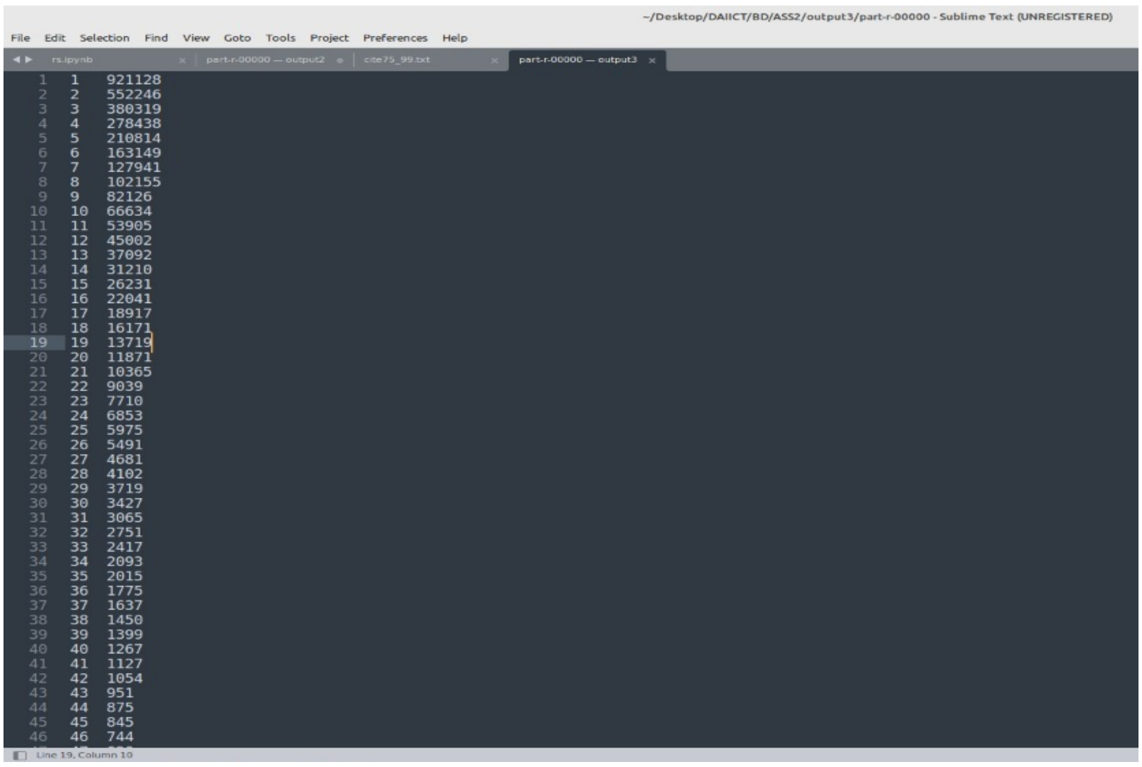
### Fine Name: CitationHistogram.java

```java
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.Tool;
import org.apache.hadoop.util.ToolRunner;

public class CitationHistogram extends Configured implements Tool {

    public static enum Counters {
        TOTAL_CITATIONS,
        TOTAL_PATENTS
    }

    public static class MapClass extends Mapper<LongWritable, Text, IntWritable,...
    IntWritable> {

        private final IntWritable one = new IntWritable(1);
        private IntWritable citationCount = new IntWritable();

        @Override
        protected void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

            String[] split = value.toString().split("\t");
            citationCount.set(Integer.parseInt(split[1]));
            context.write(citationCount, one);
            context.getCounter(Counters.TOTAL_CITATIONS).increment(1L);
        }
    }

    public static class Reduce extends Reducer<IntWritable, IntWritable, ...
   IntWritable, IntWritable> {

        private IntWritable frequency = new IntWritable();

        @Override
        protected void reduce(IntWritable key, Iterable<IntWritable> values, ...
   Context context)
                throws IOException, InterruptedException {

            int count = 0;
            for (IntWritable value : values) {
                count += value.get();
            }
            frequency.set(count);
            context.write(key, frequency);
            context.getCounter(Counters.TOTAL_PATENTS).increment(1L);
        }
    }

    public int run(String[] args) throws Exception {
        Configuration conf = getConf();

        Job job = new Job(conf, CitationHistogram.class.getSimpleName());
        job.setJarByClass(CitationHistogram.class);
        job.setMapperClass(MapClass.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int result = ToolRunner.run(new Configuration(), new CitationHistogram()...
    , args);
        System.exit(result);
    }

}
```

Running the MapReduce job on the citation count data will show the output shown below.
As we suspect,

A large number (900K+) of patents have only one citation, whereas some have hundreds of citations. The most popular patent has 779 citations.

## Output File


~/Desktop/DAIICT/BD/ASS2/output3/part-r-00000 - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
rs.ipynb
part-r-00000 — output2
cite75_99.txt
part-r-00000 — output3
Line 19, Column 10

```
1	921128
2	552246
3	380319
4	278438
5	210814
6	163149
7	127941
8	102155
9	82126
10	66634
11	53905
12	45002
13	37092
14	31210
15	26231
16	22041
17	18917
18	16171
19	13719
20	11871
21	10365
22	9039
23	7710
24	6853
25	5975
26	5491
27	4681
28	4102
29	3719
30	3427
31	3065
32	2751
33	2417
34	2093
35	2015
36	1775
37	1637
38	1450
39	1399
40	1267
41	1127
42	1054
43	951
44	875
45	845
46	744
```

Figure 6: Histogram Output

**Histogram**

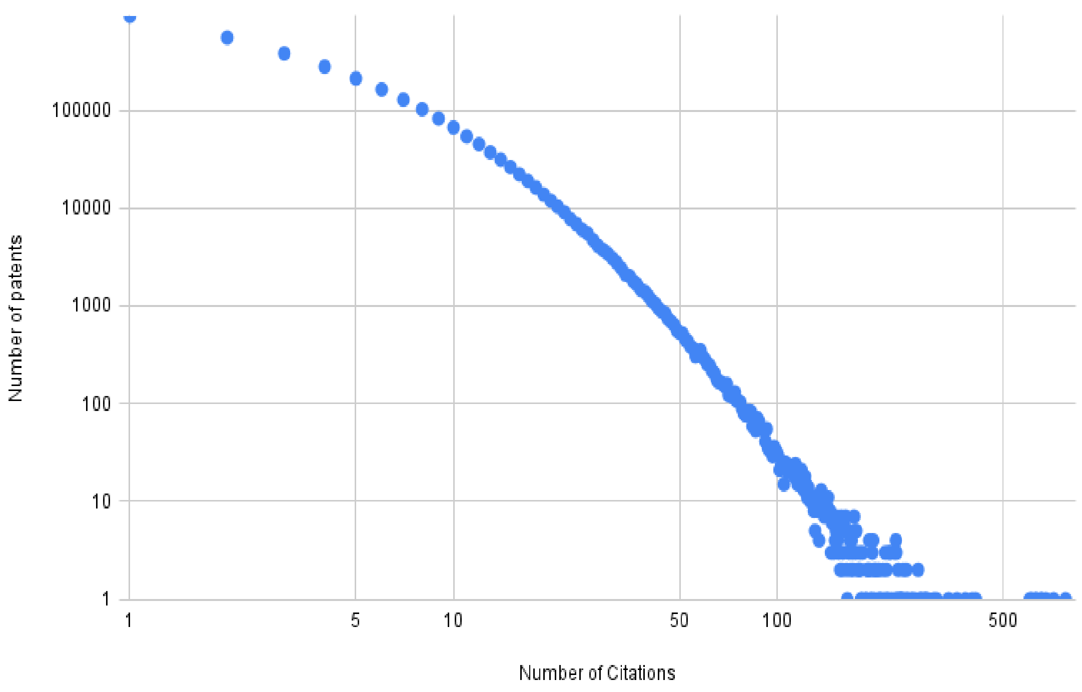


Figure 7: Histogram

Figure shows the number of patents at various citation frequencies.

The plot is on a log-log scale. When a distribution shows as a line in a log-log plot, it's considered to be a power law distribution .
The citation count histogram seems to fit the description, although its approximately parabolic curvature also suggests a log-normal distribution .

## Observation

With the Help of Hadoop, large data Execute very fast.