Big Data Programming – Assignment 3

Varaprasad Kurra Panther ID - #002430487

Task – I

Code changes in the final *HadoopPageRank* Java Project.

We added a final Mapper java class named *HadoopPageRankResultMapper* to the existing Project. The sole purpose of this class is to sort the results of the final output from the last iteration. A few changes are also made to the existing HadoopPageRank.java that contains the configurations of the other Initial, Main and Final Mapper and Reducer classes.

Changes in the HadoopPageRank.java

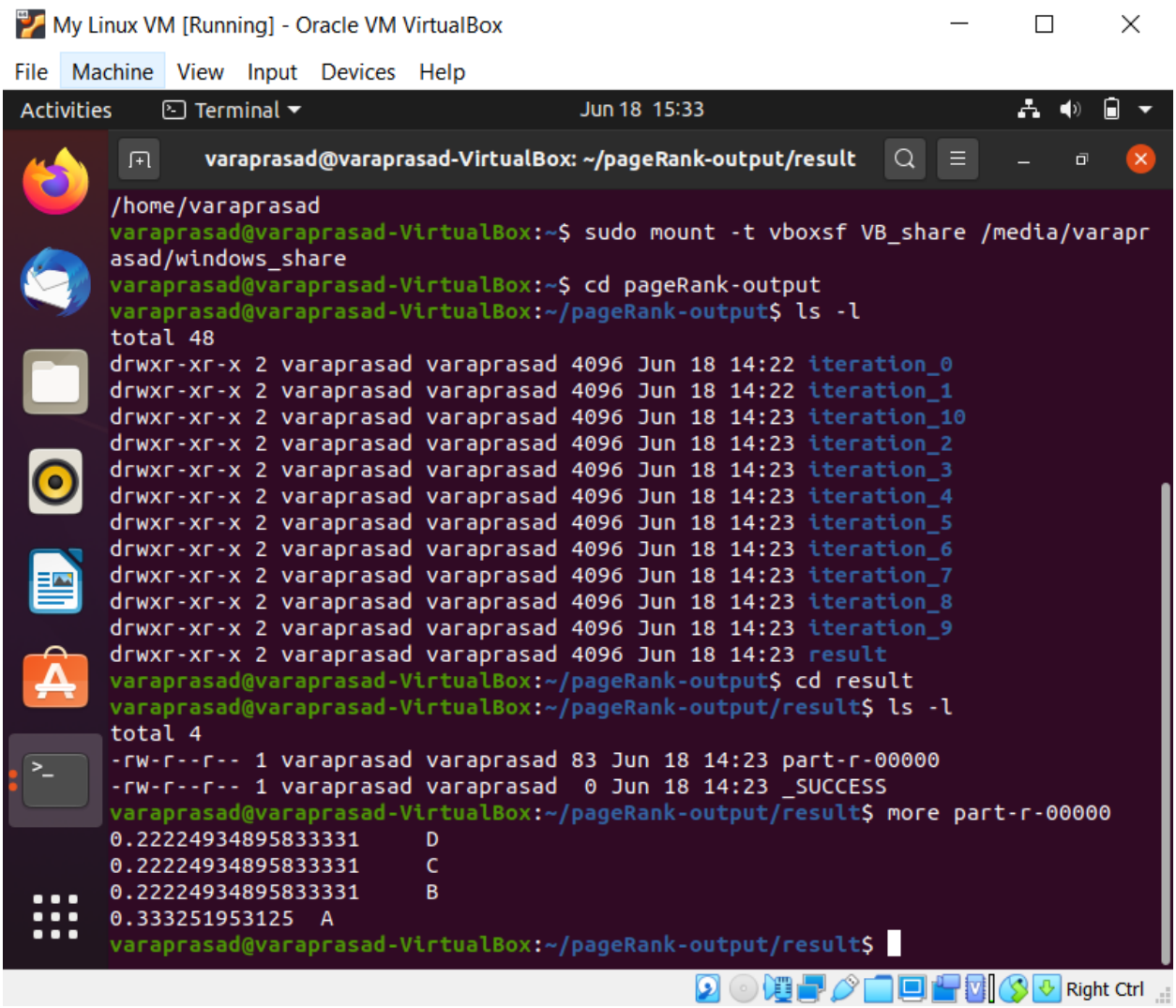
```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class HadoopPageRank extends Configured implements Tool {

public int run(String[] args) throws Exception
{
      int iteration = 0;
      Path inputPath = new Path(args[0]);
      Path basePath = new Path(args[1]);
      FileSystem fs = FileSystem.get(getConf());
      fs.delete(basePath, true);
      fs.mkdirs(basePath);
      Job initJob = Job.getInstance(getConf(),"pageRank");
      initJob.setJarByClass(HadoopPageRank.class);
      initJob.setMapperClass(HadoopPageRankInitMapper.class);
      initJob.setReducerClass(HadoopPageRankInitReducer.class);
      initJob.setOutputKeyClass(Text.class);
      initJob.setOutputValueClass(Text.class);
      initJob.setInputFormatClass(TextInputFormat.class);

      Path outputPath = new Path(basePath, "iteration_" + iteration);
      FileInputFormat.addInputPath(initJob, inputPath);
      FileOutputFormat.setOutputPath(initJob, outputPath);

if ( !initJob.waitForCompletion(true) ) {

      return -1;
}

int totalIterations = Integer.parseInt(args[2]);
```

```
while ( iteration < totalIterations ) {
      iteration = iteration +1 ;
      inputPath = outputPath;
      outputPath = new Path(basePath, "iteration_" + iteration);
      Job mainJob = Job.getInstance(getConf(),"Iteration " + iteration);
      mainJob.setJarByClass(HadoopPageRank.class);
      mainJob.setMapperClass(HadoopPageRankMainJobMapper.class);
      mainJob.setReducerClass(HadoopPageRankMainJobReducer.class);
      mainJob.setOutputKeyClass(Text.class);
      mainJob.setOutputValueClass(Text.class);
       mainJob.setInputFormatClass(TextInputFormat.class);
      FileInputFormat.setInputPaths(mainJob, inputPath);
      FileOutputFormat.setOutputPath(mainJob, outputPath);

if ( !mainJob.waitForCompletion(true) )
{
      return -1;
      }
}

// collect the result, highest rank first - you will need to finish this up
      Job resultJob = Job.getInstance(getConf(),"final result");
      resultJob.setJarByClass(HadoopPageRank.class);
      resultJob.setMapperClass(HadoopPageRankResultMapper.class);
      resultJob.setOutputKeyClass(Text.class);
      resultJob.setOutputKeyClass(Text.class);
      FileInputFormat.setInputPaths(resultJob, outputPath);
      FileOutputFormat.setOutputPath(resultJob,new Path(basePath, "result"));

if (!resultJob.waitForCompletion(true))
{
      return -1;
}
      return 0;
}

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

}
```

HadoopPageRankResultMapper (Sorting Stage)

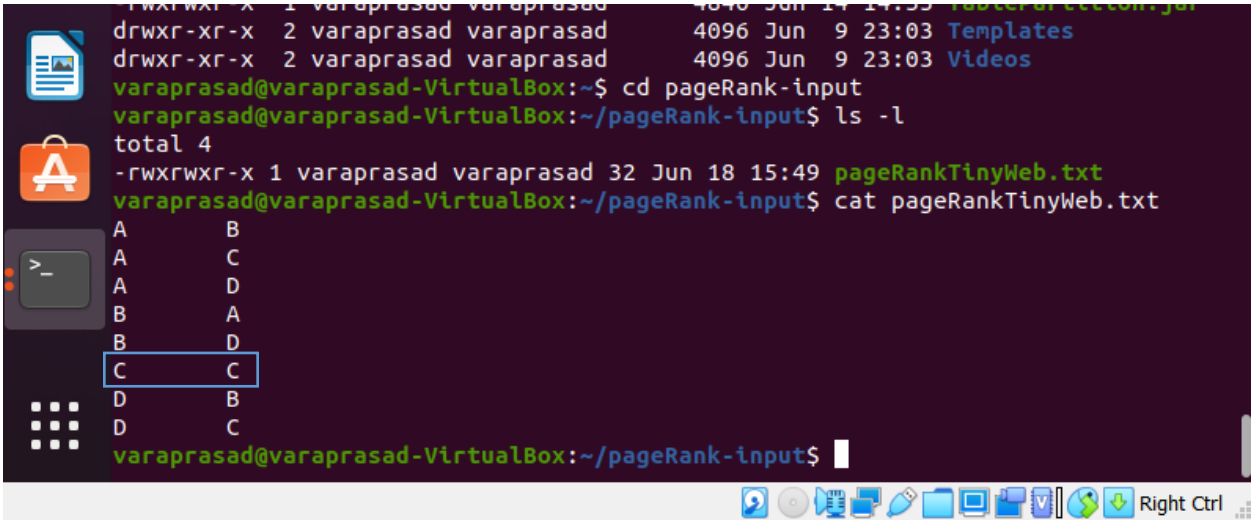
```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

import java.io.IOException;

public  class  HadoopPageRankResultMapper  extends  Mapper<LongWritable,  Text,
Text, Text>
            {
```

```
 public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {


        if ( value == null || value.charAt(0) == '#' ) {
              return;
        }

        int tabIdx1 = value.find("\t");

          String nodeA = Text.decode(value.getBytes(), 0, tabIdx1 );
        String nodeB = Text.decode(value.getBytes(), tabIdx1 + 1,
value.getLength() - (tabIdx1 + 1));
        String [] pageResult = nodeB.split("\t");
        String result = pageResult[0];
        context.write(new Text(result), new Text(nodeA));

      }
}
```



Task 2:

To avoid the spider-trap, we need to alter the calculations in the MainReducer Job. This is the part of the we are actually passing the results to the result phase. For observing the spider trap We need to change the input so that the we inject the Spider-Trap.

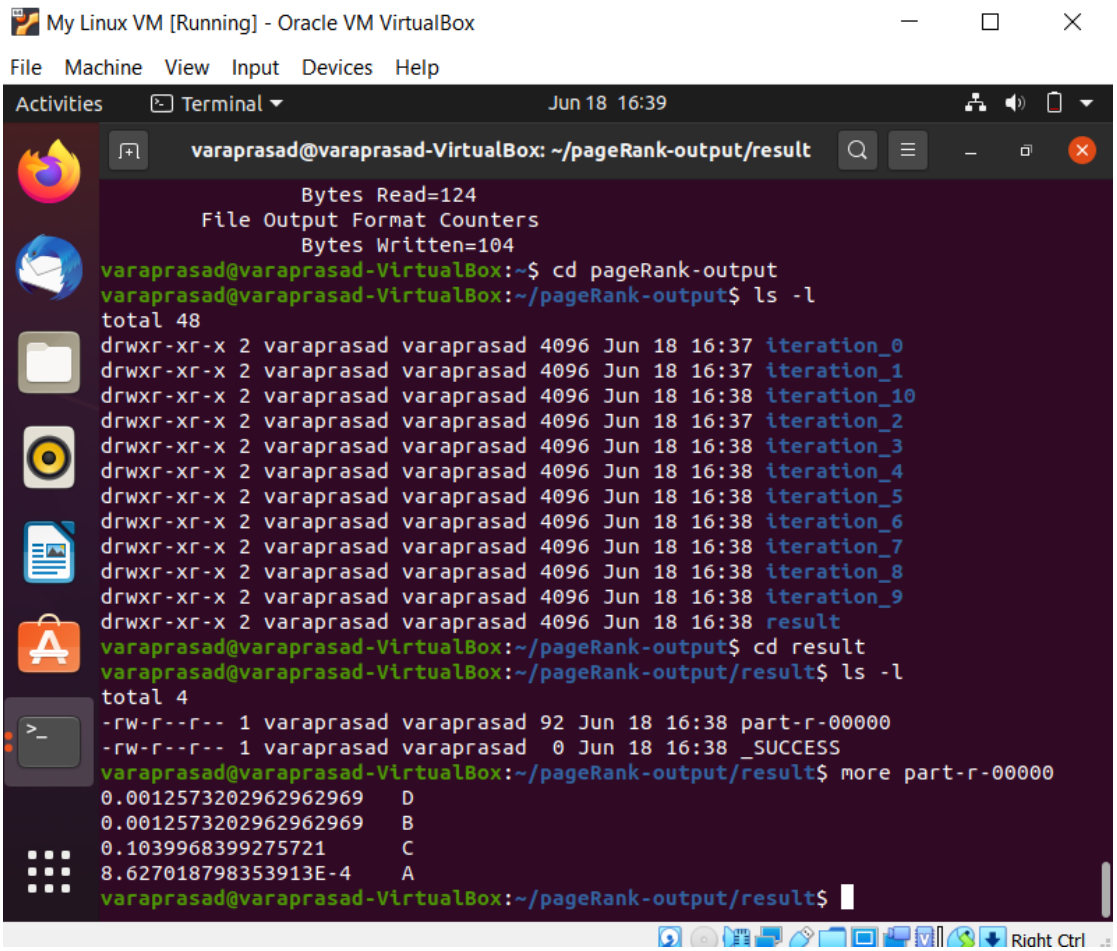


We made the spider-trap by modifying the direction from C – C.

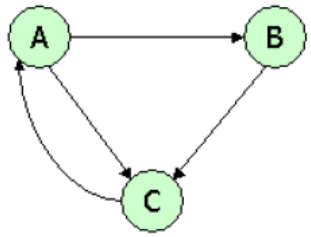
Modifications in the `HadoopPageRankMainJobReducer`.java file

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class HadoopPageRankMainJobReducer extends Reducer<Text, Text, Text,
Text>
{
    public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException
    {
      double beta = 0.8;
      if ( values == null )
      {
            return;
      }

      String links = "";
        double receivedContribution = 0.0;

        for (Text value : values) {

            String content = value.toString();
            if (content.startsWith("|"))
                  {
                        links += content.substring("|".length());
                        }
```

```
            else
                {
                receivedContribution += Double.parseDouble(content);
                 }
             }

        double newPageRank = ((receivedContribution* beta) + (1-beta)*(1/4));
        context.write(key, new Text(newPageRank + "\t" + links));
    }
}
```



Question -2: Given TinyWeb is as below.



Hence from the above Tiny Web can build our Stochastic Column Matrix that describes the probabilities of the next page to be visited from the initial.

Observed Stochastic column Matrix is: 3*3 Matrix

$$\begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix}$$

Given a=b=c=1. So the column matrix i. e the initial PageRank 3*1 Matrix is $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

So, the First iteration gives us the 3*1 Matrix that is: $\begin{pmatrix} 1 \\ 1/2 \\ 3/2 \end{pmatrix}$

After 2nd iteration the resultant Matrix is: $\begin{pmatrix} 3/2 \\ 1/2 \\ 1 \end{pmatrix}$

After 3rd iteration the obtained Matrix Is $\begin{pmatrix} 1 \\ 3/4 \\ 5/4 \end{pmatrix}$

After 4th iteration $\begin{pmatrix} 5/4 \\ 1/2 \\ 5/4 \end{pmatrix}$

After the 5th iteration the resultant column matrix is $\begin{pmatrix} 5/4 \\ 5/8 \\ 5/4 \end{pmatrix}$

Hence the answer would be b = ½. Option C.