Junhao Du

Project Files location:/ Assignment5-Resources.zip/Assignment5-Code
Video Location:/ Assignment5-Resources.zip/ assignment5.mp4
Jar Files Location: /PageRank-1.0.0.jar
Rubric location:/ Assignment5-PageRank-rubric.pdf
Output location:/output.txt
Something Deserved to be mentioned:
1)
In this assignment, we should upload the inputs into Hadoop HDFS but keep the jar file locally.

2)
As showed in the recoding mp4 file, it run with 5 reducers.
After I tried 20 reducers, the speed is much higher than 5 reducers.

3)
I just do the test locally and didn't upload it into Amazon EMR because of the deadline.

I don't have enough time to build and test the environment. But the codes work good locally.
4)
For the codes:

```java
import java.io.IOException;

public class DiffMap1 extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException,
            IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        String[] sections = line.split("\t"); // Splits each line
        if (sections.length > 2) // checks for incorrect data format
        {
            throw new IOException("Incorrect data format");
        }
        /**
         *  TOD: read node-rank pair and emit: key:node, value:rank
         */
        String[] noderank = sections[0].split("\\+");// split node+rank;
        context.write(new Text(noderank[0]),  new Text(noderank[1])); // emit node, rank
    }

}
```

```java
public class DiffMap2 extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedI
            IllegalArgumentException {
        String s = value.toString(); // Converts Line to a String

        /*
         * TOD: emit: key:"Difference" value:difference calculated in DiffRed1
         */
        String[] noderank = s.split("\t+");
        context.write(new Text("Difference"), new Text(noderank[1]));
    }

}
```

```java
olic void reduce(Text key, Iterable<Text> values, Context conte
  double[] ranks = new double[2];
  /*
   * TOD: The list of values should contain two ranks.  Compute
   */
  Iterator<Text> iterator = values.iterator();
  double diff = 0; // default diff has max value
  // caculate rank1
  if(iterator.hasNext()) {
      ranks[0] = Double.valueOf(iterator.next().toString());
  }
  // caculate rank2
  if(iterator.hasNext()) {
      ranks[1] = Double.valueOf(iterator.next().toString());
  }
  // caculate diff
  diff = Math.abs(ranks[0] - ranks[1]);
  System.out.println( key.toString() + "   " + diff);
  context.write(key, new Text(String.valueOf(diff)));
```

```java

public class DiffRed2 extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context)
        double diff_max = 0.0; // sets diff_max to a default value
        /*
         * TOD: Compute and emit the maximum of the differences
         */
        Iterator<Text> iterator = values.iterator();
        // caculate the maxium difference and output the result
        while(iterator.hasNext()) {
            double diff = Double.valueOf(iterator.next().toString());
            diff_max = diff_max > diff ? diff_max : diff;
        }
        context.write(new Text(""), new Text(String.valueOf(diff_max)));
    }
}

```

DiffRed2.java | IterMapper.java | IterReducer.java | PageRankDriver.java | PageRank/pom.xml | FindJoinMapper.java

```java
package edu.stevens.cs549.hadoop.pagerank;

import java.io.File;

public class FindJoinMapper extends Mapper<LongWritable, Text, Text, Text>  {

    @Override
       protected void setup(Context context) throws IOException, InterruptedException {
           if (context.getCacheFiles() != null && context.getCacheFiles().length > 0) {
               URI mappingFileUri = context.getCacheFiles()[0];

               if (mappingFileUri != null) {
                 // Would probably be a good idea to inspect the URI to see what the bit after the # is, as that's t
                   System.out.println("Mapping File: " + FileUtils.readFileToString(new File("./cache")));
               } else {
                   System.out.println(">>>>>> NO MAPPING FILE");
               }
           } else {
               System.out.println(">>>>>> NO CACHE FILES AT ALL");
           }
       }

   public void map(LongWritable key, Text value, Context context)
           throws IOException, InterruptedException, IllegalArgumentException {
       String line = value.toString(); // Converts Line to a String
       /*
        * Join final output with linked name
        * input: key: nodeId+rank, text: adjacent list
        * input: key: nodeId, text: name
        *
        * output: key: nodeId, text: rank
        * ouput: key: nodeId, text:names
        */

       String[] sections;
       if (line.contains(":")) {
           int index = line.indexOf(":");
           sections = new String[2];
           sections[0] = line.substring(0, index);
           sections[1] = line.substring(index + 1, line.length());
       } else {
           sections = line.split("\t"); // Splits it into two parts. Part 1: node+rank | Part 2: adj list
       }


       if (sections.length > 2) // Checks if the data is in the incorrect format
       {
           throw new IOException("Incorrect data format");
       }
       String[] noderank = sections[0].split("\\+");
       if(noderank.length == 1) {
           // it's nodeID with its name
           context.write(new Text(noderank[0]), new Text(PageRankDriver.MARKER_NAME + sections[1].trim()));
       }

       if(noderank.length == 2) {
           // it's nodeID with its rank
           context.write(new Text(noderank[0]), new Text(PageRankDriver.MARKER_RANK + noderank[1]));
       }
   }
```

iffRed2.java | IterMapper.java | IterReducer.java | PageRankDriver.ja... | PageRank/pom.xml

```java
package edu.stevens.cs549.hadoop.pagerank;

import java.io.IOException;

public class FindJoinReducer extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException, IllegalArgumentException {

        /*
         * Join final output with linked name
         * input: key: nodeId, val: (mark_rank)rank
         * input: key: nodeId, val: (mark_name))name
         *
         * output: key: nodeId+names, text: rank
         */
        Iterator<Text> iterator = values.iterator();
        String nodeName = "";
        String rank  = "";
        while(iterator.hasNext()) {
            String tmp = iterator.next().toString();
            if(tmp.startsWith(PageRankDriver.MARKER_NAME)) {
                nodeName = tmp.replaceAll(PageRankDriver.MARKER_NAME, "");
            }
            if(tmp.startsWith(PageRankDriver.MARKER_RANK)) {
                rank = tmp.replaceAll(PageRankDriver.MARKER_RANK, "");
            }
        }

        context.write(new Text(key + "+" + nodeName) , new Text(rank));

    }

}
```

```java
public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException, IllegalArgumentException {
    String line = value.toString(); // Converts Line to a String
    /*
     * TOD output key:-rank, value: node
     * See IterMapper for hints on parsing the output of IterReducer.
     */
    String[] sections = line.split("\t"); // nodeId+nodeName | rank

    if (sections.length > 2) // Checks if the data is in the incorrect format
    {
        throw new IOException("Incorrect data format");
    }
    if (sections.length != 2) {
        return;
    }
    // to reverse shuffle the reducer, we need minus rank with 0
    context.write(new DoubleWritable(0 - Double.valueOf(sections[1])), new Text(sections[0]));
}
```

```java
public class FinReducer extends Reducer<DoubleWritable, Text, Text, Text> {

    public void reduce(DoubleWritable key, Iterable<Text> values, Context context) throws IOException,
            InterruptedException {
        /*
         * TOD: For each value, emit: key:value, value:-rank
         */
        Iterator<Text> iterator = values.iterator();
        String node;
        while(iterator.hasNext()) {
            node = iterator.next().toString();
            // convert -rank back to rank
            context.write(new Text(node), new Text(String.valueOf(0 - key.get())));
        }
    }
}
```

```java
package edu.stevens.cs549.hadoop.pagerank;

import java.io.IOException;

public class InitMapper extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException, Int
            IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        /*
         * TOD: Just echo the input, since it is already in adjacency list format.
         */

        String[] pair = line.split(":");
        if(pair != null && pair.length == 2) {
            context.write(new Text(pair[0].trim()), new Text(pair[1]));
        }
    }

}
```

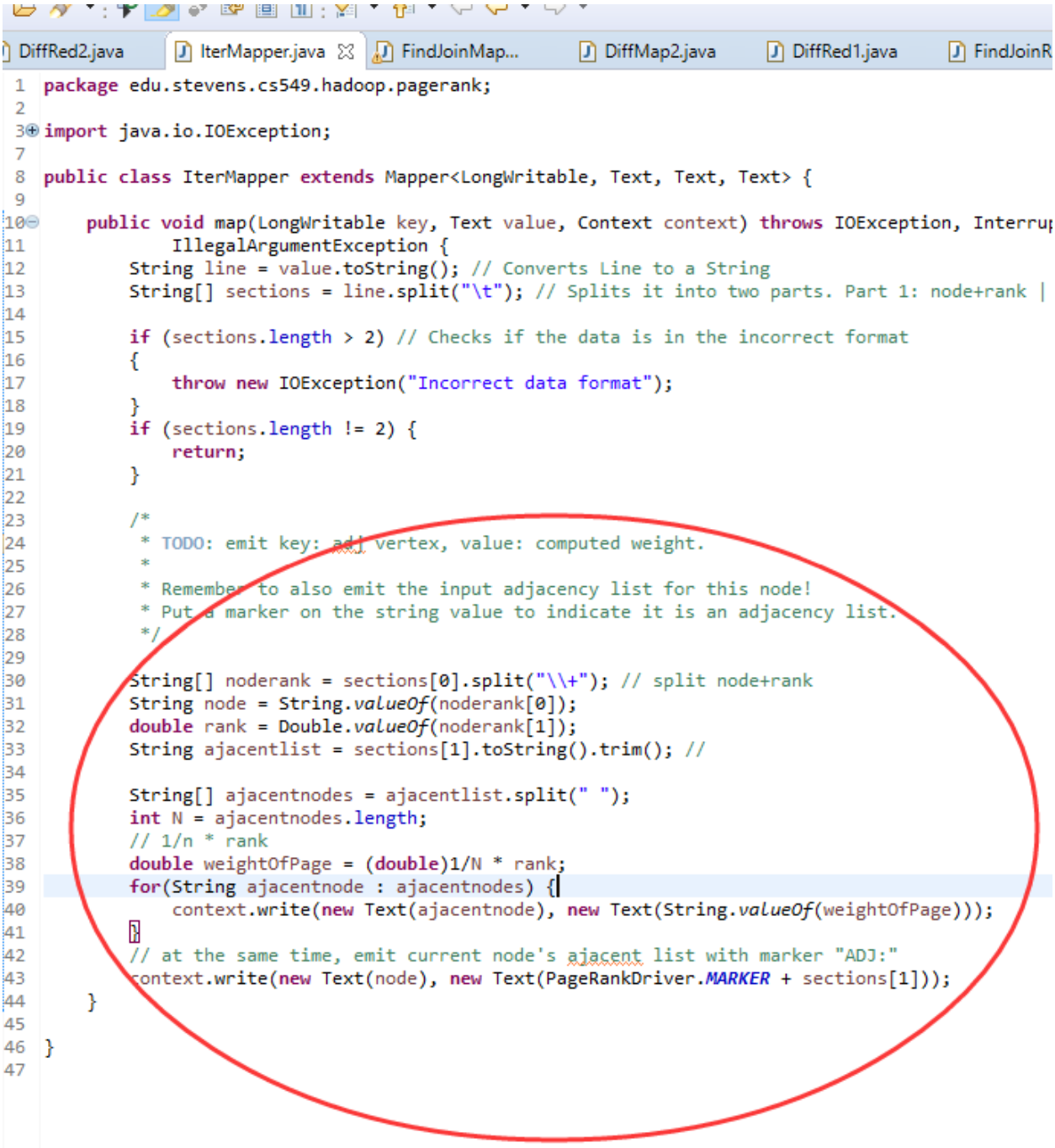
```java
package edu.stevens.cs549.hadoop.pagerank;

import java.io.*;

public class InitReducer extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) thro
        /*
         * TOD: Output key: node+rank, value: adjacency list
         */
        int defualtrank = 1;
        Iterator<Text> v = values.iterator();
        while(v.hasNext()) {
            // emit node+rank, value
            context.write(new Text(key + "+" + defualtrank), v.next());
        }
    }
}

```


DiffRed2.java
IterMapper.java
FindJoinMap...
DiffMap2.java
DiffRed1.java
FindJoinR
1 package edu.stevens.cs549.hadoop.pagerank;
2
3 import java.io.IOException;
7
8 public class IterMapper extends Mapper<LongWritable, Text, Text, Text> {
9
10 public void map(LongWritable key, Text value, Context context) throws IOException, Interru
11 IllegalArgumentException {
12 String line = value.toString(); // Converts Line to a String
13 String[] sections = line.split("\t"); // Splits it into two parts. Part 1: node+rank |
14
15 if (sections.length > 2) // Checks if the data is in the incorrect format
16 {
17 throw new IOException("Incorrect data format");
18 }
19 if (sections.length != 2) {
20 return;
21 }
22
23 /*
24 * TODO: emit key: adj vertex, value: computed weight.
25 *
26 * Remember to also emit the input adjacency list for this node!
27 * Put a marker on the string value to indicate it is an adjacency list.
28 */
29
30 String[] noderank = sections[0].split("\\+"); // split node+rank
31 String node = String.valueOf(noderank[0]);
32 double rank = Double.valueOf(noderank[1]);
33 String ajacentlist = sections[1].toString().trim(); //
34
35 String[] ajacentnodes = ajacentlist.split(" ");
36 int N = ajacentnodes.length;
37 // 1/n * rank
38 double weightOfPage = (double)1/N * rank;
39 for(String ajacentnode : ajacentnodes) {
40 context.write(new Text(ajacentnode), new Text(String.valueOf(weightOfPage)));
41 }
42 // at the same time, emit current node's ajacent list with marker "ADJ:"
43 context.write(new Text(node), new Text(PageRankDriver.MARKER + sections[1]));
44 }
45
46 }
47

```java

import java.io.*;

public class IterReducer extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        double d = PageRankDriver.DECAY; // Decay factor
        /*
         * TOD: emit key:node+rank, value: adjacency list
         * Use PageRank algorithm to compute rank from weights contributed by incoming edges.
         * Remember that one of the values will be marked as the adjacency list for the node.
         */
        Iterator<Text> iterator = values.iterator();
        double currentRank = 0; // default rank is 1 - d
        String ajacentlist = "";
        while(iterator.hasNext()) {
            String line = iterator.next().toString();
            if(!line.startsWith(PageRankDriver.MARKER)) {
                currentRank += Double.valueOf(line);
            } else {
                ajacentlist = line.replaceAll(PageRankDriver.MARKER, "");
            }
        }
        // (1-d) + d * sum(bac
        currentRank = 1 - d + currentRank * d;
        context.write(new Text(key + "+" + currentRank), new Text(ajacentlist));
    }
}



    }
  }
  os.close(); // Closes the writer
  Job job = Job.getInstance();
  job.addCacheFile(new Path(path).toUri());
catch (IOException e) {

  public static void composite(String input, String output, String interim1,
          String interim2, String diff, int reducers) throws Exception {
      /*
       * TOD
       */
      System.out.println("Junhao Du (10431023)");

      int counter = 0;
```