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

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
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.TaskCounter;
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;

import com.google.common.base.Charsets;

public class WordMedian extends Configured implements Tool {

  private double median = 0;
  private final static IntWritable ONE = new IntWritable(1);
  public static class WordMedianMapper extends
      Mapper<Object, Text, IntWritable, IntWritable> {

    private IntWritable length = new IntWritable();
    public void map(Object key, Text value, Context context)
```

```
      throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      String string = itr.nextToken();
      length.set(string.length());
      context.write(length, ONE);
    }
  }
}

public static class WordMedianReducer extends
    Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

  private IntWritable val = new IntWritable();
  public void reduce(IntWritable key, Iterable<IntWritable> values,
      Context context) throws IOException, InterruptedException {

    int sum = 0;
    for (IntWritable value : values) {
      sum += value.get();
    }
    val.set(sum);
    context.write(key, val);
  }
}
private double readAndFindMedian(String path, int medianIndex1,
    int medianIndex2, Configuration conf) throws IOException {
  FileSystem fs = FileSystem.get(conf);
  Path file = new Path(path, "part-r-00000");

  if (!fs.exists(file))
```

```
  throw new IOException("Output not found!");

BufferedReader br = null;

try {
  br = new BufferedReader(new InputStreamReader(fs.open(file), Charsets.UTF_8));
  int num = 0;

  String line;
  while ((line = br.readLine()) != null) {
    StringTokenizer st = new StringTokenizer(line);

    // grab length
    String currLen = st.nextToken();

    // grab count
    String lengthFreq = st.nextToken();

    int prevNum = num;
    num += Integer.parseInt(lengthFreq);

    if (medianIndex2 >= prevNum && medianIndex1 <= num) {
      System.out.println("The median is: " + currLen);
      br.close();
      return Double.parseDouble(currLen);
    } else if (medianIndex2 >= prevNum && medianIndex1 < num) {
      String nextCurrLen = st.nextToken();
      double theMedian = (Integer.parseInt(currLen) + Integer
          .parseInt(nextCurrLen)) / 2.0;
      System.out.println("The median is: " + theMedian);
      br.close();
```

```
        return theMedian;
      }
    }
   } finally {
    if (br != null) {
      br.close();
    }
   }
   // error, no median found
   return -1;
}

public static void main(String[] args) throws Exception {
  ToolRunner.run(new Configuration(), new WordMedian(), args);
}

@Override
public int run(String[] args) throws Exception {
  if (args.length != 2) {
    System.err.println("Usage: wordmedian <in> <out>");
    return 0;
  }

  setConf(new Configuration());
  Configuration conf = getConf();

  Job job = Job.getInstance(conf, "word median");
  job.setJarByClass(WordMedian.class);
  job.setMapperClass(WordMedianMapper.class);
  job.setCombinerClass(WordMedianReducer.class);
  job.setReducerClass(WordMedianReducer.class);
```

```
    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    boolean result = job.waitForCompletion(true);


    long totalWords = job.getCounters()
        .getGroup(TaskCounter.class.getCanonicalName())
        .findCounter("MAP_OUTPUT_RECORDS", "Map output records").getValue();
    int medianIndex1 = (int) Math.ceil((totalWords / 2.0));
    int medianIndex2 = (int) Math.floor((totalWords / 2.0));

    median = readAndFindMedian(args[1], medianIndex1, medianIndex2, conf);

    return (result ? 0 : 1);
  }

  public double getMedian() {
    return median;
  }
}
```