Cloud Computing

Problem Statement

Problem 1

> Develop a Hadoop program that produces the ***n- gram frequencies*** of the text in a given input file. *n-gram* is a contiguous sequence of n characters from a given sequence of text.so the input argument is 1.input text file 2.output directory 3.the value of n

Example Input

- "Helloworld" for the text
- 1 for the n

Algorithm and implementation

> The key point in mapping function is to **extract a continous n-character string starting from each character in the given Text value** and then write it with corresponding frequency in the context.

```
        @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
                String parameter=context.getConfiguration().get("N");
                int n=Integer.valueOf(parameter);
                String s = value.toString();
                for(int i=0;i<s.length()-n+1;i++) {
                    word.set(s.substring(i, i + n));
                    context.write(word, one);
                }
        }
```

> Reduce function is like the normal word count,adding up the corresponding frequency based on the key.

```
        @Override
        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);
            context.write(key, result);
        }
```

Output

- He-1, el-1, ll-1, lo-1, ow-1, wo-1, or-1, rl-1, ld-1

Problem 2

> given an input file consisted with logs.find how many hits were made on website item "/assets/img/home-logo.png"

```
The log file is in Common Log Format:
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/js/lowpro.js
HTTP/1.1" 200 10469

%h %l %u %t \"%r\" %>s %b
```

Algorithm and implementation

> key point in mapping function is to **check whether each text value contains the target asset and it is implemented with the String API**. Reduce function is the same as last one.

```
       @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String s = value.toString();
            String match="/assets/img/home-logo.png";
            if(s.contains(match)) {
                word.set(match);
                context.write(word, one);
            }

        }
```

Output

```
/assets/img/home-logo.png 98776
```

Problem 3

> With the same input log file find how many hits were made from the IP: 10.153.239.5.

Algorithm and implementation

> key point in mapping function is to **check whether each text value contains the target IP address and it is implemented with the String API**. Reduce function is the same as last one.

```
        @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String s = value.toString();
            String match="10.153.239.5";
            if(s.contains(match)) {
                word.set(match);
                context.write(word, one);
            }

        }
```

Output

```
10.153.239.5  547
```

Problem 4

> With the same input log file.Which path in the website has been hit most? How many hits were made to the path?

Algorithm and implementation

> key point in this algorithm is to map as previous problems but while reducing,***only keep the maximum hit***.

```
        @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String s = value.toString();
            int index_get = s.indexOf("\"");
            int index_start = s.indexOf(" ",index_get)+1;
            int index_end = s.indexOf(" ", index_start);
            String url = s.substring(index_start,index_end);
            word.set(url);
            context.write(word, one);
        }


        @Override
        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            if (sum > max) {
                max = sum;
                maxWord.set(key);
            }
        }

        @Override
        public void cleanup(Context context) throws IOException,
  InterruptedException {
            context.write(maxWord, new IntWritable(max));
        }
```

Output

```
/images/filmpics/0000/3695/Pelican_Blood_2D_Pack.jpg  86178
```

Problem 5

> Which IP accesses the website most? How many accesses were made by it?

Algorithm and implementation

> key point in this algorithm is to map as previous problems but while reducing,***only keep the maximum hit***.

```
        @Override
         public void map(Object key, Text value, Context context
         ) throws IOException, InterruptedException {
             String s = value.toString();
             int index_end = s.indexOf(" ");
             String ip = s.substring(0,index_end);
             word.set(ip);
             context.write(word, one);
         }

        @Override
         public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
         ) {
             int sum = 0;
             for (IntWritable val : values) {
                 sum += val.get();
             }
             if (sum > max) {
                 max = sum;
                 maxWord.set(key);
             }
         }

         @Override
         public void cleanup(Context context) throws IOException,
InterruptedException {
             context.write(maxWord, new IntWritable(max));
         }
```

Output

```
10.216.113.172 158614
```