EXPERIMENT 6

1. Aim: To implement a **Bloom Filter** using **Hadoop MapReduce**.

2. Theory:

2.1 What is a Bloom Filter?

A Bloom Filter is a probabilistic data structure that efficiently checks whether an element is possibly in a set or definitely not in the set.

- Space-efficient: Uses a fixed-size bit array.
- Fast: Constant-time insert and lookup.
- False positives possible, but no false negatives.

2.2 How Bloom Filter Works:

- A bit array of size m is initialized to all 0s.
- Multiple hash functions determine the bit positions for each element.
- To insert an element:
  - Hash the element with all hash functions and set the corresponding bits to 1.
- To check membership:
  - Hash the element and check if all corresponding bits are set to 1.
  - If yes → Element is probably present.
  - If no → Element is definitely not present.

2.3 Why Use Bloom Filters?

- Fast and memory-efficient for large datasets.
- Used in big data processing, caching, and network security.

3. Hadoop Setup:

Make sure Hadoop is running:

```
start-dfs.sh

start-yarn.sh

jps
```

4. Java Code:

Filename: BloomFilterMapReduce.java

```
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;

import java.io.IOException;

import java.util.BitSet;

import java.util.StringTokenizer;


public class BloomFilterMapReduce {


    private static final int BITSET_SIZE = 1000;  // Size of the bit array

    private static final BitSet bloomFilter = new BitSet(BITSET_SIZE);


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

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

        private Text word = new Text();’


        @Override

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

            StringTokenizer itr = new StringTokenizer(value.toString());

            while (itr.hasMoreTokens()) {
```

```
            String token = itr.nextToken();
            word.set(token);
            int hash = token.hashCode() % BITSET_SIZE;
            bloomFilter.set(Math.abs(hash));  // Set bit in the Bloom filter
            context.write(word, one);
        }
    }
}

public static class BloomReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Bloom Filter Example");

    job.setJarByClass(BloomFilterMapReduce.class);
```

```
        job.setMapperClass(BloomMapper.class);

        job.setCombinerClass(BloomReducer.class);

        job.setReducerClass(BloomReducer.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);

    }

}
```

5. Compile Java Code:

hadoop com.sun.tools.javac.Main BloomFilterMapReduce.java

Create JAR file:

jar cf bloomfilter.jar BloomFilterMapReduce*.class

6. Input File:

Create input text file:

echo -e "apple banana mango\napple mango banana\nbanana mango apple\norange banana apple\ngrapes apple mango" > input.txt

Upload to HDFS:

sh
CopyEdit
hdfs dfs -mkdir -p /bloom/input
hdfs dfs -put input.txt /bloom/input
7. Run MapReduce Job:

hadoop jar bloomfilter.jar BloomFilterMapReduce /bloom/input /bloom/output

Output:

hdfs dfs -cat /bloom/output/part-r-00000

```
chotu@LAPTOP-KAGQGQP5:~$ hdfs dfs -cat /bloomfilter/output/part-r-00000
Final Bloom Filter      {1170, 1834, 1862, 1877, 2117, 2267, 2703, 3527, 3550, 3
1, 6119, 6365, 7018, 7563, 7618, 7771, 7776, 7878, 7925, 9418, 9512, 9830, 9837}
chotu@LAPTOP-KAGQGQP5:~$
```