**TopThreeMax:**

**MapperCode:**

package com.topn;

import java.io.\*;

import java.util.\*;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class TopThreeMapper

extends Mapper<Object, Text, Text, LongWritable> {

private TreeMap<Long, String> tmap;

@Override

public void setup(Context context)

throws IOException, InterruptedException

{

tmap = new TreeMap<Long, String>();

}

@Override

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException

{

String[] tokens = value.toString().split("\t");

String movie\_name = tokens[0];

long no\_of\_views = Long.parseLong(tokens[1]);

tmap.put(no\_of\_views, movie\_name);

if (tmap.size() > 10) {

tmap.remove(tmap.firstKey());

}

}

@Override

public void cleanup(Context context)

throws IOException, InterruptedException

{

for (Map.Entry<Long, String> entry :

tmap.entrySet()) {

long count = entry.getKey();

String name = entry.getValue();

context.write(new Text(name),

new LongWritable(count));

}

}

}

**Reducercode:**

package com.topn;

import java.io.IOException;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class TopThreeReducer

extends Reducer<Text, LongWritable, LongWritable,

Text> {

private TreeMap<Long, String> tmap2;

@Override

public void setup(Context context)

throws IOException, InterruptedException

{

tmap2 = new TreeMap<Long, String>();

}

@Override

public void reduce(Text key,

Iterable<LongWritable> values,

Context context)

throws IOException, InterruptedException

{

String name = key.toString();

long count = 0;

for (LongWritable val : values) {

count = val.get();

}

tmap2.put(count, name);

if (tmap2.size() > 10) {

tmap2.remove(tmap2.firstKey());

}

}

@Override

public void cleanup(Context context)

throws IOException, InterruptedException

{

for (Map.Entry<Long, String> entry :

tmap2.entrySet()) {

long count = entry.getKey();

String name = entry.getValue();

context.write(new LongWritable(count),

new Text(name));

}

}

}

**DriverCode:**

package com.topn;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

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.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class TopThreeDriver {

public static void main(String[] args) throws Exception

{

Configuration conf = new Configuration();

String[] otherArgs

= new GenericOptionsParser(conf, args)

.getRemainingArgs();

// if less than two paths

// provided will show error

if (otherArgs.length & lt: 2) {

System.err.println(" Error: please provide two paths& quot");

System.exit(2);

}

Job job

= Job.getInstance(conf, " top 10 & quot");

job.setJarByClass(TopThreeDriver.class);

job.setMapperClass(TopThreeMapper.class);

job.setReducerClass(TopThreeReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(

job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(

job, new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}