1. Bonferroni Principle

(a) The number of days of observation was raised to 2000.

Sol:

Given in section 1.2.3 (MMDS book)

The probability of any two people both deciding to visit a hotel on any given day is .0001

The chance that they will visit the same hotel on one given day is 10^{-9}

The chance that they will visit the same hotel on two different given days is the square of this number, 10^{-18}.

The two people were at the same hotel on each of the two days is given by (n 2) where n is number of people 10^9 which is equal to $499999999.5 * 10^9$

The number of pairs of days is (2000 2) = 1999000

The expected number of events that look like evil-doing is given by

$(499999999.5 * 10^9) * (1999000) * 10^{-18} = 999499.999001$

(b) *The number of people observed was raised to 2 billion and there were therefore 200,000 hotels.*

The number of people observed was raised to 2 billion and there were therefore 200,000 hotels

The probability of any two people both deciding to visit a hotel on any given day is .0001

The chance that they will visit the same hotel on one given day is $.0001/200{,}000 = 0.5 * 10^{-9}$

The chance that they will visit the same hotel on two different given days is the square of this number $0.25*10^{-18}$.

The two people were at the same hotel on each of the two days is given by (n 2) where n is number of people 2 billion ($2*10^9$) which is equal to $1999999999* 10^9$

The number of pairs of days is (1000 2) = 499500

The expected number of events that look like evil-doing is given by

$(1999999999* 10^9) * (499500) * (0.25*10^{-18}) = 249749.999875$

2. The Base of Natural Logarithms / Streaming Statistics

(a) $(1.01)^{500} = 1.447e+2$

$(1.05)^{1000} = 1.5463189e+21$

$(0.9)^{40} = 1.478e-2$

(b) 0th frequency moment (F_0) is the number of distinct symbols occurring in the stream,

$F_0 = 10$

1st frequency moment (F_1) n, the length of the string

$F_1 = 1+2+\ldots+10 = 10 * 11/2 = 55$

2nd frequency moment (F_2) or surprise number is square of the number of occurrences of each element in sequence

$$\sum_{i=1}^{m} n^k{}_i = f_k$$

$$F_2 = 1^2 + 2^2 + \ldots. + 10^2 = \sqrt{\frac{1}{6}10(10+1)(2(10)+1)} = 19.621$$

3. The Distribution of Distances in a High-Dimensional Space

if X,Y are independent uniform (0,1) random variables, then the triple $(A,B,C):=(\min(X,Y), \max(X,Y)-\min(X,Y), 1-\max(X,Y))$ is an exchangeable sequence.

In particular,

$\mathbb{E}(A)=\mathbb{E}(B)=\mathbb{E}(C)$,

E(A) = E(B) =E(C) ,and since $A+B+C=1$ identically we must have $\mathbb{E}(B)=\mathbb{E}(\text{distance})= 1/3$, hence proved.

4.

(a)

Algorithm	Cluster Shapes	Input Parameters	Limitations
K-means	round shaped	Dataset/Examples, desired number of clusters	1. Does hard assignments as either a point belongs to cluster or not. 2. Sensitive to outlier examples, outliers can affect the mean
DBSCAN	arbitrarily shaped clusters	Dataset, ε(ε is the maximum radius of the neighborhood from p), minpts(the minimum number of points required to form a dense region)	1. DBSCAN is not entirely deterministic(border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed) 2. The quality of DBScan depends on distance measure, most common algorithm Euclidian distance fails in case of high dimensions 3. DBscan cannot cluster datasets well with large differences in densities, since the minpts-ε combination cannot be chosen appropriately for all clusters

Algorithm	Cluster Shapes	Input Parameters	Limitations
BIRCH	BIRCH can incrementally and dynamically cluster incoming, multi-dimensional metric data point in an attempt to produce the best quality clustering for a given set of resources and handles noise data(outliers) well	Dataset, desired number of clusters k	1. Favours only clusters with spherical shape and similar sizes, because it uses the notion of diameter to control the boundary of a cluster. 2. Handles only numeric data, and sensitive to the order of the data record.
BFR	Clusters which are normally distributed in each dimension around a centroid in euclidean space and axes are fixed(For instance, in two dimensions a cluster may be cigar-shaped, but the cigar must not be rotated off of the axes)	Dataset, desired number of clusters k	1. Assumes clusters are normally distributed and axes are fixed – ellipses at an angle are not OK.
CURE	allows clusters to assume any shape (works also well on clusters having non-spherical shapes and size variances.)	number of representative points c, k clusters, Dataset	1. For smaller values of C, quality of clustering suffers. 2. CURE ignores the information about the aggregate interconnectivity of objects in two clusters

(b) Centroid of (4,8), (9,5), (2,2) is (5, 5)
$SSE1 = (4-5)^2 + (5-8)^2 = 10$
$SSE2 = (9-5)^2 + (5-5)^2 = 16$
$SSE3 = (2-5)^2 + (2-5)^2 = 18$
$SSE = SSE1 + SSE2 + SSE3 = 10 + 16 + 18 = 44$

(c) Answer is (a) 27
If we have 1 single cluster, centroid is (3,4) which gives SSE 44

Grouping points (3,0) and (6,5) as a cluster gives centroid (4.5,2.5) which gives SSE 17.
(0,7) as a single point in cluster has SSE 0.
Hence reduction is by 27

The other groupings as mentioned below results in higher SSE values and can be discarded.
Grouping(3,0) and (0,7) as one cluster and (6,5) as another gives SSE 29
Grouping (0,7) and (6,5) as one cluster and (3,0) as another gives SSE 20.

(d)
Soln: (d) b is added second is the right answer
Given x = (0,0); y = (10,10), a = (1,6); b = (3,7); c = (4,3); d = (7,7), e = (8,2); f = (9,5)

Calculating the euclidean norm between the points a,b,c,d,e,f and x,y

ax = $\sqrt{1^2 + 6^2} = \sqrt{37}$ ay = $\sqrt{9^2 + 4^2} = \sqrt{97}$
bx = $\sqrt{3^2 + 7^2} = \sqrt{58}$ by = $\sqrt{3^2 + 7^2} = \sqrt{58}$
cx = $\sqrt{4^2 + 3^2} = \sqrt{25}$ cy = $\sqrt{6^2 + 7^2} = \sqrt{85}$
dx = $\sqrt{7^2 + 7^2} = \sqrt{98}$ dy = $\sqrt{3^2 + 3^2} = \sqrt{18}$
ex = $\sqrt{8^2 + 2^2} = \sqrt{68}$ ey = $\sqrt{8^2 + 2^2} = \sqrt{68}$
fx = $\sqrt{9^2 + 5^2} = \sqrt{106}$ fy = $\sqrt{1^2 + 5^2} = \sqrt{26}$

point e is at maximum distance from x and y, hence **e is added first**.

Calculating the euclidean norm between the points a,b,c,d,f and e(we can reuse the distance between a, b, c, d, f and x,y)

ae = $\sqrt{65}$
be = $\sqrt{50}$
ce = $\sqrt{17}$
de = $\sqrt{26}$
fe = $\sqrt{10}$

point b is at maximum distance from 3 representative points x,y,e , hence b is added second

Calculating the euclidean norm between the points a,c,d,f and e(we can reuse the distance between a, c, d, f and e, x,y)

ab = $\sqrt{5}$
cb = $\sqrt{17}$
db = $\sqrt{16}$
fb = $\sqrt{38}$

point c is at maximum distance from 3 representative points x,y, e, b, hence c is added third

similarly d is added fourth and f is added 5th.

6.

(a) WordCount - Hello World in MapReduce

Classes used:

FilteringWordCountMapper as Mapper class:

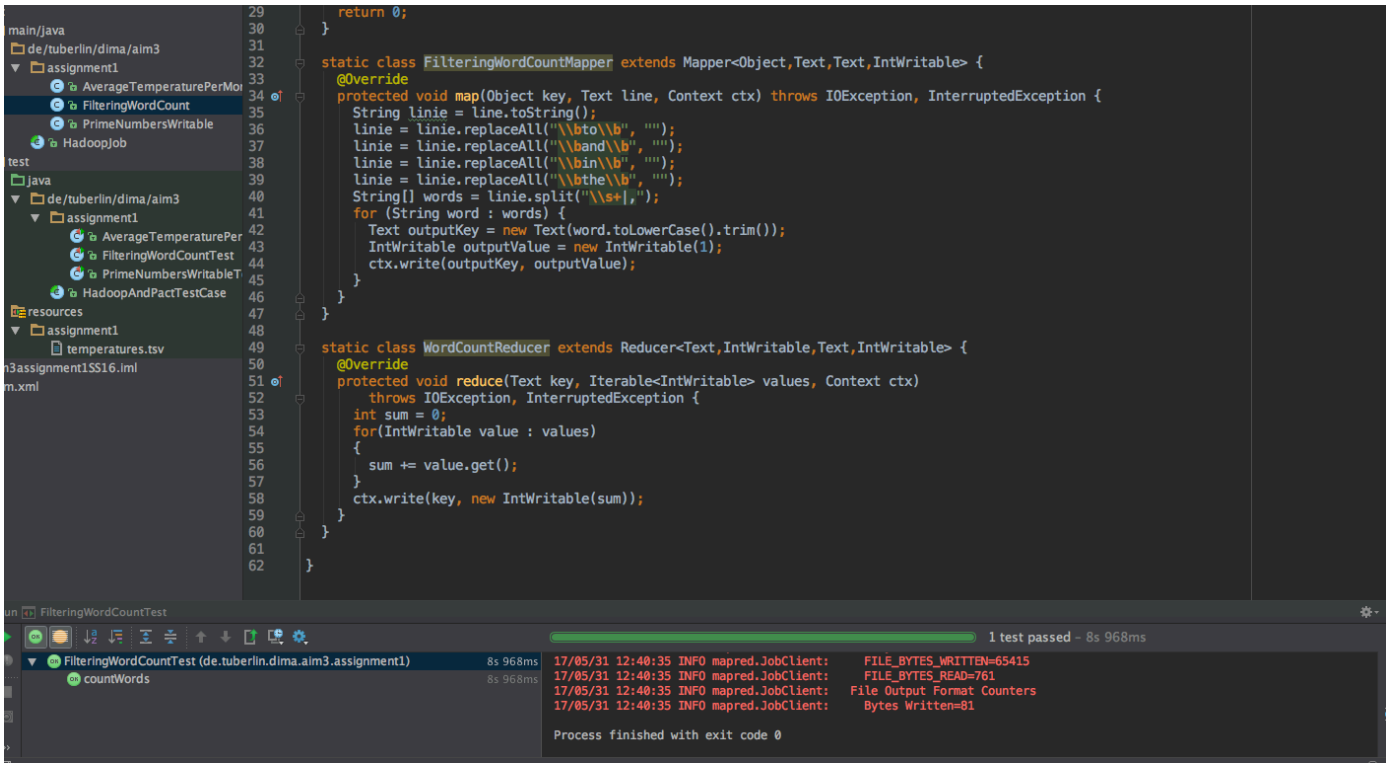
This class filters the words like to, and, in, the and emits (word,1) for each occurrence of a word.

WordCountReducer class is used as a reducer class:

This class combines the result of mapper output and outputs(word, count) where count indicates the total number of occurrence of each word.

Result of unit test can be seen in attached Screen shot 1: FilteringWordCount

Screen shot 1: FilteringWordCount



(b) A Custom Writable for Prime Numbers

The write and readFields methods of Writable class are implemented.

In write(DataOutput out) method:

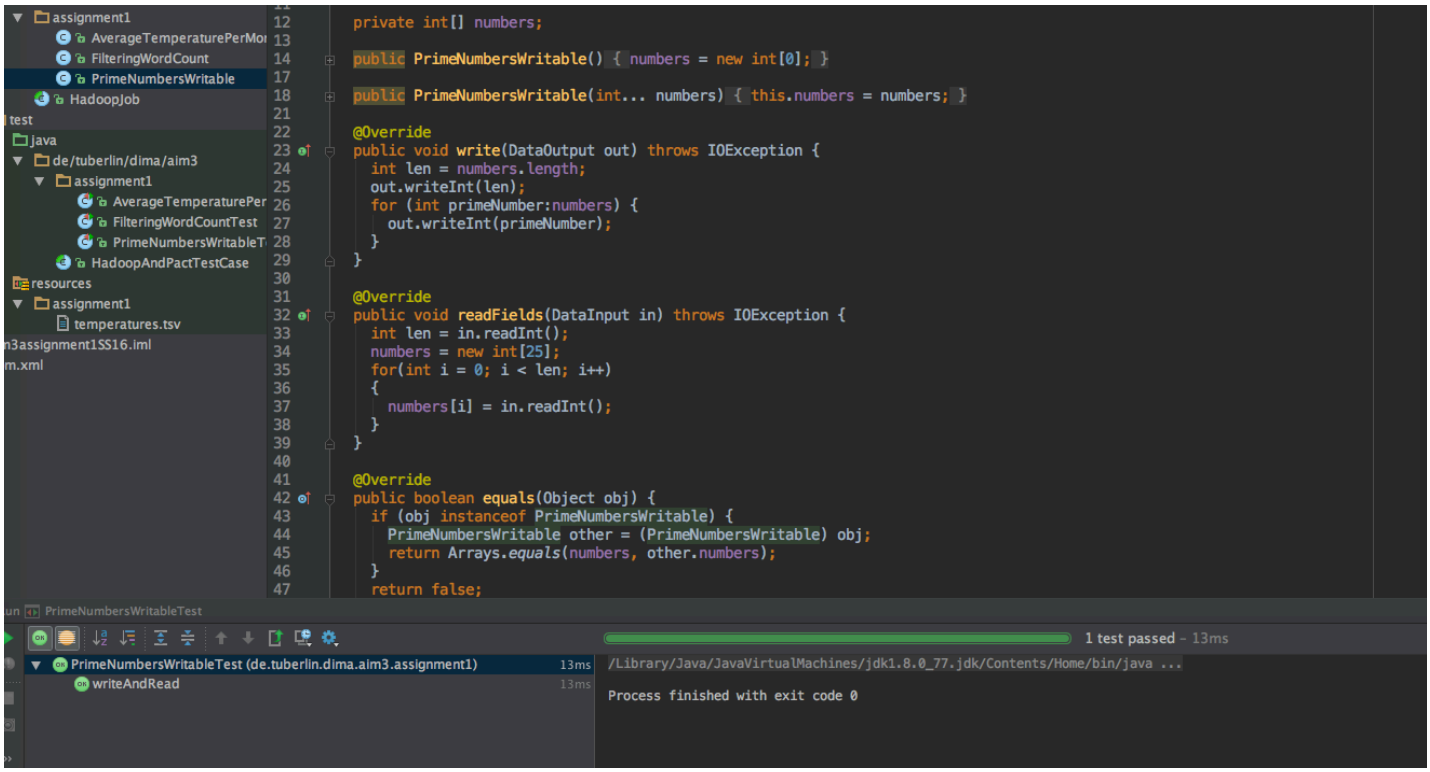
The count of number of elements in the array are calculated and count is written first followed by the elements in the array for serialisation purpose

In readFields(DataInput in) method:

The count is read first from the input stream and elements of the array are read based on the count value for deserialisation purpose

Result of unit test can be seen in attached Screen shot 2: Prime Number Writable

Screen Shot 2: Prime Number Writable



(c) Postprocessing of Temperature Sensor Readings

In this task, as attached screen shot titled " Map reduce implementation"

AverageTemperaturePerMonthMapper implemented ad Mapper class:

In this class, map method filters the data which has sensor quality lesser than threshold quality and emits year and month as key and temperature as the value.
emit(yearAndMonth, temperature)

AverageTemperaturePerMonthReducer is implemented as Reducer class:
In this class, the temperature of each year and month is combined and average is calculated.
The output is emit(yearAndMonth, avgTemperature).

Additionally in this task, since the data needs to be plot, I am reading the data from output file of reducer task and updating it in a csv file(titled "updating output to csv") and the screen shot of the plot is attached(titled as "tableau plot averageTemperaturePerMonth").

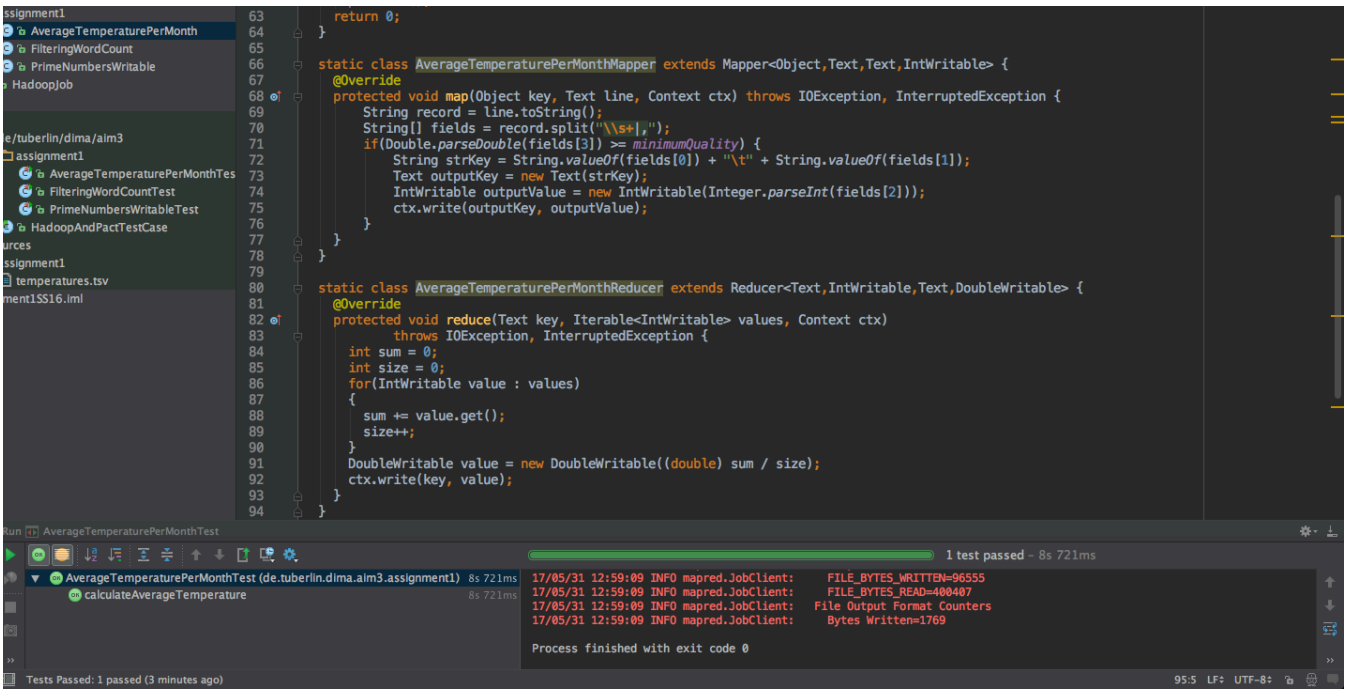
The plot indicates that averageTemperature has reached its maximum in February 1992 and minimum in December 1993.

The result of unit test can be seen in Screen shot 3: Map Reduce Implementation

Screen shot 3: " Map reduce implementation"

```
Path inputPath = new Path(parsedArgs.get("--input"));
Path outputPath = new Path(parsedArgs.get("--output"));

  if (averageTemperatureOutput == null) {
      averageTemperatureOutput = new File("/Users/seema/Documents/Semester2/AIM3/Assignment1/output/averageTemperatureOutput.csv"
      if(averageTemperatureOutput.exists()) averageTemperatureOutput.delete();
      if (!averageTemperatureOutput.getParentFile().exists()) {
          if(!averageTemperatureOutput.getParentFile().mkdir()) {
              throw new IOException("Could not create " + averageTemperatureOutput);
          }
      }
      averageTemperatureOutput.createNewFile();
      pw = new PrintWriter(averageTemperatureOutput);
  }

double minimumQuality = Double.parseDouble(parsedArgs.get("--minimumQuality"));
this.minimumQuality = minimumQuality;
Job avgTempPerMonth = prepareJob(inputPath, outputPath, TextInputFormat.class, AverageTemperaturePerMonth.AverageTemperaturePerMo
        Text.class, IntWritable.class, AverageTemperaturePerMonth.AverageTemperaturePerMonthReducer.class, Text.class, DoubleWrit
avgTempPerMonth.waitForCompletion(true);

  //write the output to csv file to plot the graph
  StringBuilder sb = new StringBuilder();
  sb.append("Year and Month" + "\t" + "Avg.Temperature");
  pw.println(sb);
  List<String> records = readLines(outputPath.toString()+ "/" + "part-r-00000");
  for(String record:records){
      String[] tokens = record.split("\t");
      String YearAndMonth = tokens[0] + "/" + tokens[1];
      pw.println(YearAndMonth + "\t" + tokens[2]);
  }

  pw.close();
return 0;
```

Screen shot 4: " updating output to csv"



Screen Shot 5 : Tableau plot averageTemperaturePerMonth