Goals :

To implement K-means algorithm using hadoop map reduce for performanace enhancement .

Procedure :

Traditional K-means clustering algorithm

```
1) Input: K, set of points x 1 ...x n
2) Place centroids c 1 ... c k at random locations
3) Repeat until convergence:
            //Phase 1    Allocation of centroid
      for each point x i :
            find nearest centroid c j
            min D(x i ,c j ) { Euclidean distance}
             assign the xi to cluster j

            // Phase 2  Re allocation of Centroid
      for each cluster j=1 ... K:
            new centroid cj = mean of all point x i assigned to cluster j in previous step
            c j (a) = (1/n j )∑ xi→cj xi(a) for a = 1 ... d
4) stop when none of the cluster assignment change
```

Hadoop Mapreduce K-means clustering algorithm

```
1) Input: K, set of points x 1 ...x n
2) Place centroids c 1 ... c k at random locations
3) Repeat until convergence:
            //Phase 1    Allocation of centroid   --- THIS IS DONE IN MAP function
      for each point x i :
            find nearest centroid c j
            min D(x i ,c j ) { Euclidean distance}
             assign the xi to cluster j

            // Phase 2  Re allocation of Centroid  ----   THIS IS DONE IN REDUCE function
      for each cluster j=1 ... K:
            new centroid cj = mean of all point x i assigned to cluster j in previous step
            c j (a) = (1/n j )∑ xi→cj xi(a) for a = 1 ... d
4) stop when none of the cluster assignment change
```

Execution of algorithm :

classes

KmeansMapper	: Mapper class
KmeansReducer	: Reducer class
Point	: class implementing WritableComparable to represent a point
Kmeans	: class containing all the remaining classes --- Main class

Input:

The Input file will be consisting of points having x and y coordinates separated by comma.

```
1,-1
-99,-98
-78,-98
1,2
2,3
3,4
5,6
2,4
78,79
78,90
90,100
```

this will be the short version of the input file.

What mapper is doing is getting tokens from the tokenizer class as “x,y” and then extracting x and y coordinates by removing the comma. Then it is creating an object of Point class having x and y as their attributes. After that, the nearest centroid is found by checking the distance between them. All the centroids are stored in an array. The proper index of the centroid is then passed as key and the point object as a value to the output collector which is further given as input to the reducer class.

In combiner phase the keys are sorted so that all the points clustered according to a given cemtroid are present in same iterable object . This < key , iterable<value>> is passed to reduce function .

In reduce phase , recaluculation of centroid is done , by considerable all the points in Iterable<values> so that we caluculate mean of the points to get centroid . The new cluster is then written on to output file .

This is continued until new centriod group is same as old centroid pair .

Performance is about 30 sec for 1000 point file .

Expected input and output :
Input and their corresponding output files are attached .