/\*

\* @Student : Kanimozhi Kalaichelvan

\* @Professor : Cong Pu

\* @Marshall ID : 901881363

\* @Date : 17 Nov 2018

\* @Course : CS620 Applied Algorithms

\* @Program : Clustering Algorithm

\*/

import java.util.ArrayList;

// Class to implement K-Means Clustering Algorithm

public class ClustersK {

// Give Number of Clusters required to be 2

private static int NumofClusters = 2, randomNum1, randomNum2,status;

private static float[] mean= new float[NumofClusters],prevmean = new float[NumofClusters];

private static int[] inputsequence = {4,10,1,7,3,8,9,2};//{3,15,2,5,17,1,18,4,20,16}; //{3,15,2,5,17,1,18,4,20,16,11,9,13,14};

private static ArrayList<Integer> Array1 = new ArrayList<Integer>();

private static ArrayList<Integer> Array2 = new ArrayList<Integer>();

// Function to generate Random Number

public int randomGen()

{

int randomNum = (int) (Math.random()\*8);

System.out.println("Randomnly chosen Number is : "+randomNum);

return randomNum;

}

// Function to Calculate Mean of Each Cluster

public static float[] calculateMean()//(float old\_mean1,float old\_mean2)

{

int sum1 = 0, sum2 = 0;

float avg1, avg2, size1, size2;

size1 = Array1.size();

size2 = Array2.size();

for (int i=0; i< Array1.size(); i++)

sum1 += Array1.get(i);

avg1 = sum1 / size1;

for (int j=0; j< Array2.size(); j++)

sum2 += Array2.get(j);;

avg2 = sum2 / size2;

mean[0] = avg1;

mean[1] = avg2;

System.out.println("Mean 0 is : "+mean[0]);

System.out.println("Mean 1 is : "+mean[1]);

return mean;

}

// Function to Calculate the Array with Nearest mean for Each element of the InputSequence Array

public int calculateNearestMean(int inputseqnum)

{

calculateMean();

float minmean = Math.abs(mean[0]-inputseqnum);

int array=0;

System.out.println();

System.out.println("Number is "+inputseqnum+"\nMinmean is "+minmean);

for(int d=0;d<mean.length;d++)

{

float temp = Math.abs(mean[d]-inputseqnum);

if (temp<minmean)

{

minmean = temp;

System.out.println("Minmean is "+minmean);

array = d;

System.out.println("Recommended Array is: "+ array);

}

}

return array;

}

// Function to add Elements Initially

public void addElements()

{

for (int i=0; i< inputsequence.length; i++)

{

// Add Elements to Array1

int arr = calculateNearestMean(inputsequence[i]);

if(arr==0 && i!=randomNum1 && i!=randomNum2)

{

Array1.add(inputsequence[i]);

}

// Add Elements to Array2

else if(arr==1 && i!=randomNum1 && i!=randomNum2)

{

Array2.add(inputsequence[i]);

}

}

}

// Function that adds elements to array in each iteration until mean remains same

public void continueLoopAddElements()

{

for (int i=0; i< inputsequence.length; i++)

{

// Add Elements to Array1

int arr = calculateNearestMean(inputsequence[i]);

if(arr==0)

{

if(Array1.contains(inputsequence[i]))

{

Array1.remove(Array1.indexOf(inputsequence[i]));

}

if(Array2.contains(inputsequence[i]))

{

Array2.remove(Array2.indexOf(inputsequence[i]));

}

Array1.add(inputsequence[i]);

}

// Add Elements to Array2

else if(arr==1)

{

if(Array1.contains(inputsequence[i]))

{

Array1.remove(Array1.indexOf(inputsequence[i]));

}

if(Array2.contains(inputsequence[i]))

{

Array2.remove(Array2.indexOf(inputsequence[i]));

}

Array2.add(inputsequence[i]);

}

}

}

/\* Function to check if mean remains SAME

\* If mean remains same stop the loop

\* If mean is different for all elements of sequence, calculate minimum mean distance and add element to that array

\*/

public static int checkstatus(float[] prevmean)

{

mean=calculateMean();

for(int l=0;l<mean.length;l++)

{

if(mean[l]!=prevmean[l])

{

return status=0;

}

}

status=1;

return status;

}

// MAIN Function where program starts Execution

public static void main(String[] args)

{

// TODO Auto-generated method stub

// Instantiate the Clustering Algorithm class

ClustersK ca = new ClustersK();

randomNum1 = 6;//ca.randomGen();

Array1.add(inputsequence[randomNum1]);

System.out.println("Array 1 is "+Array1);

randomNum2 = 1;//ca.randomGen();

while(randomNum2==randomNum1)

{

randomNum2 = ca.randomGen();

}

Array2.add(inputsequence[randomNum2]);

System.out.println("Array 2 is "+Array2);

System.out.println();

calculateMean();

System.out.println();

ca.addElements();

System.out.println("Array 1 is "+Array1);

System.out.println("Array 2 is "+Array2);

System.out.println();

while(status!=1)

{

prevmean = calculateMean();

ca.continueLoopAddElements();

status = checkstatus(prevmean);

System.out.println("Array 1 is "+Array1);

System.out.println("Array 2 is "+Array2);

System.out.println();

}

System.out.println("Cluster 1 is "+Array1);

System.out.println("Cluster 2 is "+Array2);

}

}

Array 1 is [9]

Array 2 is [10]

Mean 0 is : 9.0

Mean 1 is : 10.0

Mean 0 is : 9.0

Mean 1 is : 10.0

Number is 4

Minmean is 5.0

Mean 0 is : 6.5

Mean 1 is : 10.0

Number is 10

Minmean is 3.5

Minmean is 0.0

Recommended Array is: 1

Mean 0 is : 6.5

Mean 1 is : 10.0

Number is 1

Minmean is 5.5

Mean 0 is : 4.6666665

Mean 1 is : 10.0

Number is 7

Minmean is 2.3333335

Mean 0 is : 5.25

Mean 1 is : 10.0

Number is 3

Minmean is 2.25

Mean 0 is : 4.8

Mean 1 is : 10.0

Number is 8

Minmean is 3.1999998

Minmean is 2.0

Recommended Array is: 1

Mean 0 is : 4.8

Mean 1 is : 9.0

Number is 9

Minmean is 4.2

Minmean is 0.0

Recommended Array is: 1

Mean 0 is : 4.8

Mean 1 is : 9.0

Number is 2

Minmean is 2.8000002

Array 1 is [9, 4, 1, 7, 3, 2]

Array 2 is [10, 8]

Mean 0 is : 4.3333335

Mean 1 is : 9.0

Mean 0 is : 4.3333335

Mean 1 is : 9.0

Number is 4

Minmean is 0.3333335

Mean 0 is : 4.3333335

Mean 1 is : 9.0

Number is 10

Minmean is 5.6666665

Minmean is 1.0

Recommended Array is: 1

Mean 0 is : 4.3333335

Mean 1 is : 9.0

Number is 1

Minmean is 3.3333335

Mean 0 is : 4.3333335

Mean 1 is : 9.0

Number is 7

Minmean is 2.6666665

Minmean is 2.0

Recommended Array is: 1

Mean 0 is : 3.8

Mean 1 is : 8.333333

Number is 3

Minmean is 0.79999995

Mean 0 is : 3.8

Mean 1 is : 8.333333

Number is 8

Minmean is 4.2

Minmean is 0.33333302

Recommended Array is: 1

Mean 0 is : 3.8

Mean 1 is : 8.333333

Number is 9

Minmean is 5.2

Minmean is 0.666667

Recommended Array is: 1

Mean 0 is : 2.5

Mean 1 is : 8.5

Number is 2

Minmean is 0.5

Mean 0 is : 2.5

Mean 1 is : 8.5

Array 1 is [4, 1, 3, 2]

Array 2 is [10, 7, 8, 9]

Cluster 1 is [4, 1, 3, 2]

Cluster 2 is [10, 7, 8, 9]

Array 1 is [18]

Array 2 is [15]

Array 1 is [18, 17, 20, 16]

Array 2 is [15, 3, 2, 5, 1, 4]

Array 1 is [15, 17, 18, 20, 16]

Array 2 is [3, 2, 5, 1, 4]

Cluster 1 is [15, 17, 18, 20, 16]

Cluster 2 is [3, 2, 5, 1, 4]

/\*

\* @Student : Kanimozhi Kalaichelvan

\* @Professor : Cong Pu

\* @Marshall ID : 901881363

\* @Date : 17 Nov 2018

\* @Course : CS620 Applied Algorithms

\* @Program : Clustering Algorithm

\*/

import java.util.ArrayList;

// Class to implement K-Means Clustering Algorithm

public class ClustersK {

// Give Number of Clusters required to be 2

private static int NumofClusters = 2, randomNum1, randomNum2,status;

private static float[] mean= new float[NumofClusters],prevmean = new float[NumofClusters];

private static int[] inputsequence = {3,15,2,5,17,1,18,4,20,16};//{4,10,1,7,3,8,9,2};//{3,15,2,5,17,1,18,4,20,16}; //{3,15,2,5,17,1,18,4,20,16,11,9,13,14};

private static ArrayList<Integer> Array1 = new ArrayList<Integer>();

private static ArrayList<Integer> Array2 = new ArrayList<Integer>();

// Function to generate Random Number

public int randomGen()

{

int randomNum = (int) (Math.random()\*8);

System.out.println("Randomnly chosen Number is : "+randomNum);

return randomNum;

}

// Function to Calculate Mean of Each Cluster

public static float[] calculateMean()//(float old\_mean1,float old\_mean2)

{

int sum1 = 0, sum2 = 0;

float avg1, avg2, size1, size2;

size1 = Array1.size();

size2 = Array2.size();

for (int i=0; i< Array1.size(); i++)

sum1 += Array1.get(i);

avg1 = sum1 / size1;

for (int j=0; j< Array2.size(); j++)

sum2 += Array2.get(j);;

avg2 = sum2 / size2;

mean[0] = avg1;

mean[1] = avg2;

// System.out.println("Mean 0 is : "+mean[0]);

// System.out.println("Mean 1 is : "+mean[1]);

return mean;

}

// Function to Calculate the Array with Nearest mean for Each element of the InputSequence Array

public int calculateNearestMean(int inputseqnum)

{

calculateMean();

float minmean = Math.abs(mean[0]-inputseqnum);

int array=0;

// System.out.println();

// System.out.println("Number is "+inputseqnum+"\nMinmean is "+minmean);

for(int d=0;d<mean.length;d++)

{

float temp = Math.abs(mean[d]-inputseqnum);

if (temp<minmean)

{

minmean = temp;

// System.out.println("Minmean is "+minmean);

array = d;

// System.out.println("Recommended Array is: "+ array);

}

}

return array;

}

// Function to add Elements Initially

public void addElements()

{

for (int i=0; i< inputsequence.length; i++)

{

// Add Elements to Array1

int arr = calculateNearestMean(inputsequence[i]);

if(arr==0 && i!=randomNum1 && i!=randomNum2)

{

Array1.add(inputsequence[i]);

}

// Add Elements to Array2

else if(arr==1 && i!=randomNum1 && i!=randomNum2)

{

Array2.add(inputsequence[i]);

}

}

}

// Function that adds elements to array in each iteration until mean remains same

public void continueLoopAddElements()

{

for (int i=0; i< inputsequence.length; i++)

{

// Add Elements to Array1

int arr = calculateNearestMean(inputsequence[i]);

if(arr==0)

{

if(Array1.contains(inputsequence[i]))

{

Array1.remove(Array1.indexOf(inputsequence[i]));

}

if(Array2.contains(inputsequence[i]))

{

Array2.remove(Array2.indexOf(inputsequence[i]));

}

Array1.add(inputsequence[i]);

}

// Add Elements to Array2

else if(arr==1)

{

if(Array1.contains(inputsequence[i]))

{

Array1.remove(Array1.indexOf(inputsequence[i]));

}

if(Array2.contains(inputsequence[i]))

{

Array2.remove(Array2.indexOf(inputsequence[i]));

}

Array2.add(inputsequence[i]);

}

}

}

/\* Function to check if mean remains SAME

\* If mean remains same stop the loop

\* If mean is different for all elements of sequence, calculate minimum mean distance and add element to that array

\*/

public static int checkstatus(float[] prevmean)

{

mean=calculateMean();

for(int l=0;l<mean.length;l++)

{

if(mean[l]!=prevmean[l])

{

return status=0;

}

}

status=1;

return status;

}

// MAIN Function where program starts Execution

public static void main(String[] args)

{

// TODO Auto-generated method stub

// Instantiate the Clustering Algorithm class

ClustersK ca = new ClustersK();

randomNum1 = 6;//ca.randomGen();

Array1.add(inputsequence[randomNum1]);

System.out.println("Array 1 is "+Array1);

randomNum2 = 1;//ca.randomGen();

while(randomNum2==randomNum1)

{

randomNum2 = ca.randomGen();

}

Array2.add(inputsequence[randomNum2]);

System.out.println("Array 2 is "+Array2);

System.out.println();

calculateMean();

System.out.println();

ca.addElements();

System.out.println("Array 1 is "+Array1);

System.out.println("Array 2 is "+Array2);

System.out.println();

while(status!=1)

{

prevmean = calculateMean();

ca.continueLoopAddElements();

status = checkstatus(prevmean);

System.out.println("Array 1 is "+Array1);

System.out.println("Array 2 is "+Array2);

System.out.println();

}

System.out.println("Cluster 1 is "+Array1);

System.out.println("Cluster 2 is "+Array2);

}

}