**CSC 323-21: Project7: K means Clustering problem (CPP)**

**Adil Abuwani**

**Due date: Nov. 13, 2018**

**Algorithm Steps for Computing:**

**main (..)**

Step 0:

inFile 🡨 Open the input file argv[]

numRows, numCols, minVal, maxVal🡨 get from inFile.

imgAry 🡨 Dynamically allocate a 2-D arrays, size numRows X numCols.

outFile\_1, outFile\_2, outFile\_3 🡨 open from argv[], argv[], argv[]

K 🡨 ask user from console

Kcentroids[K] 🡨 Dynamically allocate centroids struct, size of K+1

Step 1: numPts 🡨 extractPts (inFile, outFile\_1) // see algorithm below.

pointSet 🡨 Dynamically allocate the pointSet array, size of numPts

Step 2: close outFile\_1

outFile\_1 🡨 reopen outFile\_1

Step 3: loadPointSet (outFile\_1, pointSet)

Step 4: kMeansClustering (pointSet, K)

Step 5: writePtSet (pointSet, outFile\_3)

Step 6: close all files

**int extractPts(inFile, outFile\_1)**

Step 0: numPts 🡨 0

Step 1: read inFile from left to right and top to bottom

Use i, j as index for rows and cols bounded by numRows and numCols

Step 2: if p(i, j) > 0

numPts++

write i and j to outFile1

Step 3: repeat step 1 and step 2 until EOF inFile

Step 4: return numPts

**kMeansClustering (pointSet, K)**

Step 0: iteration 🡨 0

Step 1: assignLable (pointSet, K)

Step 2: Point2Image (pointSet, imgAry)

printImage (imgAry, outFile\_2, iteration)

Step 3: change 🡨 0

Step 4: computeCentroids (pointSet, Kcentroids) // see algorithm below

Step 5: index 🡨 0

Step 6: DistanceMinLable(pointSet[index], Kcentroids) // see algorithm below

Step 7: index ++

Step 8: repeat step 6 to step 7 while index < numPts

Step 9: iteration++

Step 10: repeat step 2 to step 9 until no more changes

**computeCentroids (pointSet, Kcentroids )**

Step 0: sumX[K] 🡨 dynamically allocate 1-D array, size of K

sumY[K] 🡨 dynamically allocate 1-D array, size of K

totalPt[K] 🡨 dynamically allocate 1-D array, size of K

// initialize the above 3 1D arrays to 0

Step 1: index 🡨 0

Step 2: label 🡨 pointSet[index].label // get the point’s cluster label

sumX[label] += pointSet[index].Xcoord

sumY[label] += pointSet[index].Ycoord

totalPt[label] ++

Step 3: index++

Step 4: repeat step 2 to step 3 while index < numPts

step 5: label 🡨 1

step 6: Kcentroids[label].Xcoord 🡨 (sumX[label]/ totalPt[label])

Kcentroids[label].Ycoord 🡨 (sumY[label]/ totalPt[label])

Step 7: label ++

Step 8: repeat step 6 to step 7 while label <= K

**DistanceMinLable (p, Kcentroids)**

Step 1: label 🡨 1

Step 2: dist 🡨 computeDist(p, Kcentroids[label]) //

if p’s Distance > dist

p’s label 🡨 label

change++

Step 3: label ++

Step 4: repeat step 2 to step 3 while label <= K

**assignLabel (pointSet, K)**  
Step 0: front <-- 0  
           back <-- numPts - 1  
           label <-- 1  
Step 1: if label >= K  
          Label <-- 1  
Step 2:  pointSet[front]'s label <-- label  
            front ++  
            label++  
Step 3:  pointSet[back]'s label <-- label  
            back --  
            label++  
Step 4: repeat step 1 to step 3 while front <= back

b) Source code in CPP:

**#include** <iostream>

**#include** <fstream>

**#include** <cmath>

**using** **namespace** std;

**class** Point{

**public**:

**int** Xcoord;

**int** Ycoord;

**int** Label;

**double** Distance; //the distance to its own cluster centroid

**Point**(){

**this**->Xcoord=0;

**this**->Ycoord=0;

**this**->Label=0;

**this**->Distance=9999.0; //initial

}

**Point**(**int** xc, **int** yc, **int** l, **double** d){

**this**->Xcoord=xc;

**this**->Ycoord=yc;

**this**->Label=l;

**this**->Distance=d;

}

};

**class** xyCoord{

**public**:

**int** Xcoord;

**int** Ycoord;

**int** Label;

**xyCoord**(){

**this**->Xcoord=0;

**this**->Ycoord=0;

**this**->Label=0;

}

}; //end classXYcord

**class** Kmean{

**public**:

**int** K;

**int** numPts; //initialize to 0

Point\* pointSet; //1d array of point class to be dynamacally allocated

**int** numRows;

**int** numCols;

**int** minVal;

**int** maxVal;

xyCoord\* Kcentroids;

**int**\*\* imgAry;

**int** change;

**Kmean**(**int** k, **int** nr, **int** nc, **int** minVal, **int** maxVal){

**this**->K=k;

**this**->numRows=nr;

**this**->numCols=nc;

**this**->minVal=minVal;

**this**->maxVal=maxVal;

**this**->numPts=0; //we will initialize to 0

**this**->change=0;

**this**->pointSet= **new** Point[**this**->numPts]; //point set to 0 at first

**this**->Kcentroids= **new** xyCoord[**this**->K+1]; //we dont want to consider the 0

//initialize the imgArr

**this**->imgAry= **new** **int**\*[**this**->numRows];

**for**(**int** i=0; i<nr; i++){

**this**->imgAry[i]= **new** **int**[nc];

}//end for

}//end constructor

**int** **extractPts**(ofstream& outFile1){

**this**->numPts=0;

**for**(**int** i=0; i<**this**->numRows; i++){

**for**(**int** j=0; j<**this**->numCols; j++){

**if**(**this**->imgAry[i][j]>0){

**this**->numPts++; //number of points that are greater then 0

outFile1<<i<<" "<<j<<**endl**;

}//end if

}

}

**this**->pointSet= **new** Point[**this**->numPts]; //initialize pointset with updated numPts

**return** **this**->numPts;

}//end extractPoints

**void** **loadPointSet** (ifstream& outFile1){

//read, and store in the pointset array, and assign a label

//begin from 1

**int** xCord;

**int** yCord;

**int** i=0;

**while**(outFile1>>xCord>>yCord && i<**this**->numPts){

**this**->pointSet[i].Xcoord=xCord;

**this**->pointSet[i].Ycoord=yCord;

**this**->pointSet[i].Label=0;

**this**->pointSet[i].Distance=9999.0;

i++;

}//end while

}

**void** **assignLabel**(){

**int** front=0;

**int** back=numPts-1;

**int** label=1;

**while**(front<=back){

**if**(label>K){

label=1;

}

**this**->pointSet[front].Label=label;

front++;

label++;

**if**(label>K){

label=1;

}

pointSet[back].Label=label;

back--;

label++;

}

}

**void** **writePtSet**(ofstream& outFile3){

outFile3<<**this**->numPts<<**endl**;

outFile3<<**this**->numRows<<" "<<**this**->numCols<<**endl**;

**for**(**int** i=0; i<**this**->numPts; i++){

outFile3<<pointSet[i].Xcoord<<" "<<pointSet[i].Ycoord<<" "<<pointSet[i].Label<<**endl**;

}//end for

}

**void** **Point2Image**(){

**for**(**int** i=0;i<**this**->numPts;i++){

imgAry[pointSet[i].Xcoord][pointSet[i].Ycoord] = pointSet[i].Label;

}

}//end Point2Image

**void** **printImage** (ofstream& outFile2, **int** it){

outFile2 << "\*\*\* Result of iteration " <<it<< " \*\*\*"<<**endl**;

**for**(**int** i=0;i<**this**->numRows;i++){

**for**(**int** j=0;j<numCols;j++){

**if**(**this**->imgAry[i][j]>0){

outFile2<<**this**->imgAry[i][j]<<" "; //display its label

}

**else**{

outFile2 <<" "; //display with a blank

}

}

outFile2<<**endl**;

}

outFile2<<**endl**<<**endl**;

}//end printImage

//

**double** **computeDist**(Point p1, xyCoord Kcentroids){ //distance between point p1, and a sentroid

**double** distance = **sqrt**(pow((p1.Xcoord - Kcentroids.Xcoord),2) + pow((p1.Ycoord-Kcentroids.Ycoord),2));

**return** distance;

}

**void** **computeCentroids**(){

**int** sumX[**this**->K+1] = {0};

**int** sumY[**this**->K+1] = {0};

**int** totalPt[**this**->K+1] = {0};

**int** label;

**int** index = 0; //begin from the first

**while**(index<**this**->numPts){

label = pointSet[index].Label; //get label of the pointset

sumX[label]+=pointSet[index].Xcoord; //add the x cordinates with that lable

sumY[label]+= pointSet[index].Ycoord; //add the Y cordinates with that lable

totalPt[label]++; //total number of points for that lable

index++;

}

label = 1;

**while**(label<=K){ //get the centroid for each lable

**if**(totalPt[label]>0){

Kcentroids[label].Xcoord = (**int**)(sumX[label]/totalPt[label]);

Kcentroids[label].Ycoord = (**int**)(sumY[label]/totalPt[label]);

}

label++;

}//end while

}

**void** **DistanceMinLabel**(**int** index){

// compute the distance from a point p to each of the K centroids.

// check to see if p’s label need to be change or not.

//step 0

**double** minDistance=99999;

**int** minLabel=0;

**int** label = 1;

**double** dist=0.0;

**while**(label<=K){

dist = computeDist(**this**->pointSet[index], Kcentroids[label]); //al the K centroids

**if**(dist<minDistance){

minLabel=label;

minDistance=dist;

}//end if

label++;

}//end while

**if**(minDistance < **this**->pointSet[index].Distance){

**this**->pointSet[index].Distance=minDistance;

**this**->pointSet[index].Label=minLabel;

**this**->change++;

}

}

**void** **kMeansClustering**(ofstream& outFile2){

**int** it = 0;

**this**->assignLabel();

**this**->change = 0;

**do**{

**this**->Point2Image();

**this**->printImage(outFile2, it);

**this**->change = 0;

**this**->computeCentroids(); //compute centriods of each

**int** index=0;

**while**(index<**this**->numPts){

DistanceMinLabel(index);

index++;

}//end while

it++;

} **while** (change!=0);

}//end KmeanClustering

};

**int** **main**(**int** argc, **char**\*\*argv) {

//step1

ifstream inFile1;

ofstream outFile1;

ofstream outFile2;

ofstream outFile3;

inFile1.open(argv[1]);

outFile1.open(argv[2]);

outFile2.open(argv[3]);

outFile3.open(argv[4]);

**int** numRows;

**int** numCols;

**int** minVal;

**int** maxVal;

**int** k;

Kmean\* theKmean;

**if**(inFile1.is\_open()){

inFile1>>numRows;

inFile1>>numCols;

inFile1>>minVal;

inFile1>>maxVal;

cout<<"Enter Number of klusters K"<<**endl**;

cin>>k;

cout<<**endl**;

theKmean= **new** Kmean(k, numRows, numCols, minVal, maxVal);

//imgAry Dynamically allocate a 2-D arrays, size numRows X numCols.

**for**(**int** i=0; i<numRows; i++){

**for**(**int** j=0; j<numCols; j++){

inFile1>>theKmean->imgAry[i][j];

}

}

//Step 1: numPts  extractPts (inFile, outFile\_1) // see algorithm below.

// pointSet  Dynamically allocate the pointSet array, size of numPts

**int** numPts=theKmean->extractPts(outFile1);

//step2-close outFile, and open it again

outFile1.close();

ifstream myFile;

myFile.open(argv[2]); //has the outFile1

//STEP 3:loadPointSet, and assign label

theKmean->loadPointSet(myFile);

//step4

theKmean->kMeansClustering(outFile2);

//step5

theKmean->writePtSet(outFile3);

inFile1.close();

outFile1.close();

outFile2.close();

outFile3.close();

}**else**{

cerr<<"Can not open the file!";

}

**return** 0;

}

c) Input- a text file argv[1]:

35 40 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0

0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0

0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0

0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0

0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0

0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0

0 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0

0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0

0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0

0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0

0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0

0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

d) OutputFile1- a text file arv[2]:

1 6

1 7

2 4

3 3

3 5

3 7

4 4

4 5

4 6

4 8

4 31

4 32

5 3

5 5

5 7

5 29

5 30

6 4

6 5

6 8

6 9

6 10

6 25

6 26

6 28

6 29

7 2

7 4

7 5

7 6

7 8

7 24

7 26

7 29

7 30

8 2

8 3

8 4

8 5

8 7

8 22

8 24

8 25

8 27

8 28

8 29

8 31

8 32

9 3

9 6

9 7

9 8

9 9

9 20

9 23

9 24

9 26

9 29

9 32

9 33

10 2

10 3

10 5

10 7

10 8

10 10

10 19

10 22

10 25

10 27

10 28

10 30

10 31

10 33

10 34

11 4

11 5

11 7

11 9

11 12

11 22

11 23

11 27

11 30

11 31

12 3

12 4

12 6

12 7

12 10

12 11

12 25

12 27

12 29

12 31

13 5

13 6

13 8

13 9

13 24

13 26

13 28

13 30

13 32

13 33

14 7

14 25

14 26

14 30

14 32

14 33

15 4

15 5

15 9

15 10

15 21

15 23

15 26

15 28

15 29

15 31

15 32

16 7

16 24

16 27

16 29

16 30

17 26

17 28

17 31

17 32

18 27

18 29

19 24

19 28

22 7

22 10

23 28

24 3

24 4

24 6

24 7

24 8

25 5

25 6

25 28

26 4

26 7

26 8

26 27

26 28

26 29

26 30

27 3

27 5

27 6

27 7

27 8

27 24

27 25

27 28

27 29

28 4

28 6

28 7

28 9

28 10

28 11

28 12

28 14

28 23

28 24

28 25

28 26

28 29

28 30

29 2

29 3

29 5

29 6

29 7

29 8

29 24

29 25

29 28

29 29

29 31

29 32

30 4

30 5

30 7

30 10

30 11

30 20

30 23

30 24

30 26

30 27

30 32

30 33

31 3

31 5

31 6

31 9

31 10

31 26

31 27

31 29

31 30

31 33

31 34

32 2

32 3

32 6

32 7

32 25

32 26

32 28

32 29

32 32

32 33

33 26

33 27

33 30

33 31

e) OutputFile2- a text file arv[3]:

\*\*\* Result of iteration 0 \*\*\*

1 3

1

3 1 3

1 3 1 3 1 3

1 3 1 3 1

3 1 3 1 3 1 3 1 3

1 3 1 3 1 3 1 3 1

3 1 3 1 3 1 3 1 3 1 3 1 3

1 3 1 3 1 3 1 3 1 3 1 3

1 3 1 3 1 3 1 3 1 3 1 3 1 3 1

3 1 3 1 3 1 3 1 3 1

3 1 3 1 3 1 3 1 3 1

3 1 3 1 3 1 3 1 3 1

3 1 3 1 3 1

3 2 4 2 4 2 4 2 4 2 4

2 4 2 4 2

4 2 4 2

4 2

4 2

4 2

4

2 4 2 4 2

4 2 4

2 4 2 4 2 4 2

4 2 4 2 4 2 4 2 4

2 4 2 4 2 4 2 4 2 4 2 4 2 4

2 4 2 4 2 4 2 4 2 4 2 4

2 4 2 4 2 4 2 4 2 4 2 4

2 4 2 4 2 4 2 4 2 4 2

4 2 4 2 4 2 4 2 4 2

4 2 4 2

\*\*\* Result of iteration 1 \*\*\*

3 3

3

3 3 3

3 3 3 3 1 1

3 3 3 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1

3 3 3 3 1 1 1 1 1 1

3 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1

3 1 1 4 4

4 4 4 4

4 4

4 4

2 2

4

2 2 2 2 2

2 2 4

2 2 2 4 4 4 4

2 2 2 2 2 4 4 4 4

2 2 2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 4 4 4 4 4 4

4 4 4 4

\*\*\* Result of iteration 2 \*\*\*

3 3

3

3 3 3

3 3 3 3 1 1

3 3 3 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1

3 3 3 3 1 1 1 1 1 1

3 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1

3 1 1 1 1

1 1 1 1

1 1

4 4

2 2

4

2 2 2 2 2

2 2 4

2 2 2 4 4 4 4

2 2 2 2 2 4 4 4 4

2 2 2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 4 4 4 4 4 4

4 4 4 4

\*\*\* Result of iteration 3 \*\*\*

3 3

3

3 3 3

3 3 3 3 1 1

3 3 3 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1 1 1 1 1 1

3 3 3 3 3 1 1 1 1 1

3 3 3 3 3 3 1 1 1 1

3 3 3 3 1 1 1 1 1 1

3 1 1 1 1 1

3 3 3 3 1 1 1 1 1 1 1

3 1 1 1 1

1 1 1 1

1 1

4 4

2 2

4

2 2 2 2 2

2 2 4

2 2 2 4 4 4 4

2 2 2 2 2 4 4 4 4

2 2 2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4 4

2 2 2 2 2 4 4 4 4 4 4

2 2 2 2 4 4 4 4 4 4

4 4 4 4

e) OutputFile3- a text file arv[3]:

225

35 40

1 6 3

1 7 3

2 4 3

3 3 3

3 5 3

3 7 3

4 4 3

4 5 3

4 6 3

4 8 3

4 31 1

4 32 1

5 3 3

5 5 3

5 7 3

5 29 1

5 30 1

6 4 3

6 5 3

6 8 3

6 9 3

6 10 3

6 25 1

6 26 1

6 28 1

6 29 1

7 2 3

7 4 3

7 5 3

7 6 3

7 8 3

7 24 1

7 26 1

7 29 1

7 30 1

8 2 3

8 3 3

8 4 3

8 5 3

8 7 3

8 22 1

8 24 1

8 25 1

8 27 1

8 28 1

8 29 1

8 31 1

8 32 1

9 3 3

9 6 3

9 7 3

9 8 3

9 9 3

9 20 1

9 23 1

9 24 1

9 26 1

9 29 1

9 32 1

9 33 1

10 2 3

10 3 3

10 5 3

10 7 3

10 8 3

10 10 3

10 19 1

10 22 1

10 25 1

10 27 1

10 28 1

10 30 1

10 31 1

10 33 1

10 34 1

11 4 3

11 5 3

11 7 3

11 9 3

11 12 3

11 22 1

11 23 1

11 27 1

11 30 1

11 31 1

12 3 3

12 4 3

12 6 3

12 7 3

12 10 3

12 11 3

12 25 1

12 27 1

12 29 1

12 31 1

13 5 3

13 6 3

13 8 3

13 9 3

13 24 1

13 26 1

13 28 1

13 30 1

13 32 1

13 33 1

14 7 3

14 25 1

14 26 1

14 30 1

14 32 1

14 33 1

15 4 3

15 5 3

15 9 3

15 10 3

15 21 1

15 23 1

15 26 1

15 28 1

15 29 1

15 31 1

15 32 1

16 7 3

16 24 1

16 27 1

16 29 1

16 30 1

17 26 1

17 28 1

17 31 1

17 32 1

18 27 1

18 29 1

19 24 4

19 28 4

22 7 2

22 10 2

23 28 4

24 3 2

24 4 2

24 6 2

24 7 2

24 8 2

25 5 2

25 6 2

25 28 4

26 4 2

26 7 2

26 8 2

26 27 4

26 28 4

26 29 4

26 30 4

27 3 2

27 5 2

27 6 2

27 7 2

27 8 2

27 24 4

27 25 4

27 28 4

27 29 4

28 4 2

28 6 2

28 7 2

28 9 2

28 10 2

28 11 2

28 12 2

28 14 2

28 23 4

28 24 4

28 25 4

28 26 4

28 29 4

28 30 4

29 2 2

29 3 2

29 5 2

29 6 2

29 7 2

29 8 2

29 24 4

29 25 4

29 28 4

29 29 4

29 31 4

29 32 4

30 4 2

30 5 2

30 7 2

30 10 2

30 11 2

30 20 4

30 23 4

30 24 4

30 26 4

30 27 4

30 32 4

30 33 4

31 3 2

31 5 2

31 6 2

31 9 2

31 10 2

31 26 4

31 27 4

31 29 4

31 30 4

31 33 4

31 34 4

32 2 2

32 3 2

32 6 2

32 7 2

32 25 4

32 26 4

32 28 4

32 29 4

32 32 4

32 33 4

33 26 4

33 27 4

33 30 4

33 31 4