****
CSCI 381-26 Project 9: Hough Transform for line detection Language: JAVA

Name: Akshar Patel

Due date: Soft copy: 4/12/2020
Hard copy: 4/12/2020

Submission date: Soft copy: 4/12/2020
Hard copy: 4/12/2020

****

Part 1: Algorithm

```
*******************************
I. main (…)
*******************************
step 0: inFile ← open input file from args
        outFile1, outFile2 ← open from args
        numRows, numCols, minVal, maxVal ←- read from inFile
        HoughAngle ← 180
        HoughDist ←2 * (the diagonal of the input image)
        imgAry ← dynamically allocate
        HoughAry ← dynamically allocate HoughAry, size of
                   HoughDist by HoughAngle and initialize to zero

step 1: loadImage(imgAry, inFile)
Step 2: buildHoughSpace (...)
Step 3: prettyPrint(HoughAry, outFile1)
Step 4: determineMinMax (HoughAry)
Step 5: write HoughDist, HoughAngle, HoughMinVal, HoughMaxVal to outFile2
            // as the header of Hough image
step 6: ary2File (HoughAry, outFile2) // output HoughAry to outFile2

Step 7: close all files
```

Part 2: Source code

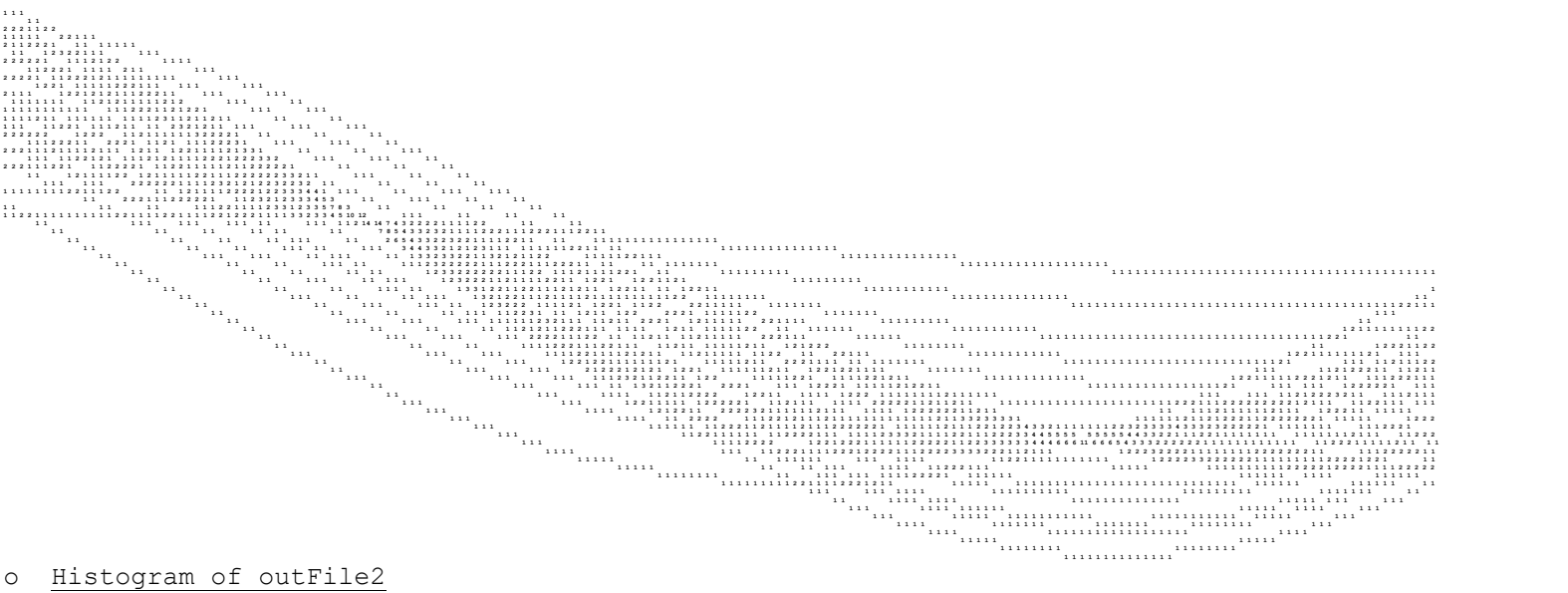
```
import java.io.*;
import java.util.*;
class Main {
       public static int numRows, numCols, minVal, maxVal;
    public static int[][] imgAry;
    static void loadImage(int[][] Ary,Scanner file){
        for(int i = 0 ; i < numRows ; i++){
            for(int j = 0 ; j < numCols ; j++){
                Ary[i][j] = file.nextInt();
            }
        }
    }
    public static void main(String[] args) throws IOException{
        HoughTransform HT = new HoughTransform();
        Scanner inFile = new Scanner(new FileInputStream(args[0]));
        PrintWriter outFile1 = new PrintWriter(new FileOutputStream(args[1]));
             PrintWriter outFile2 = new PrintWriter(new FileOutputStream(args[2]));
           numRows = inFile.nextInt();
        numCols = inFile.nextInt();
        minVal = inFile.nextInt();
        maxVal = inFile.nextInt();
        int diagonal = (int) Math.sqrt((numRows * numRows) + (numCols * numCols));
        HT.HoughAngle = 180;
        HT.HoughDist = 2 * diagonal;
        imgAry = new int[numRows][numCols];
        HT.HoughAry = new int[HT.HoughDist][HT.HoughAngle];
        for(int i = 0 ; i < HT.HoughDist ; i++){
```

```
            for(int j = 0 ; j < HT.HoughAngle ; j++){
                HT.HoughAry[i][j] = 0;
            }
        }
        loadImage(imgAry, inFile);
        HT.buildHoughSpace(imgAry, numRows, numCols);
        HT.prettyPrint(HT.HoughAry, outFile1);
        HT.determineMinMax(HT.HoughAry);
        outFile2.println(HT.HoughDist + " " + HT.HoughAngle + " " + HT.HoughMinVal + " " + HT.HoughMaxVal);
        HT.ary2File(HT.HoughAry, outFile2);
        inFile.close();
        outFile1.close();
        outFile2.close();
    }
}
class HoughTransform{
    private static class xyCoord{
        int x;
        int y;
        xyCoord(){
            this.x = x;
            this.y = y;
        }
    }
    public static int angleInDegree;
       public static double angleInRadians;
       public static int HoughDist;
    public static int HoughAngle;
    public static int HoughMinVal = 99999;
    public static int HoughMaxVal = 0;
    public static int[][] HoughAry;
    static void buildHoughSpace(int[][] Ary, int row, int col){
        xyCoord point = new xyCoord();
        for(int r = 0 ; r < row ; r++){
            for(int c = 0 ; c < col ; c++){
                if(Ary[r][c] > 0){
                    point.x = c;
                    point.y = r;
                    angleInDegree = 0;
                    while(angleInDegree <= 179){
                        angleInRadians = angleInDegree / 180.00 * Math.PI;
                        double dist = computeDistance(point, angleInRadians);
                        int distInt = (int) dist;
                        HoughAry[distInt][angleInDegree]++;
                        angleInDegree++;
                    }
                }
            }
        }
    }
    static double computeDistance(xyCoord point,double angleRadians){
        double dis = 0.00, offset = HoughDist/2;
        double x = point.x;
        double y = point.y;
        double t = angleRadians - Math.atan(y/x) - (Math.PI/2);
        dis = Math.sqrt((x * x)+(y * y)) * Math.cos(t) + offset;
        return dis;
    }
    public static void determineMinMax(int[][] Ary){
        for(int i = 0 ; i < HoughDist; i++){
            for(int j = 0 ; j < HoughAngle ; j++){
                if(Ary[i][j] > HoughMaxVal){
                    HoughMaxVal = Ary[i][j];
                }
                if (Ary[i][j] < HoughMinVal) {
                    HoughMinVal = Ary[i][j];
                }
            }
        }
```

```
    }
    public static void prettyPrint(int[][] Ary, PrintWriter file){
        for(int i = 0 ; i < HoughDist; i++){
            for(int j = 0 ; j < HoughAngle ; j++){
                if(Ary[i][j] == 0){
                                file.print("  ");
                            }
                            else{
                                file.print(Ary[i][j] + " ");
                            }
            }
            file.println();
        }
    }
    public static void ary2File(int[][] Ary, PrintWriter file){
        for(int i = 0 ; i < HoughDist ; i++){
            for(int j = 0 ; j < HoughAngle ; j++){
                            if(Ary[i][j] >=1 ){
                                file.print(Ary[i][j] + " ");
                            }
                            else{
                                file.print(0 + " ");
                            }
            }
            file.println();
        }
    }
}
```

Part 3: Output

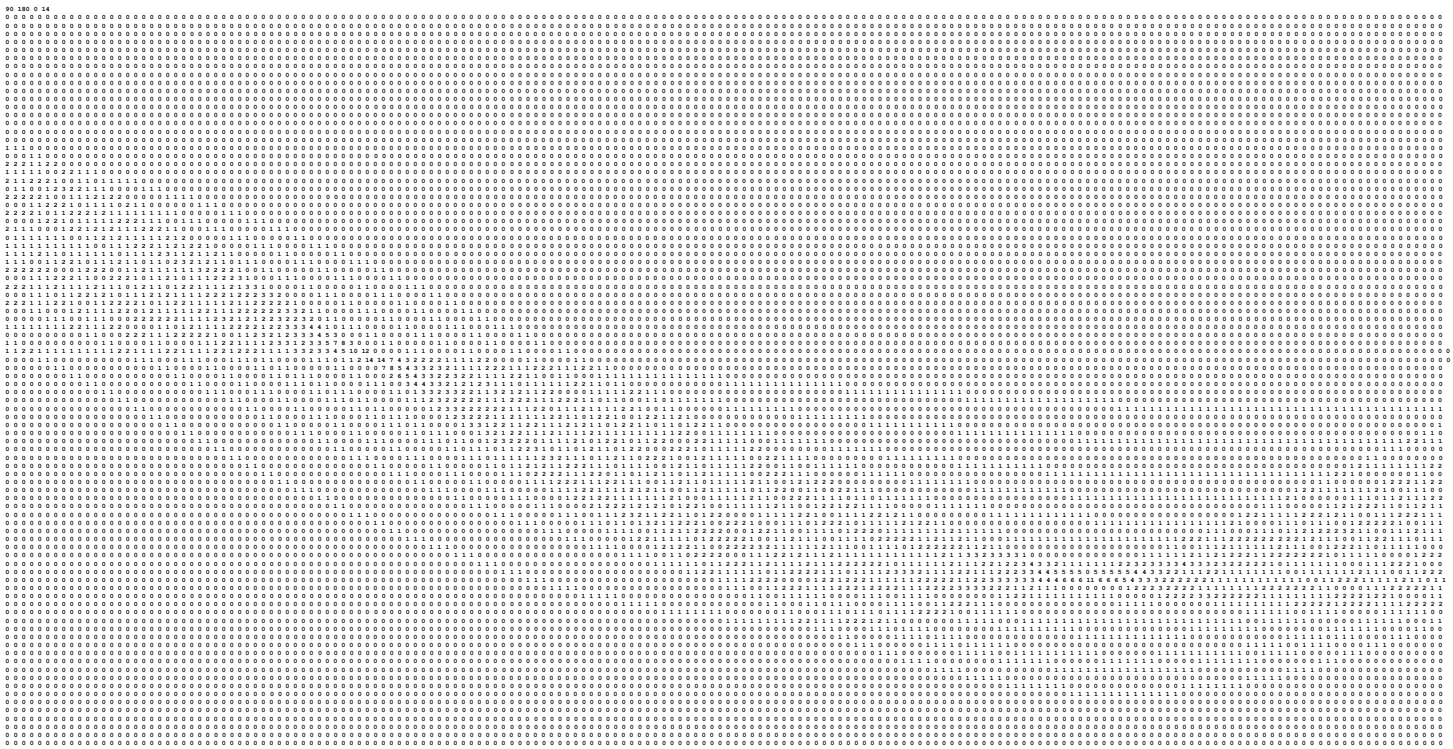
- HoughTransform_2lines
  - outFile1



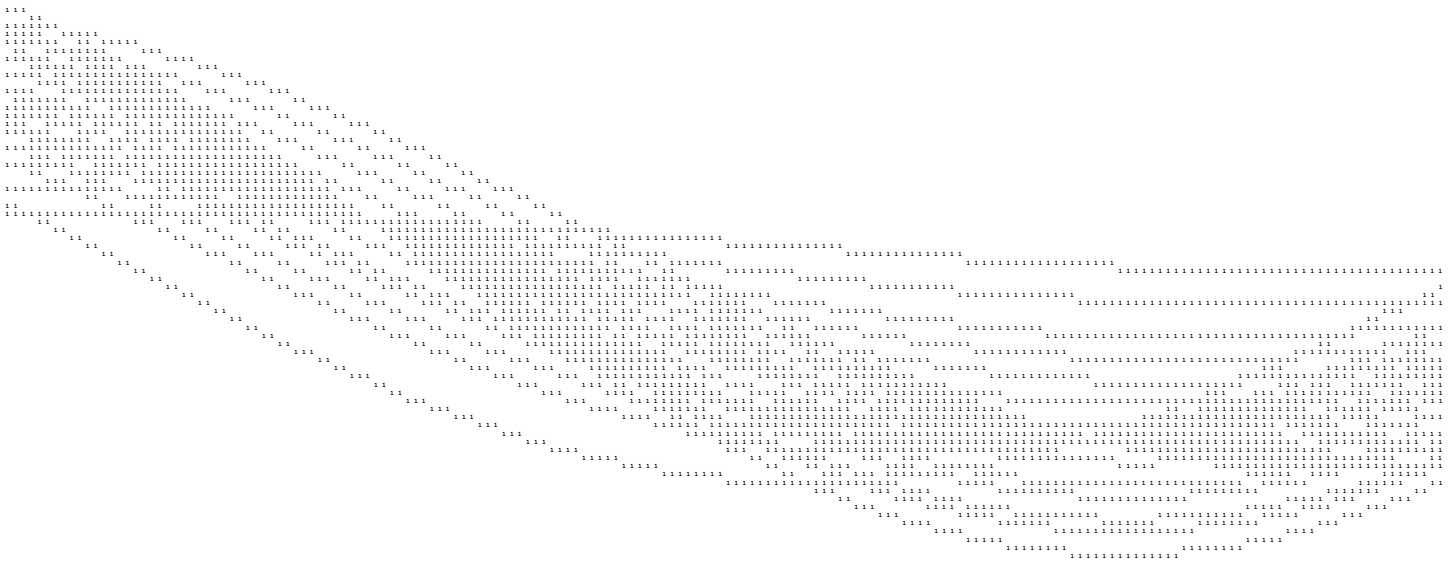
  - Histogram of outFile2

```
90 180 0 14
0  13015
1  2343
2  690
3  102
4  19
5  15
6  7
7  3
8  2
9  0
10  1
11  1
12  1
13  0
14  2
```

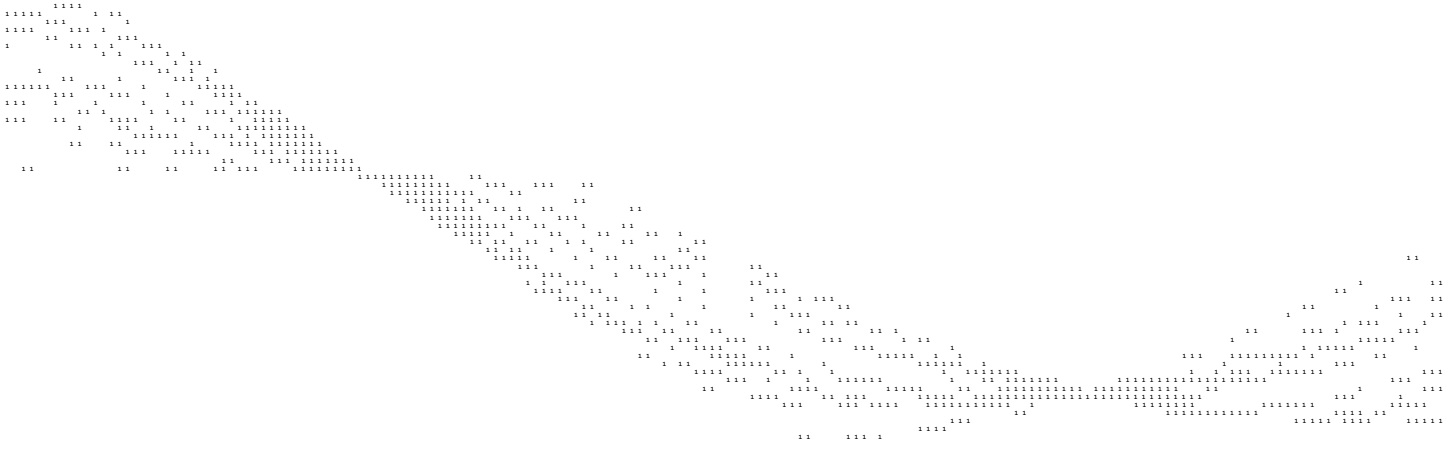
- outFile2



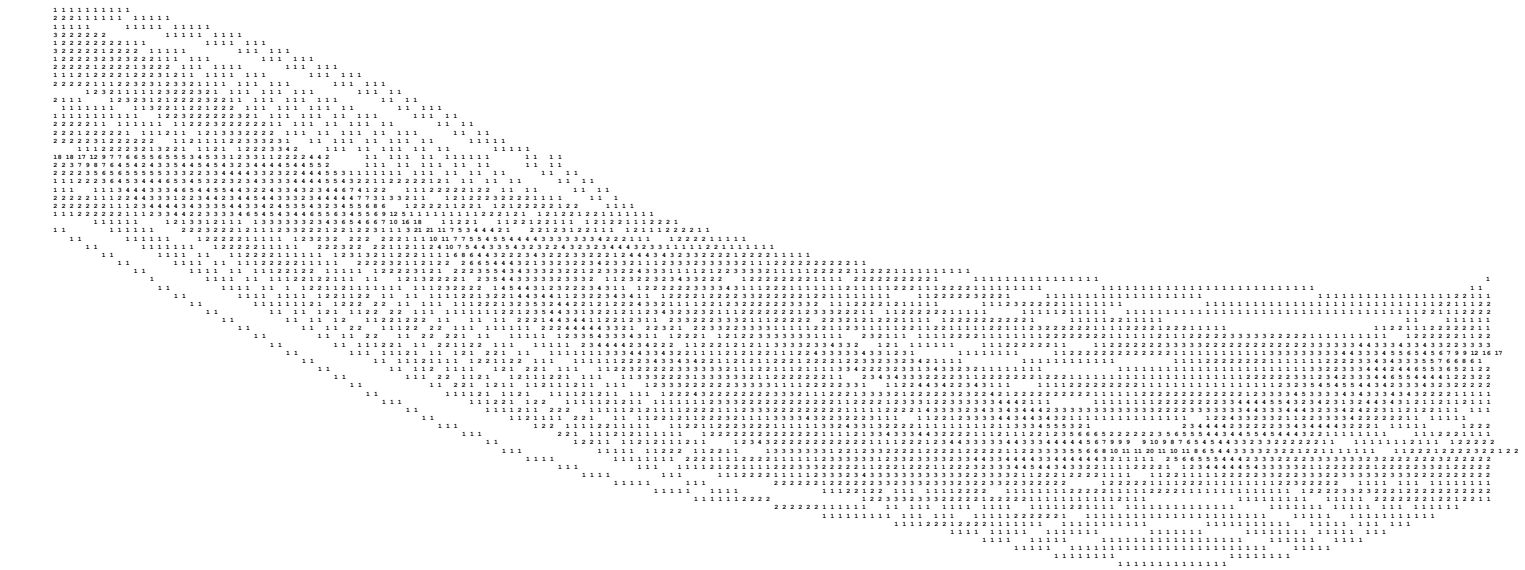
- pretty print of threshold outFile2
  - threshold value is 1



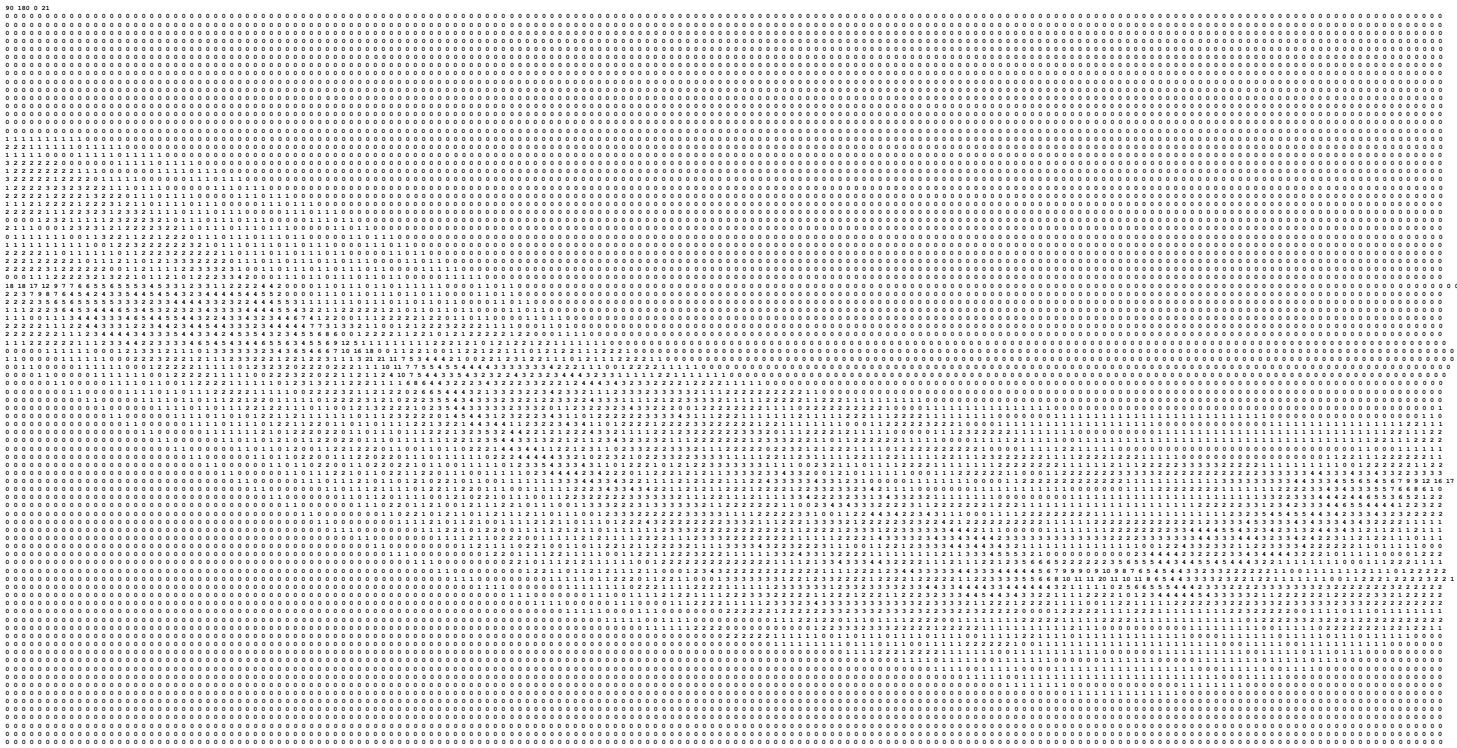
  - threshold value is 2



- HoughTransform_3lines
  - outFile1



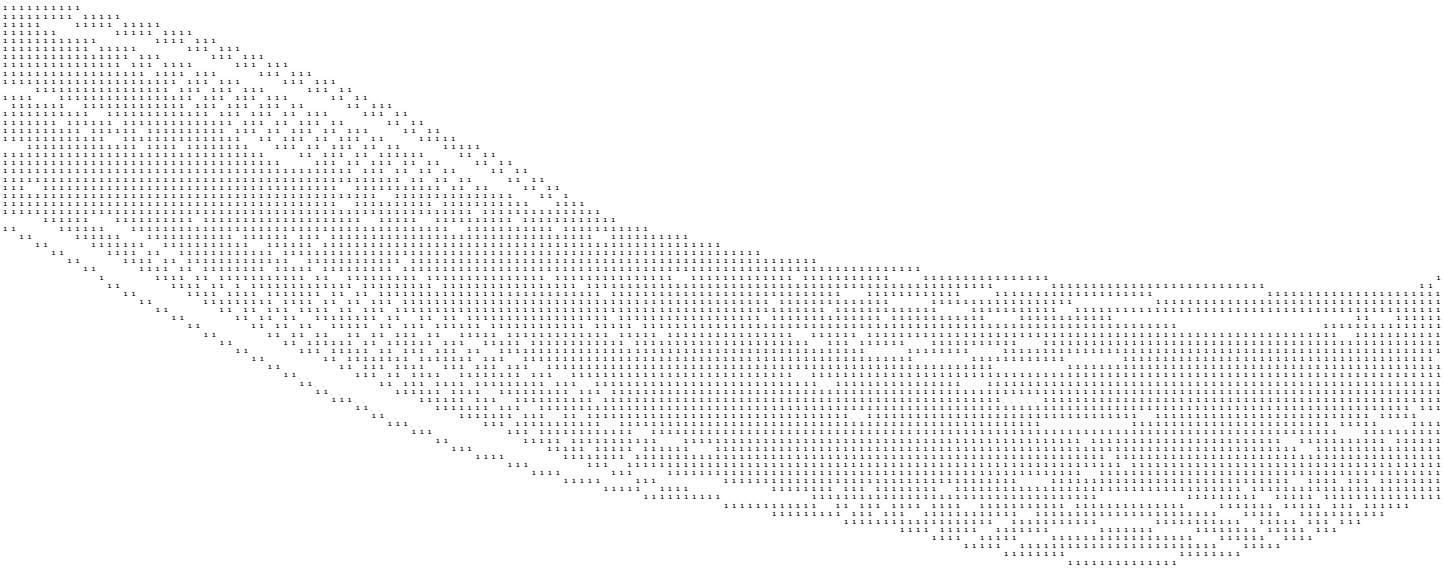
  - outFile2



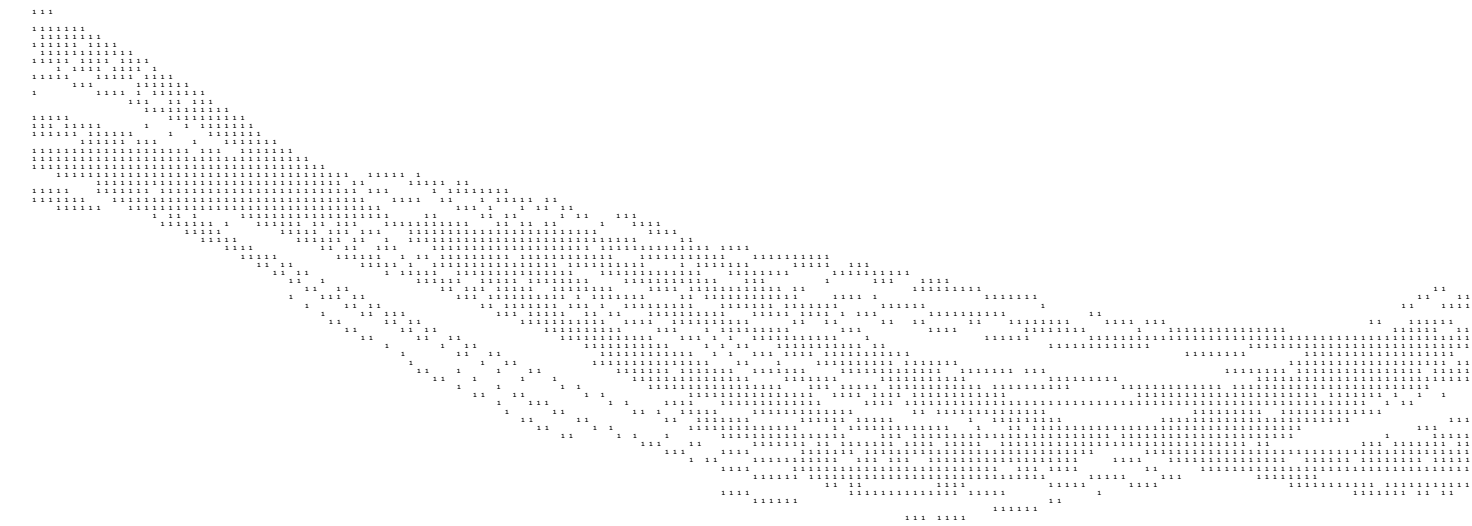
  - Histogram of outFile2

90 180 0 21	14 0
0 11072	15 0
1 2417	16 2
2 1528	17 2
3 684	18 3
4 299	19 0
5 101	20 1
6 42	21 2
7 16	
8 7	
9 10	
10 6	
11 6	
12 3	
13 0	

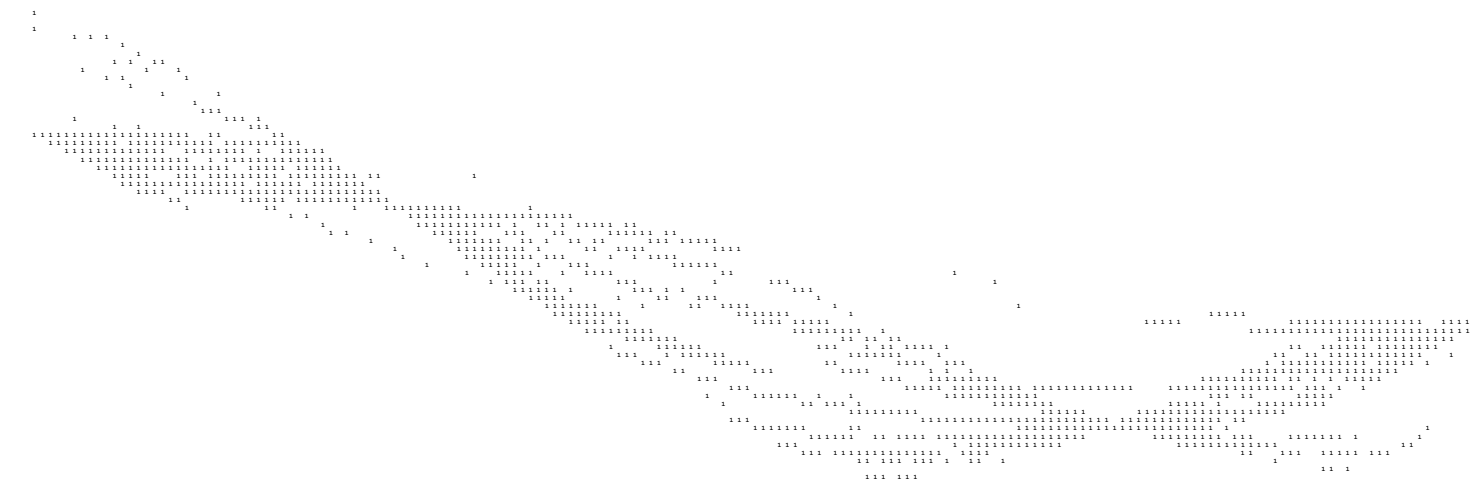
- pretty print of threshold outFile2
  - threshold value is 1



  - threshold value is 2

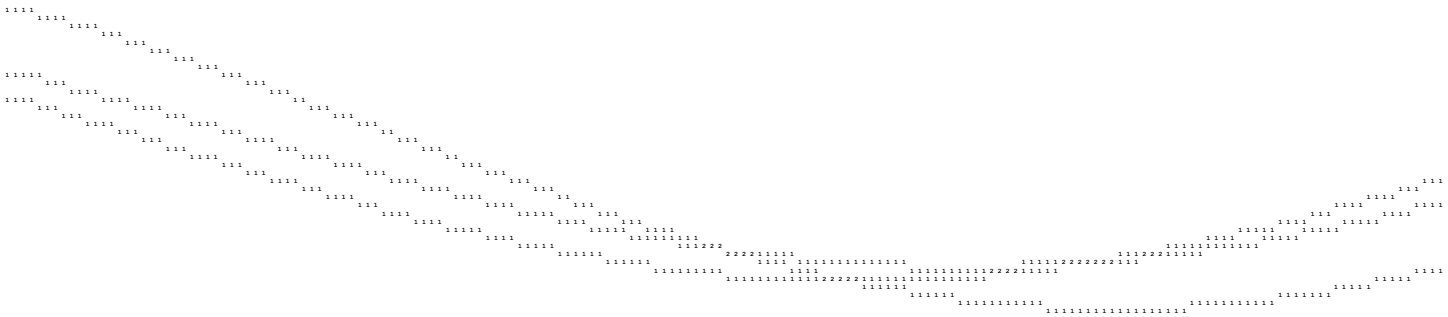


  - threshold value is 3

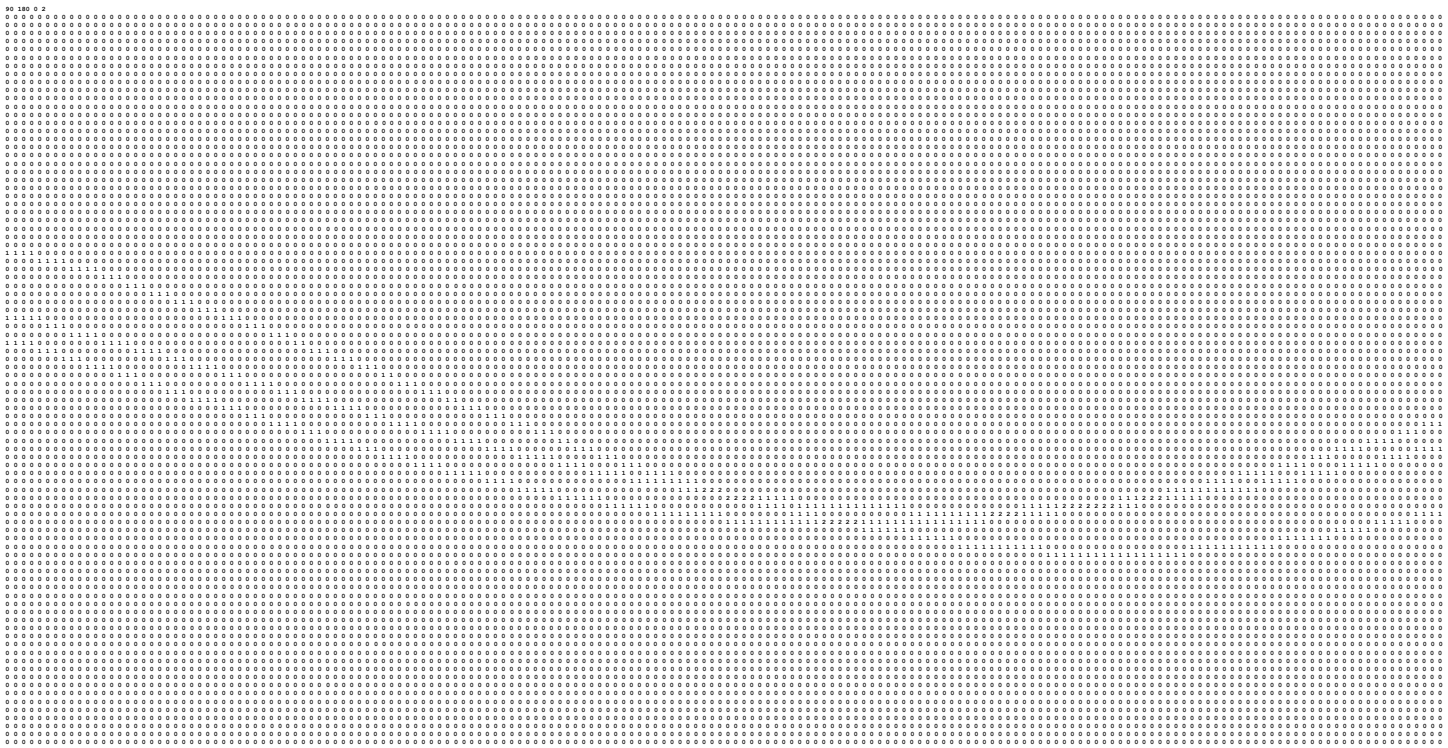


- HoughTransform_2Pts

  - outFile1



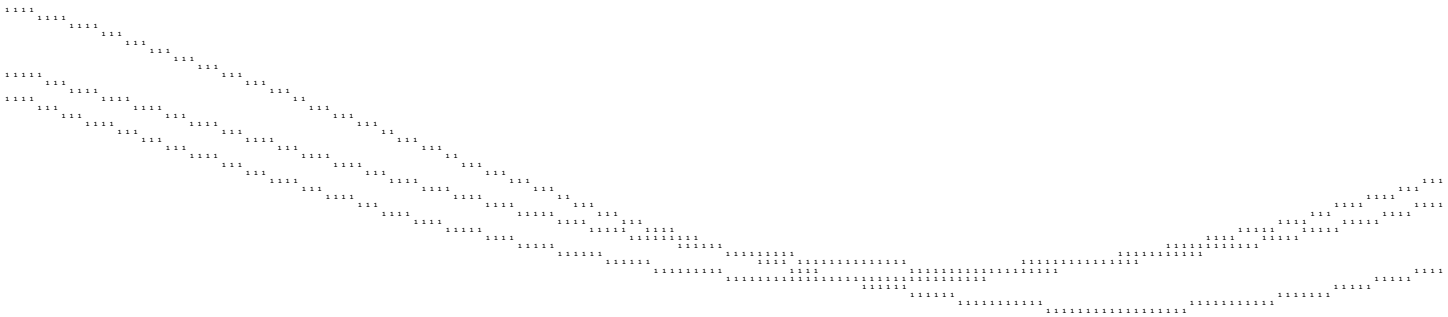
  - outFile2



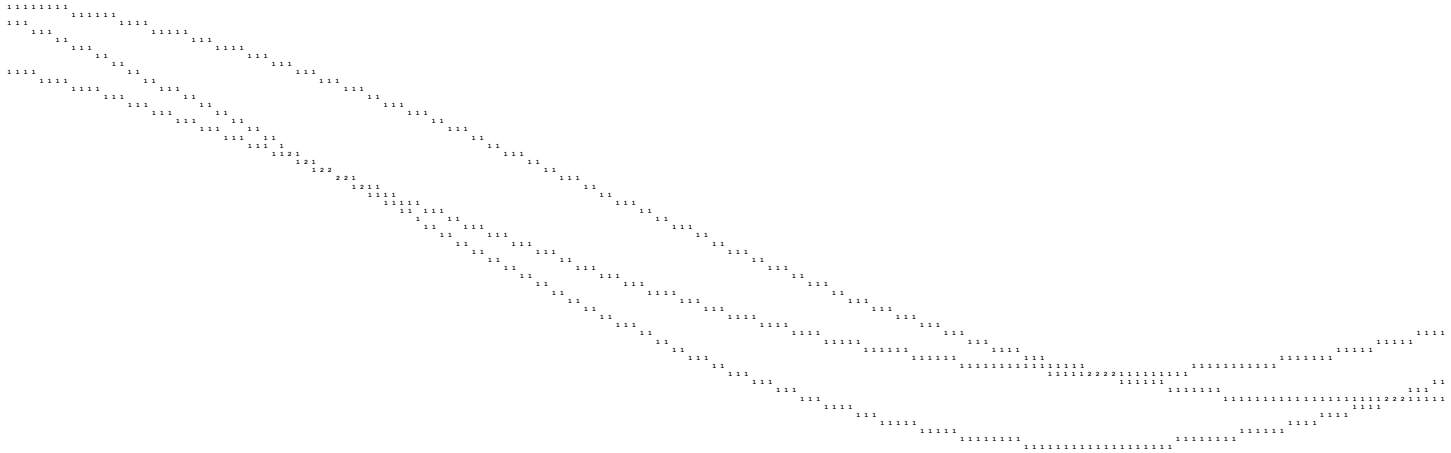
  - Histogram of outFile2

```
90 180 0 2
0  15687
1  488
2  26
```

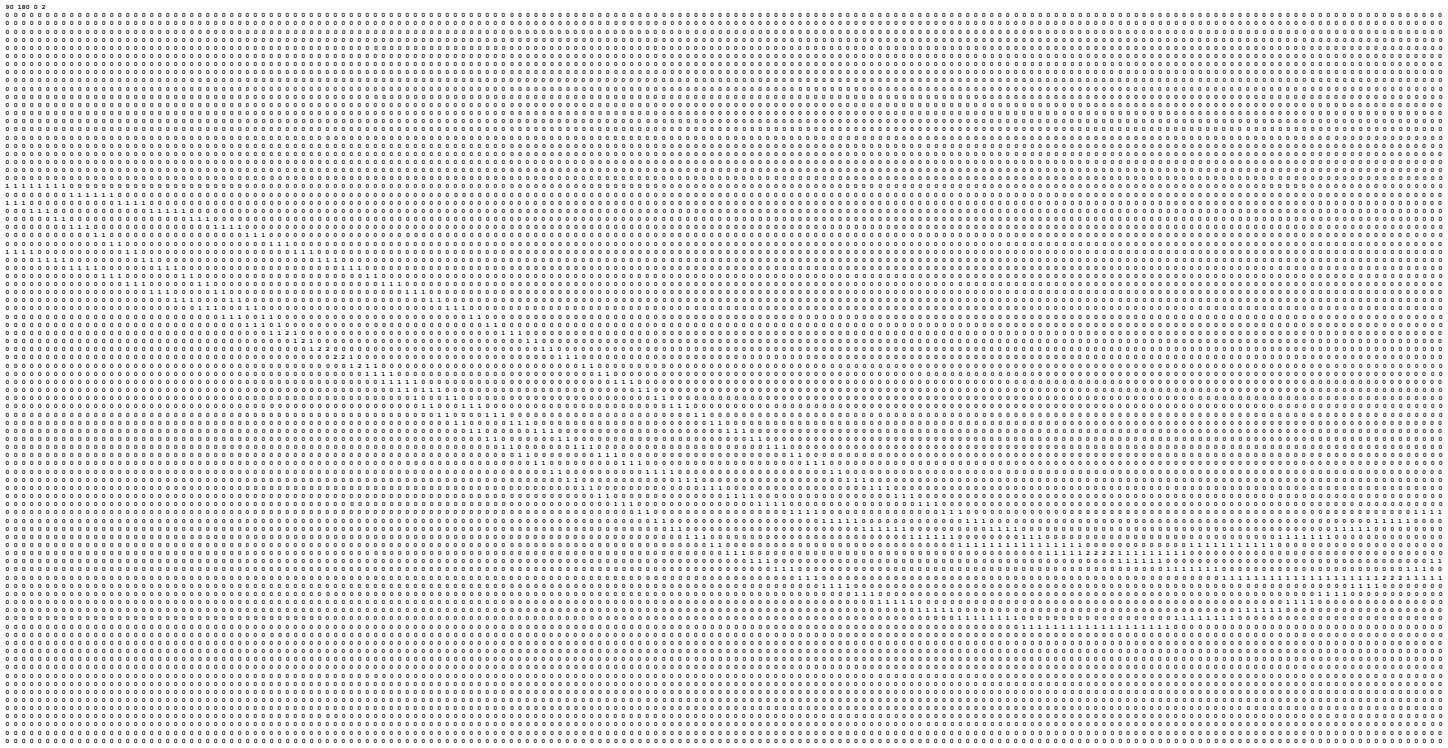
- pretty print of threshold outFile2
  - threshold value is 1



- HoughTransform_3Pts
- outFile1



- outFile2



- Histogram of outFile2

```
90 180 0 2
0  15675
1  512
2  14
```

- pretty print of threshold outFile2
  - threshold value is 1