CV Programming Language: Java

Project #9 Hough Transform

Andres Quintero

Due Date:

Soft copy: 4/12/2020

Hard copy: 4/12/2020

```
****************************************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 (imageAry)

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
```

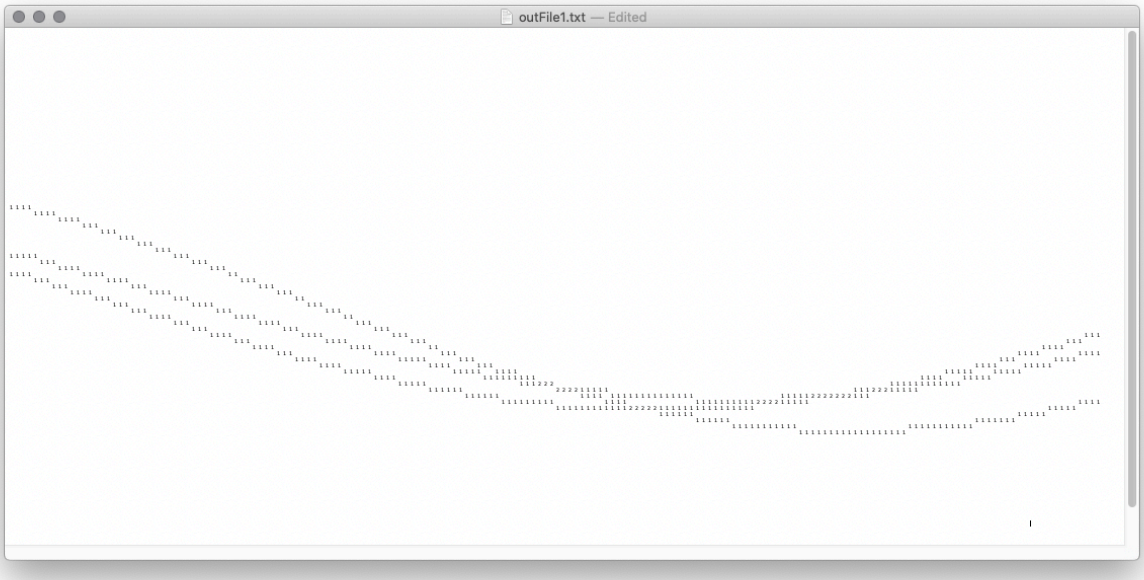
Source code:

```
import java.util.*;
import java.io.*;
import java.lang.Math;

public class Main{
    public static void main(String args[]){
        ImageProcessing image = new ImageProcessing();
        HoughTransform hough = new HoughTransform();

        Scanner inFile = null;
        PrintWriter outFile1 = null;
        PrintWriter outFile2 = null;

        // 0
        try {
            inFile = new Scanner(new File(args[0]));
        } catch (FileNotFoundException err) {
            System.out.println("Error in opening input file from CLI: " + err);
        }

        try {
            outFile1 = new PrintWriter(args[1]);
            outFile2 = new PrintWriter(args[2]);
        } catch (FileNotFoundException err) {
            System.out.println("Error in opening output files from CLI: " + err);
        }

        image.numRows = inFile.nextInt();
        image.numCols = inFile.nextInt();
        image.minVal = inFile.nextInt();
        image.maxVal = inFile.nextInt();

        // dynamically allocating imgAry
        image.imgAry = new int[image.numRows][image.numCols];


        // HOUGH STUFF
        // a^2 + b^2 = c^2
        int diagonal = (int) Math.sqrt( Math.pow(image.numRows, 2) + Math.pow(image.numCols, 2)
);

        hough.HoughAngle = 180;
        hough.HoughDist = 2 * diagonal;
        hough.HoughAry = new int[hough.HoughDist][hough.HoughAngle];

        // 1
        image.loadImage(inFile);

        // 2
        hough.buildHoughSpace(image);

        // 3
        hough.prettyPrint(outFile1);

        // 4
        hough.determineMinMax();

        // 5
        outFile2.println(hough.HoughDist + " " + hough.HoughAngle + " " + hough.HoughMinVal + " "
+ hough.HoughMaxVal);

        // 6
        hough.ary2File(outFile2);

        // 7
        // Closing files
        inFile.close();
        outFile1.close();
        outFile2.close();
    }
```

```
}

public class ImageProcessing {
    int numRows;
    int numCols;
    int minVal;
    int maxVal;

    int[][] imgAry;

    ImageProcessing(){
        numRows = 0;
        numCols = 0;
        minVal = 0;
        maxVal = 0;
    }

    void loadImage(Scanner inFile){
        for(int i = 0; i < numRows; i++){
            for(int j = 0; j < numCols; j++){
                imgAry[i][j] = inFile.nextInt();
            }
        }
    }




}



public class xyCoord{
    int x;
    int y;
}




public class HoughTransform {


    xyCoord point = new xyCoord();
    int angleInDegree;
    double angleInRadians;

    int HoughDist;
    int HoughAngle = 180;
    int HoughMinVal;
    int HoughMaxVal;
    int[][] HoughAry;

    // Functions

    void buildHoughSpace(ImageProcessing image){
        for(int r = 0; r < image.numRows; r++){
            for(int c = 0; c < image.numCols; c++){

                if(image.imgAry[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++;
                    }
                }
            }
        }
    }

    double computeDistance(xyCoord point, double angleInRadians){
        double a = angleInRadians;
        double x = point.x ;
        double y = point.y ;

        double c = Math.atan(y/x);
        double t = a - c - (Math.PI/2);

        double distance = (   Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2))   )   *  Math.cos(t) +
(HoughDist/2);

        // System.out.println(distance);

        return distance;
    }

    void determineMinMax() {
        int max = 0;

        for(int i = 0; i < HoughDist; i++){
            for(int j = 0; j < HoughAngle; j++){
                if(HoughAry[i][j] > max){ max = HoughAry[i][j];}
            }
        }

        HoughMaxVal = max;
        int min = max;

        for(int i = 0; i < HoughDist; i++){
            for(int j = 0; j < HoughAngle; j++){
                if(HoughAry[i][j] < min){ min = HoughAry[i][j];}
            }
        }
        HoughMinVal = min;
    }


    void prettyPrint(PrintWriter outFile){
        for(int i = 0; i < HoughDist; i++){
            for(int j = 0; j < HoughAngle; j++){
                if(HoughAry[i][j] > 0) {outFile.print(HoughAry[i][j] + " ");}
                else {outFile.print("  ");}
            }
            outFile.println();
        }
    }

    void ary2File(PrintWriter outFile){
        for(int i = 0; i < HoughDist; i++){
            for(int j = 0; j < HoughAngle; j++){
                if(HoughAry[i][j] > 0) {outFile.print(HoughAry[i][j] + " ");}
                else {outFile.print("0 ");}
            }
            outFile.println();
        }
    }

}
```

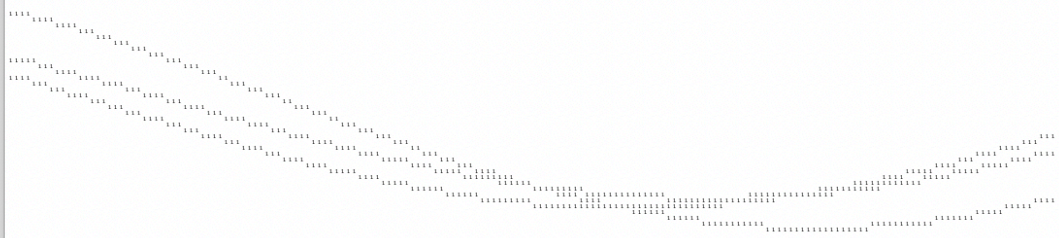
Outputs for 2pts



histogram.txt

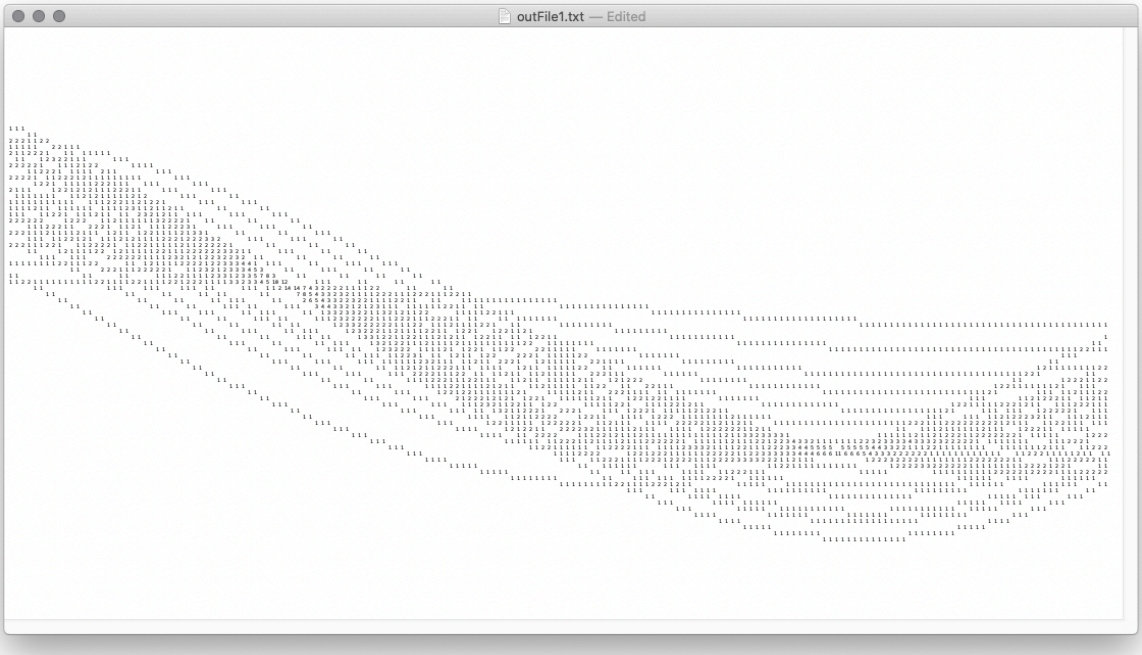
```
90 180 0 2
0  (15686):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1  (488):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2  (26):++++++++++++++++++++++++++++
```

prettyThreshold.txt — Edited



END OF 2pt OUTPUTS

Outputs for 2lines



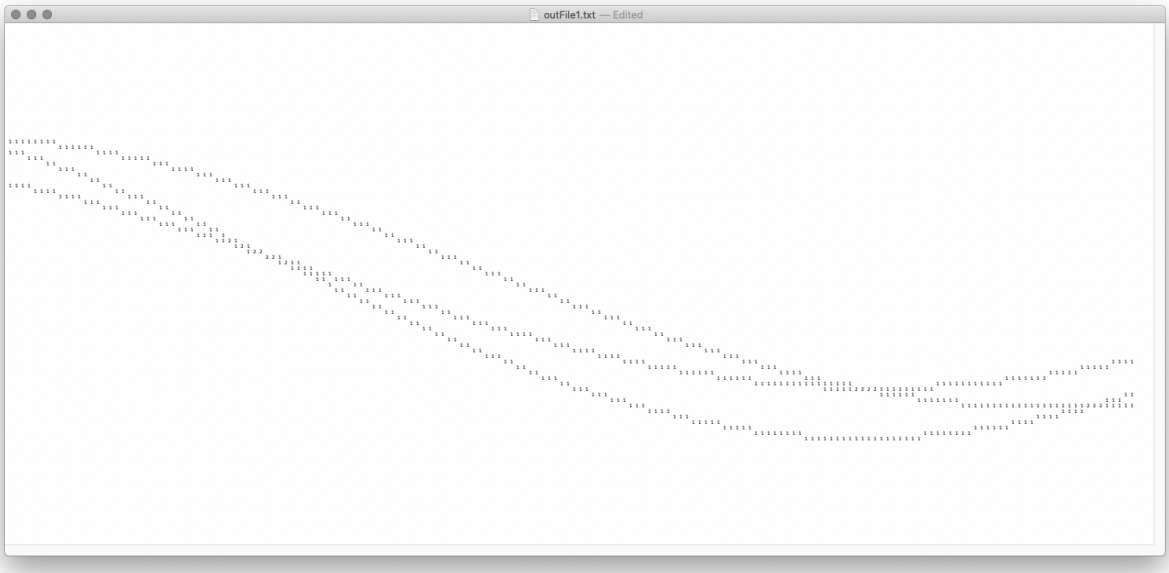
histogram.txt

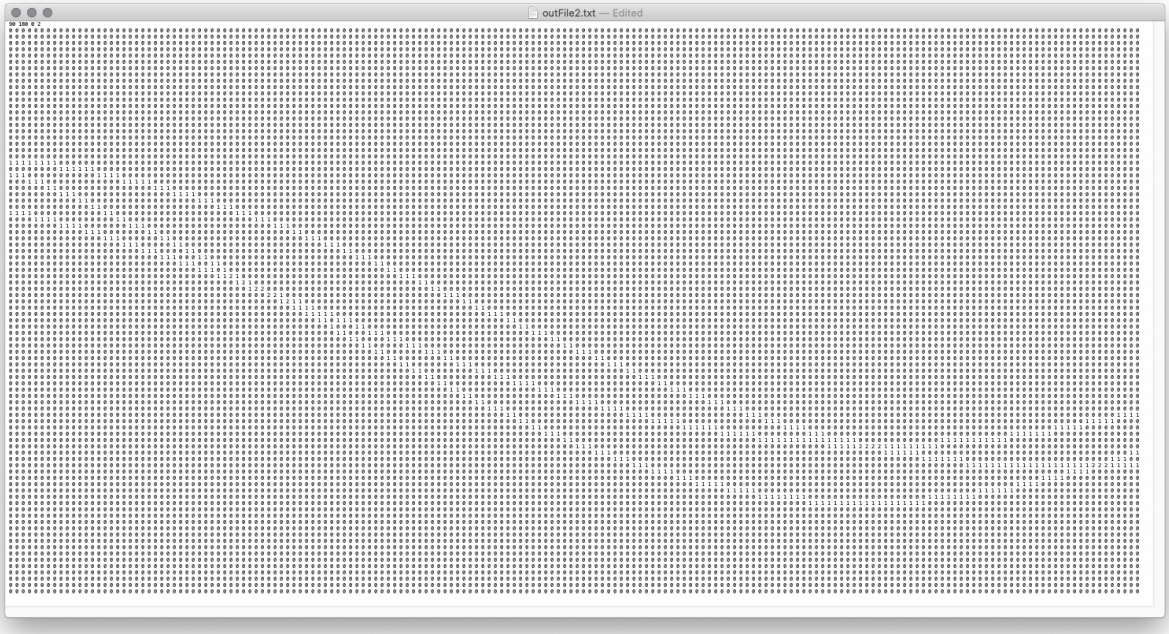
```
90 180 0 14
0  (13014):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1  (2343):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2  (690):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3  (102):++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
4  (19):++++++++++++++++++++
5  (15):++++++++++++++++
6  (7):+++++++
7  (3):+++
8  (2):++
9  (0):
10  (1):+
11  (1):+
12  (1):+
13  (0):
14  (2):++
```

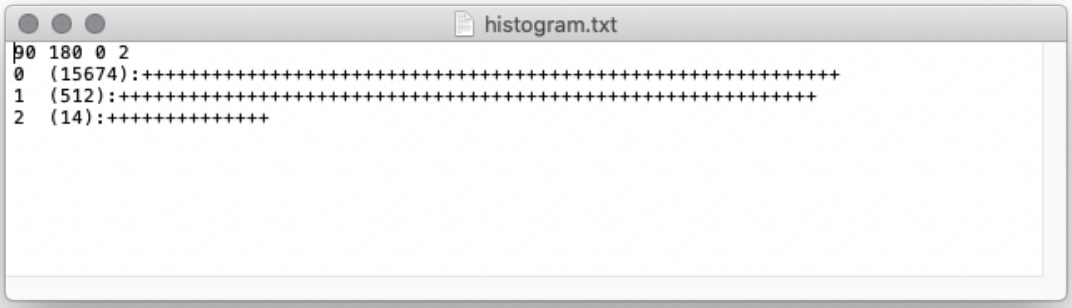
prettyThreshold.txt — Edited

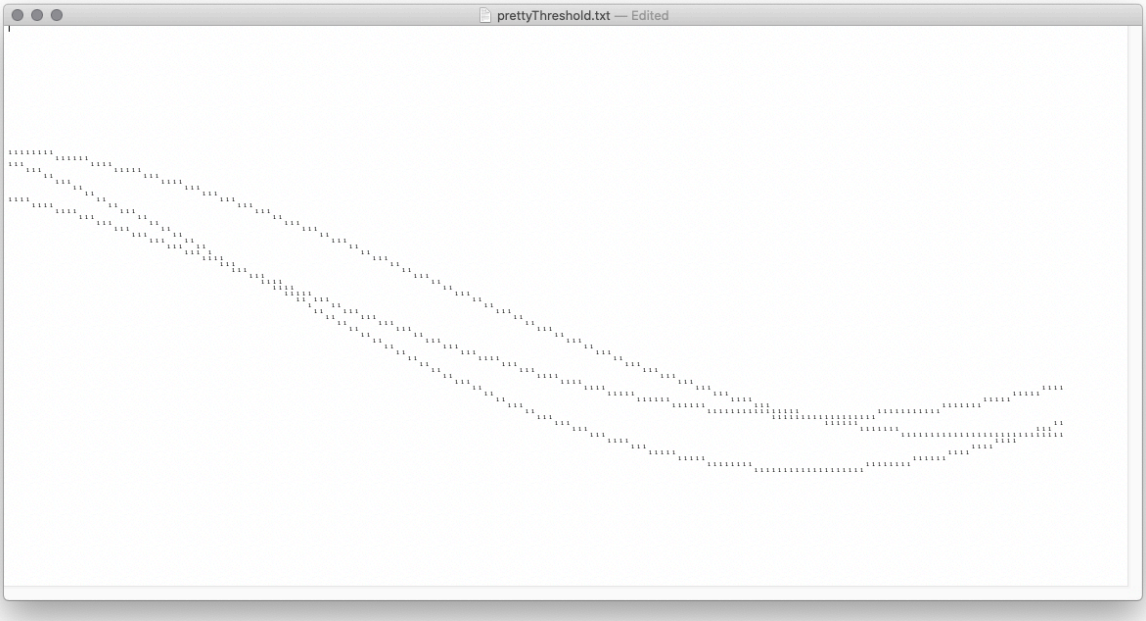
END OF 2lines OUTPUTS

Outputs for 3pts



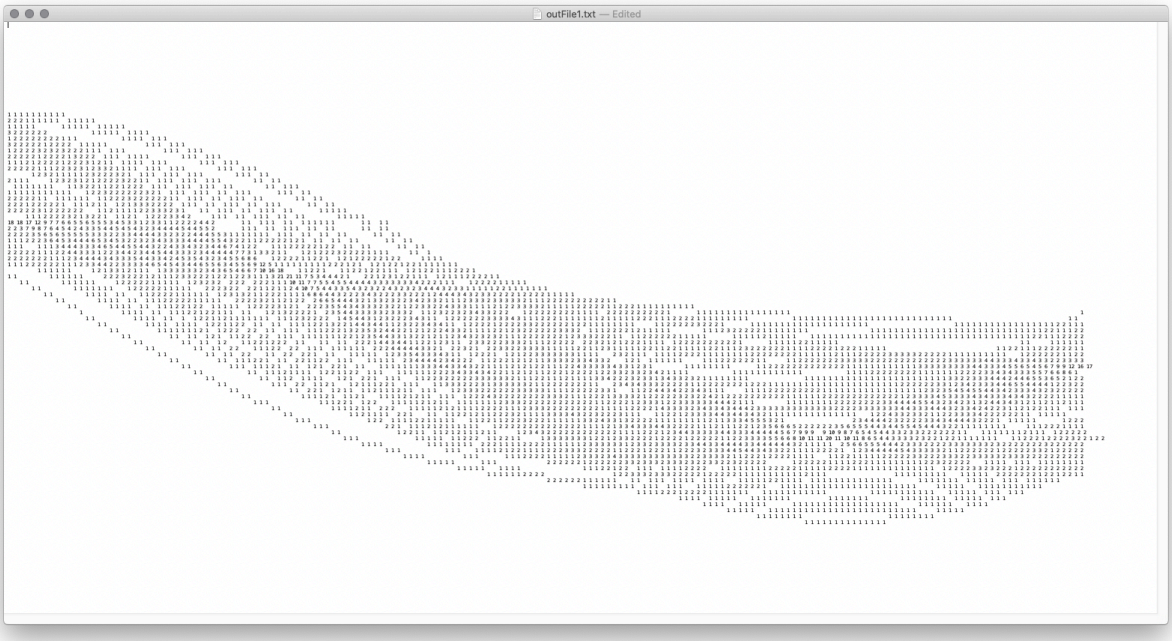


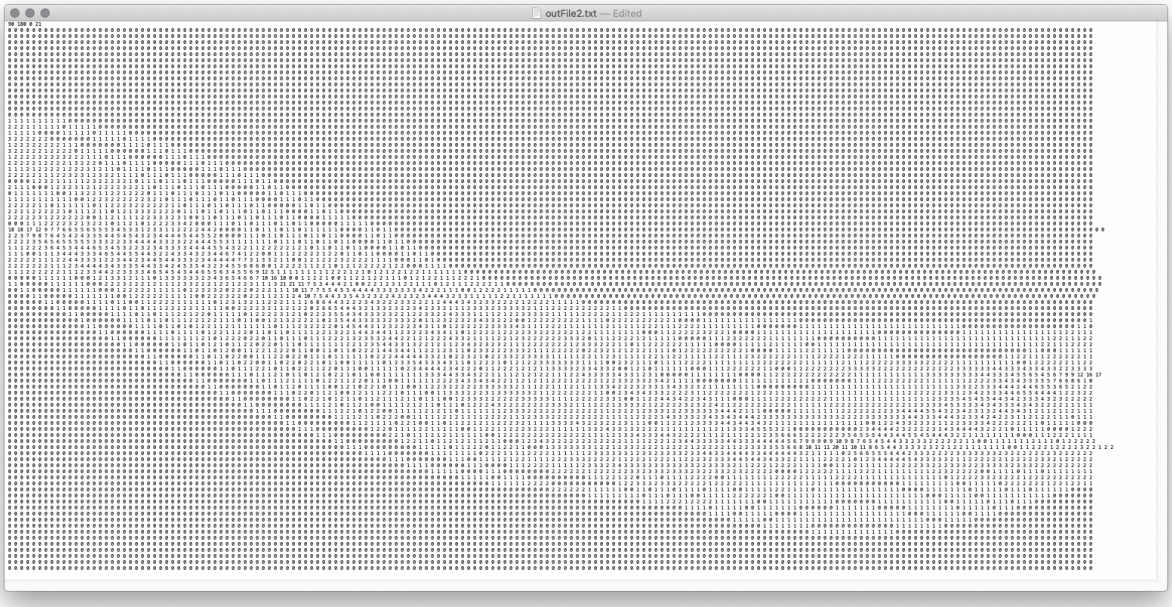


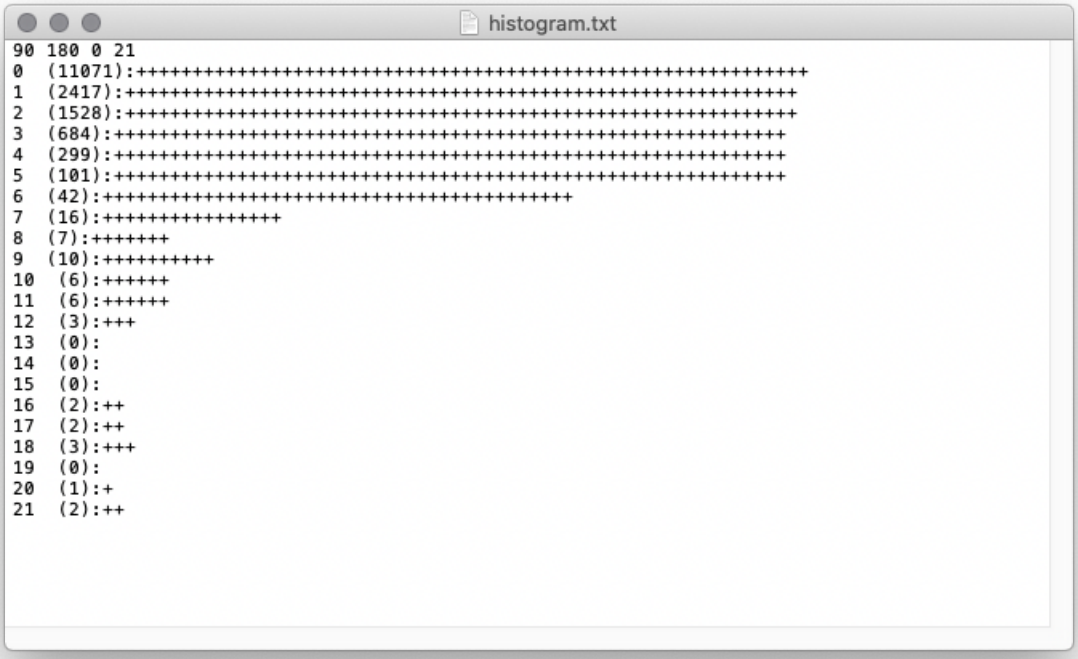


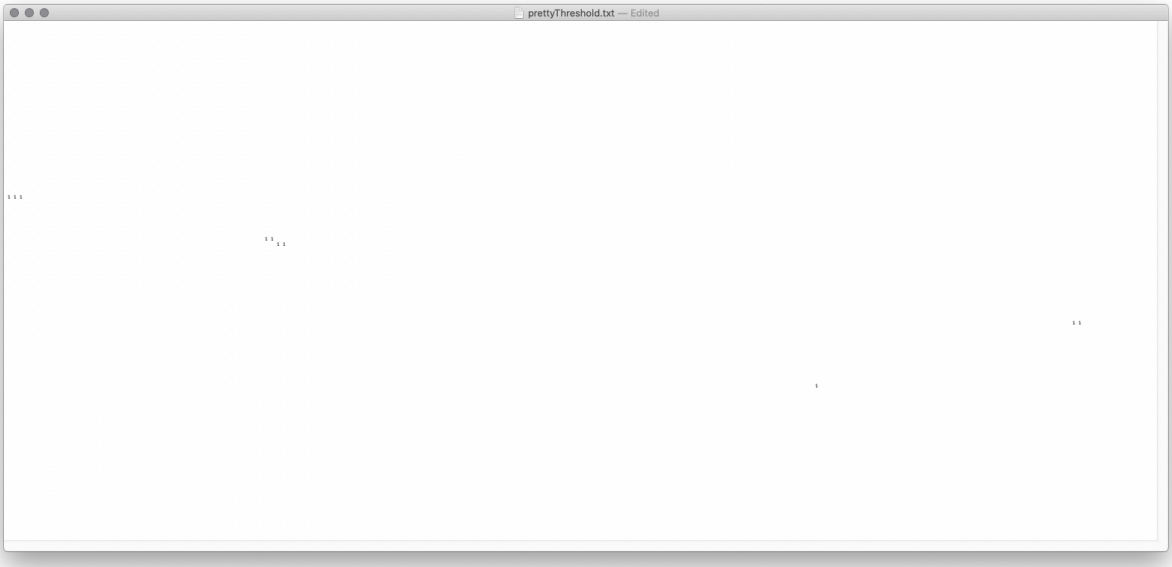
END OF 3pts OUTPUTS

Outputs for 3lines









END OF 3lines OUTPUTS