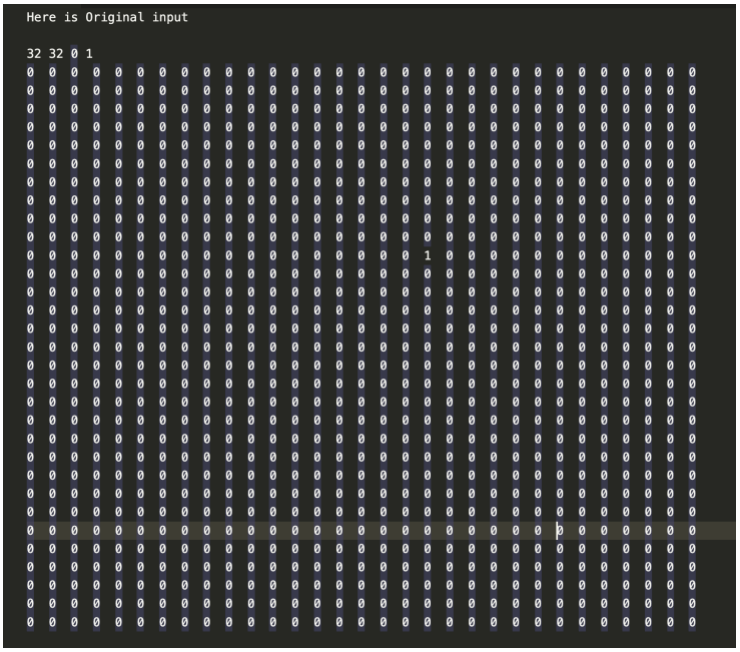
Jonathan Mathew
CV Project 5 ImageCompression
Due 04/19/23

Algo Steps:

Step 0:
inFile, outFile1-->open from args []
numRows, numCols, minVal, maxVal-->read from
inFile HoughAngle --> 180
HoughDist-->2 * (the diagonal of the input image)
imgAry-->dynamically allocate
CartesianHoughAry-->dynamically allocate and initialize to zero
PolarHoughAry-->dynamically allocate and initialize to zero
offSet --> // See your lecture note.
Step 1: loadImage (inFile, imgAry) prettyPrint (imgAry, outFile1)
Step 2: buildHoughSpace (...)
Step 3: prettyPrint (CartesianHoughAry, outFile1) // with caption indicate it is Cartesian Hough space
prettyPrint (PolarHoughAry, outFile1) // with caption indicate it is Polar Hough space Step 4: close all files

img1pt-

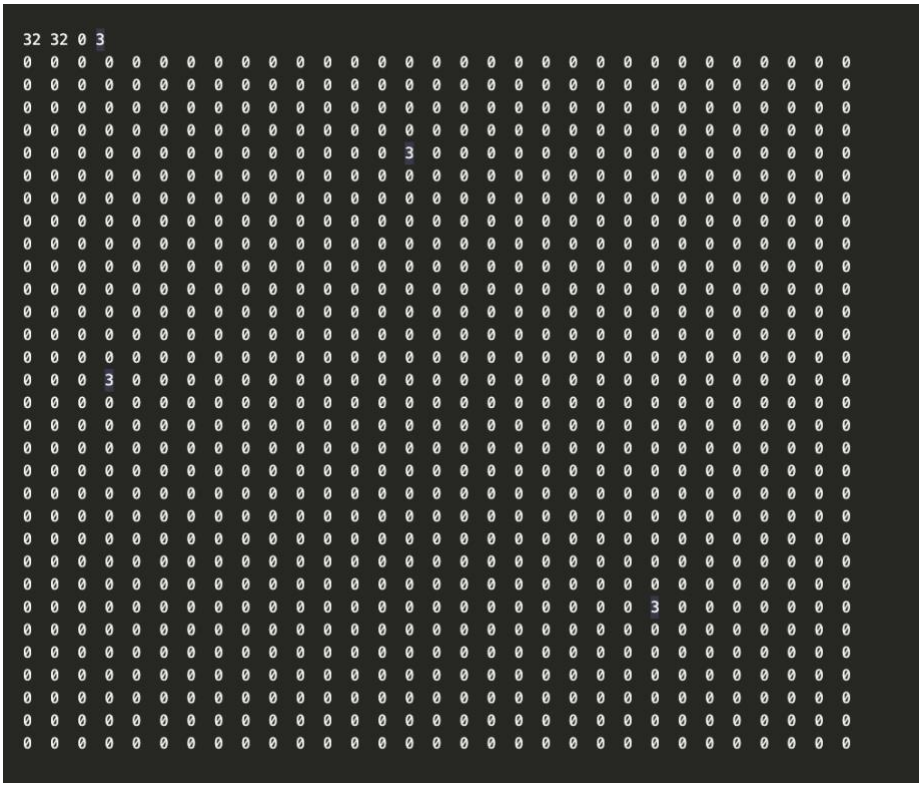
Here is Original input



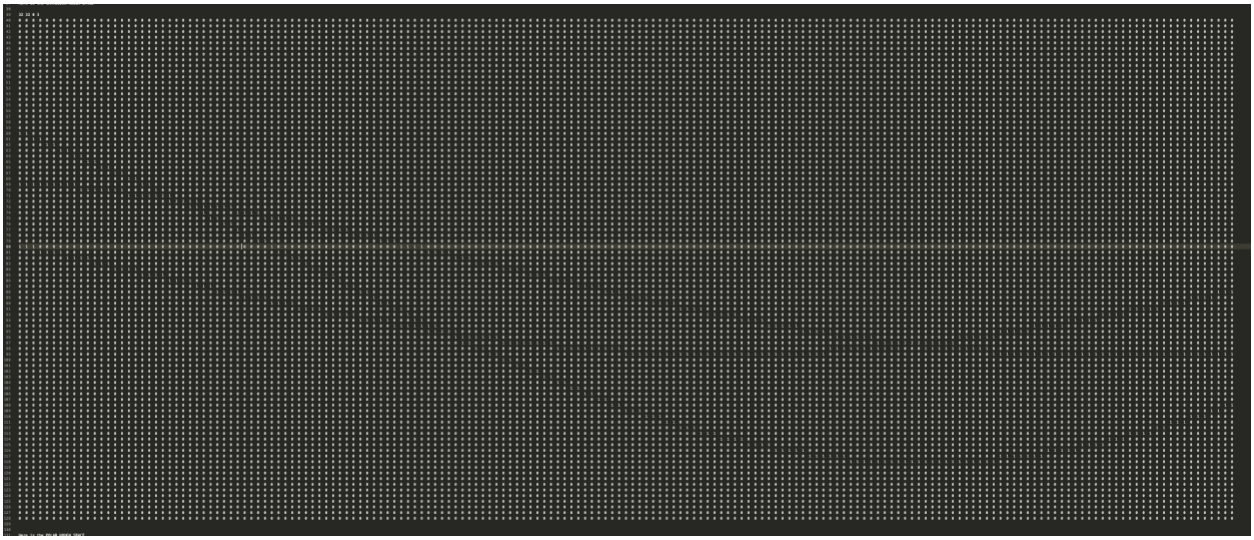
Cartersian

Polar

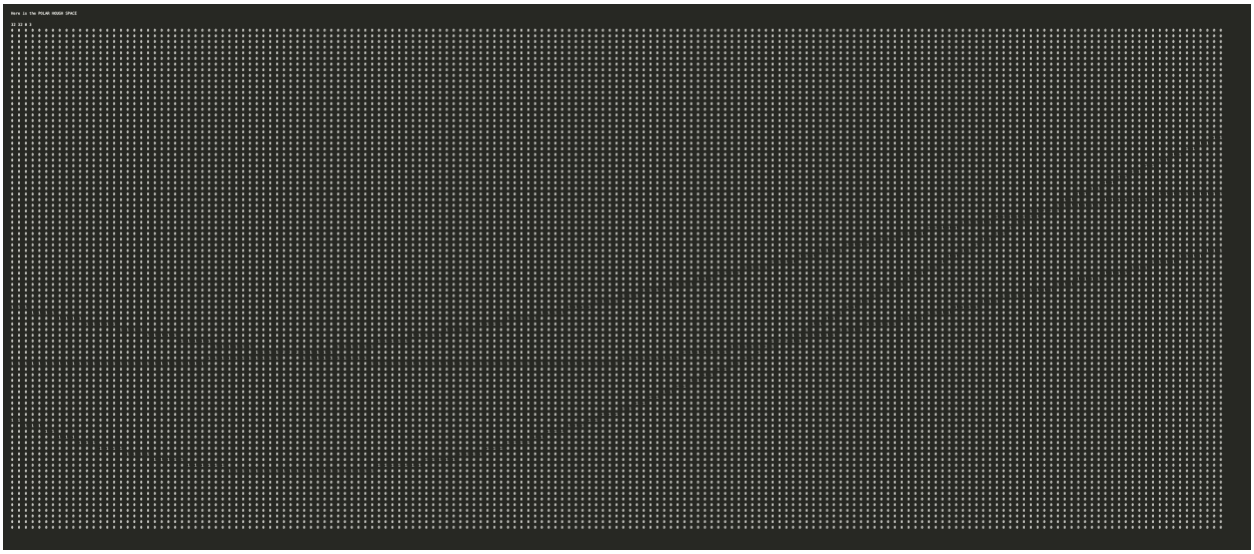
img3pt-



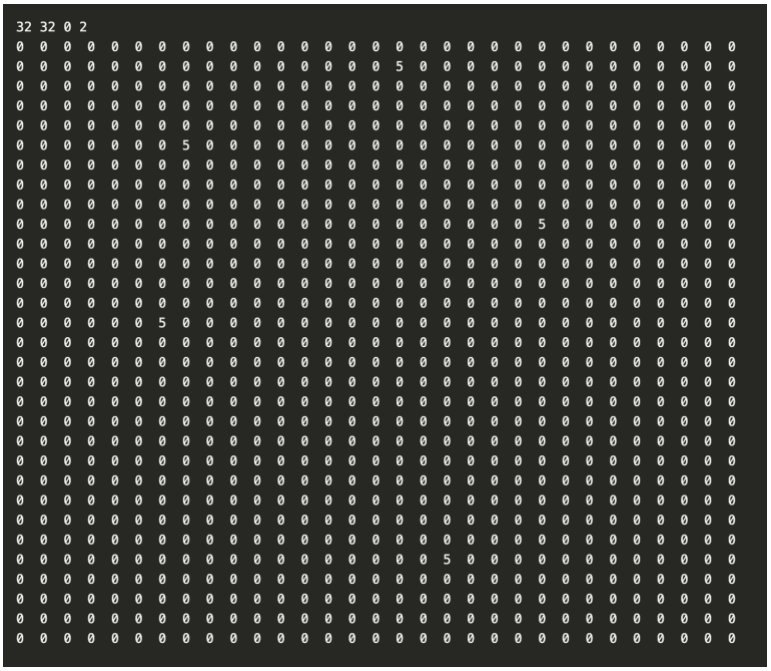
Cartersian



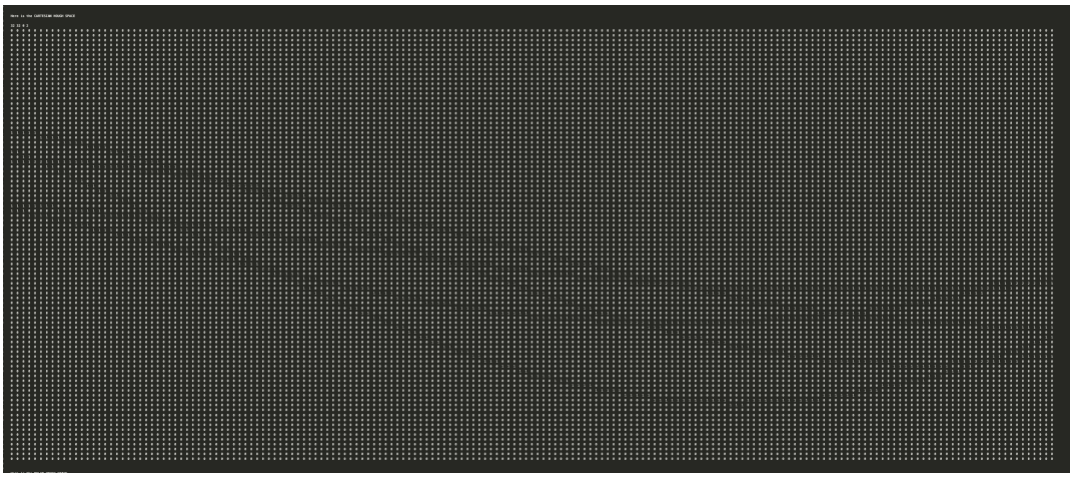
Polar



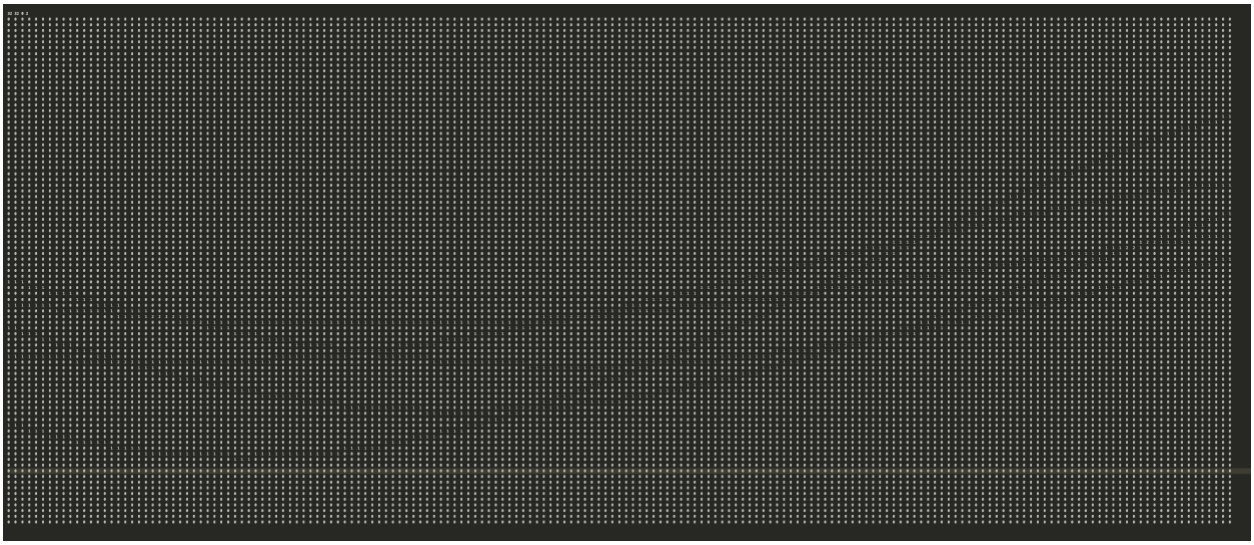
img5pt-



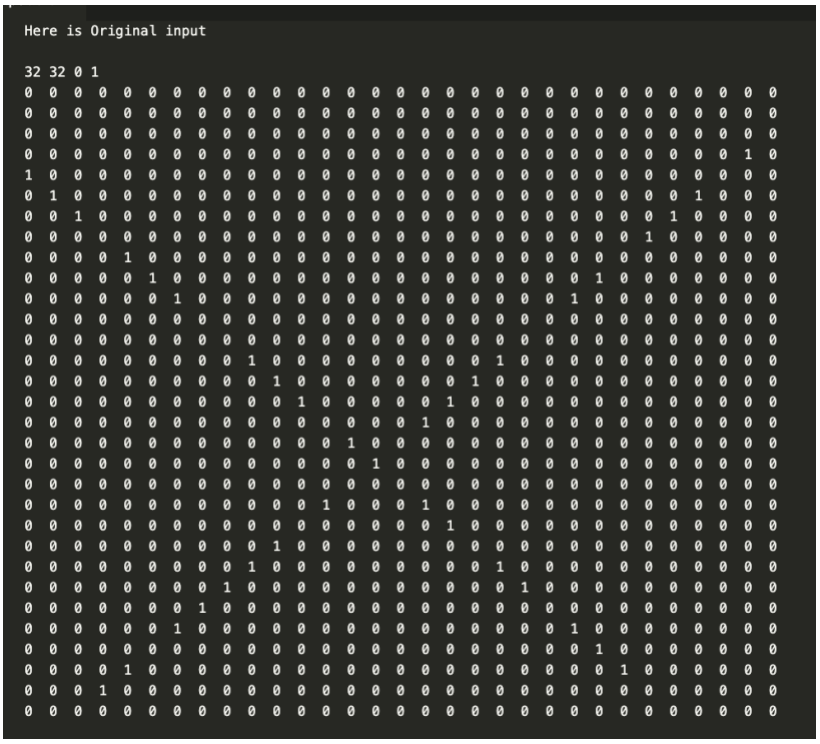
Cartersian



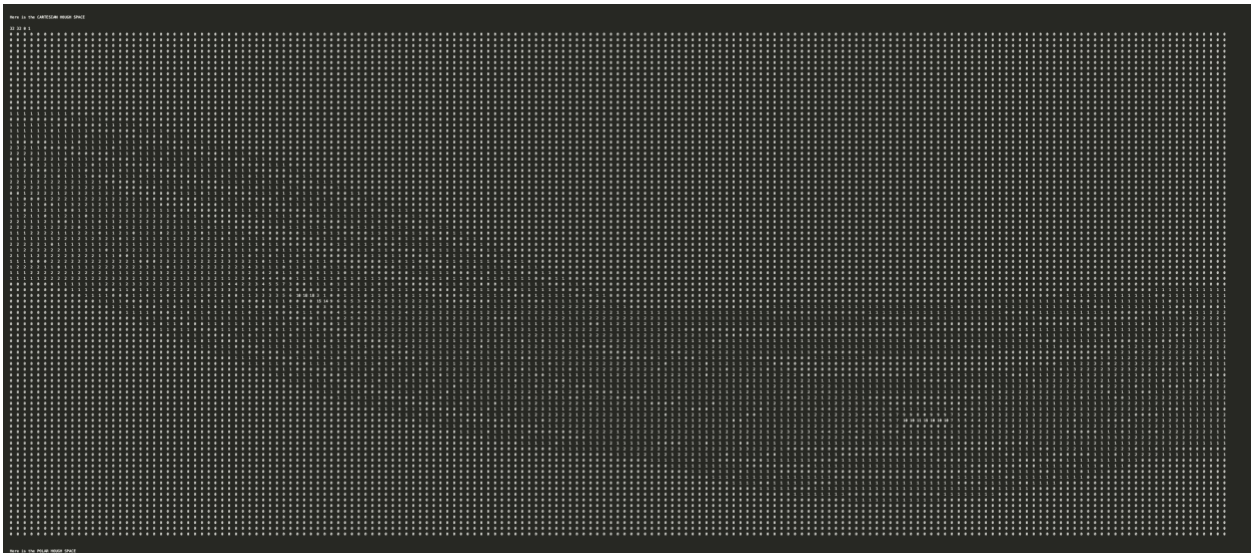
Polar



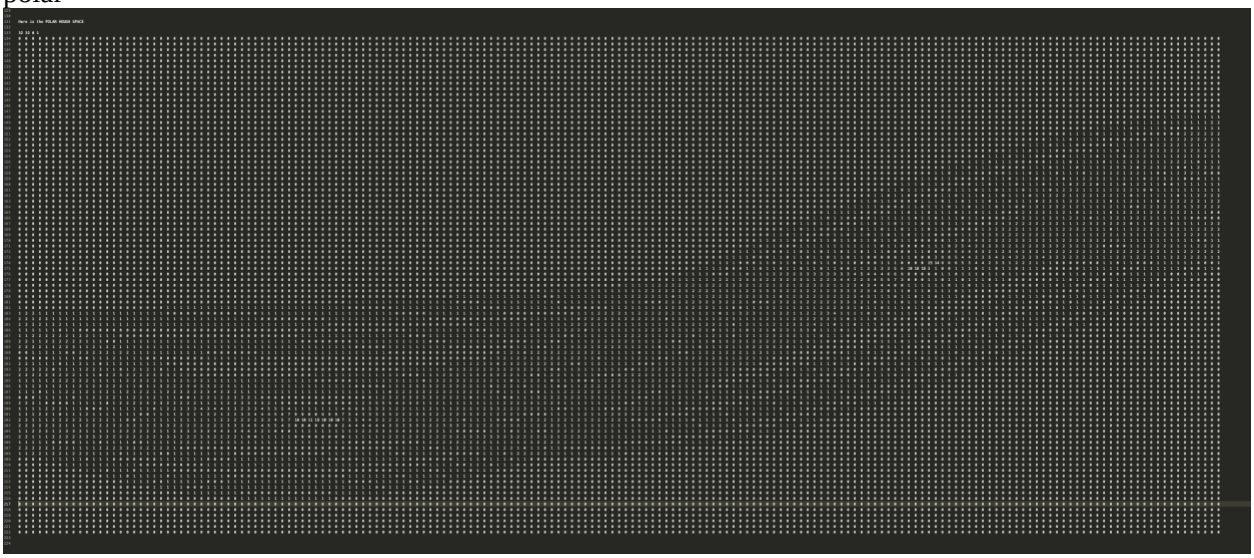
img2lines-



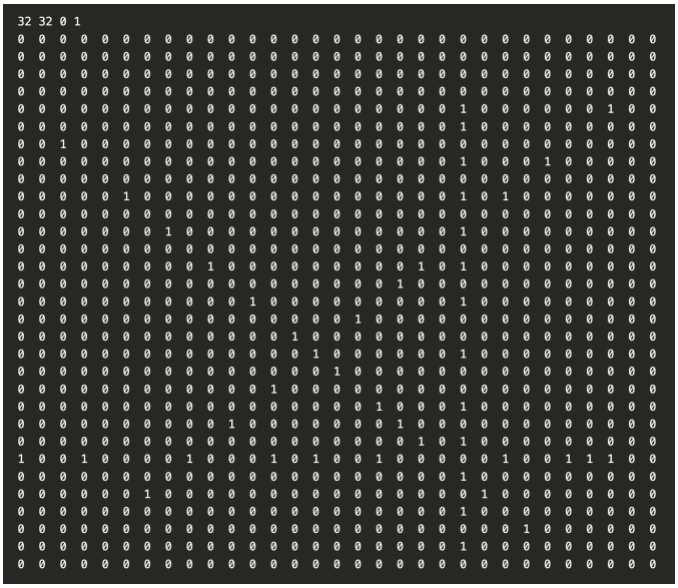
Cartersian



polar

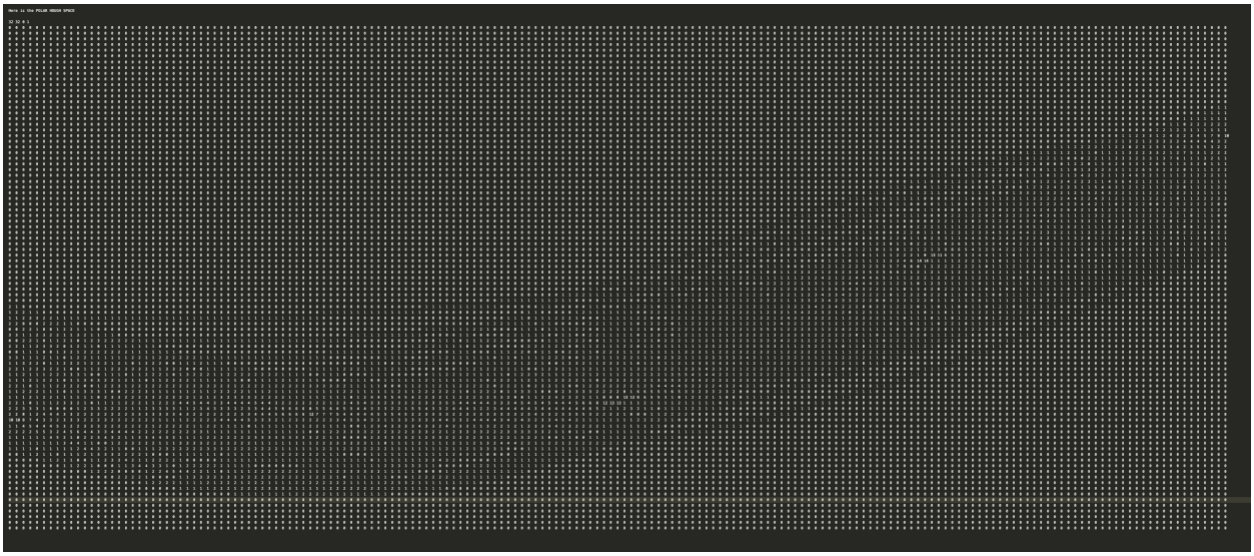


img5lines-



Cartersian

Polar



src

Main.java

```
import java.io.*;

public class Main {

    public static void main(String[] args) throws IOException {

        File infile = new File(args[0]),
                outFile = new File(args[1]);

        FileReader in = new FileReader(infile);
        FileWriter out = new FileWriter(outFile);

        BufferedReader br = new BufferedReader(in);
        String line = br.readLine();
        String [] header = line.split(" ");
        int numRow = Integer.parseInt(header[0]),
                numCol = Integer.parseInt(header[1]),
                minVal = Integer.parseInt(header[2]),
                maxVal  = Integer.parseInt(header[3]);

        HoughTransform proj = new HoughTransform(numRow, numCol, minVal, maxVal);


        in.close();
        in = new FileReader(infile);
        proj.loadImg(in);
        out.write("Here is Original input\n\n");
        proj.reformatPrettyPrint(proj.imgArr, out);
        proj.buildHSpace();

        out.write("\nHere is the CARTESIAN HOUGH SPACE\n\n");
        proj.reformatPrettyPrint(proj.cartHoughArr, out);
        out.write("\nHere is the POLAR HOUGH SPACE\n\n");
        proj.reformatPrettyPrint(proj.polarHoughArr, out);


        in.close();
        out.close();
    }



}
```

HoughTransform.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class HoughTransform {

    public int numRows,
            numCols,
            minVal,
            maxVal,
            HoughDist,
```

```
            HoughAngle,
            angleInDegree,
            offSet;

    public int[][] imgArr,
            cartHoughArr,
            polarHoughArr;

    public double angleInRadians;


    public HoughTransform(int row, int col, int min, int max){
        numRows = row;
        numCols = col;
        minVal = min;
        maxVal = max;
        int pyth = (int) Math.sqrt(numCols * numCols + numRows *numRows);
        HoughDist = 2 *pyth;
        HoughAngle = 180;
        offSet = (int) Math.sqrt(numRows*numRows + numCols*numCols);
        imgArr = new int[numRows][numCols];
        cartHoughArr = new int[HoughDist][HoughAngle];
        polarHoughArr = new int[HoughDist][HoughAngle];

    }


    public void loadImg(FileReader in) throws IOException {

        int count = 0;
        BufferedReader br = new BufferedReader(in);
        String line = br.readLine();
        while ((line= br.readLine()) != null){
            String[] data = line.split(" ");
            for (int j = 0; j<numCols; j++){
                imgArr[count][j] = Integer.parseInt(data[j]);
            }
            count++;
        }

    }

    public void printArr(FileWriter out) throws IOException {

        for (int r = 0; r<numRows; r++){
            for(int c = 0; c<numCols; c++){
                out.write(imgArr[r][c] + " ");
            }
            out.write("\n");
        }
    }

    public void buildHSpace(){

        for (int x = 0; x<numRows; x++){
            for(int y = 0; y<numCols; y++){

                if (imgArr[x][y]>0){
                    computeSinusoid(x,y);
                }
            }
        }
```

```
    }

    public void computeSinusoid(int x, int y) {

        int angInDegrees = 0;

        while (angInDegrees <=179) {
            double angleInRadians = (double) (angInDegrees * (Math.PI/180));
            double dist = cartDist(x, y, angleInRadians);
            int distInt = (int) dist;
            cartHoughArr[distInt][angInDegrees]++;
            dist = polDist(x, y, angleInRadians);
            distInt = (int) dist;
            polarHoughArr[distInt][angInDegrees]++;
            angInDegrees++;
        }

    }

    private double polDist(int x, int y, double angleInRadians) {
        return x*Math.cos(angleInRadians) + y*Math.sin(angleInRadians) + offSet;
    }

    public double cartDist(int x, int y, double angleInRadians) {
        double t = angleInRadians - Math.atan((double)y/(double)x) - Math.PI/2;
        return Math.sqrt(x*x + y*y) * Math.cos(t) + offSet;
    }


    public void reformatPrettyPrint(int[][] arr, FileWriter out1) throws IOException {

        out1.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");
        String str = Integer.toString(99);
        int width = str.length();

        int ww;
        int r = 1;

        while (r < arr.length) {
            int c = 1;
            while (c < arr[0].length) {

                out1.write(arr[r][c] + "");
                str = Integer.toString(arr[r][c]);
                ww = str.length();
                while (ww <= width) {
                    out1.write(" ");
                    ww++;
                }
                c++;
            }
            r++;
            out1.write("\n");

        }
        out1.write("\n");

    }

}
```