CV
Project 2: Average, Median and Gaussian Filtering
Java

Adrian Noa
Due 9/16/2022

My Algorithms:

loadImage():
To load the input image into a 2D array
Read in from input image file

```
1. For i=1<rows+1:
2. For j=1<cols+1:
    a. inputArray[i][j] ← inputFile Read
```

mirrorFraming():
The purpose of this algorithm is to fill the empty array cells that are padding the outer walls of the input image
This can be used after having loaded the image into a 2D array(row+2, col+2) **mirrorAry** by mirroring the original input image edge pixels.

```
1. Corner Pixels:
        1.1 mirrorAry[0][0] ← mirrorAry[2][2] // top left
        1.2 mirrorAry[0][x] ←  mirrorAry[2][x-2] // top right
        1.3 mirrorAry[x][0] ←  mirrorAry[x-2][2] // bottom left
        1.4 mirrorAry[x][x] ←  mirrorAry[x-2][x-2] // bottom left
2. Edge Pixels:
    1. Horizontal
            For i=1 < columns+1
                    mirrorAry[0][i] ← mirrorAry[2][i]
                    mirrorAry[rows+1][i] ← mirrorAry[rows-1][i]
    2. Vertical
            For i =1< rows+1
                    mirrorAry[i][0] ← mirrorAry[i][2]
                    mirrorAry[i][cols+1] ← mirrorAry[i][cols-1]
```

loadMask():
Same as loadImage, read in from mask file, rows and columns are the size of the mask. 3x3

loadMask1DAry():

This method creates a 1D array from mask2DArray and calculates the total weight of the mask

```
1. Maskweight ← 0
2. For i=0 <  3:
3. For j=0 < 3:
    a. mask1D[ (i*3) + j ] ←  mask2DArray[i][j];
    b. Maskweight ←  mask2DArray[i][j];
```

loadNeighbor1DAry(r,c):

Load the target pixel in location (r,c) and 8 neighbors into 1D array

```
1. For i=0 < 3:
2. For j=0 < 3:
    a. neighborArray[i*3 +j] = mirrorAry[r-1+i][c-1+j]
```

computeAvg():

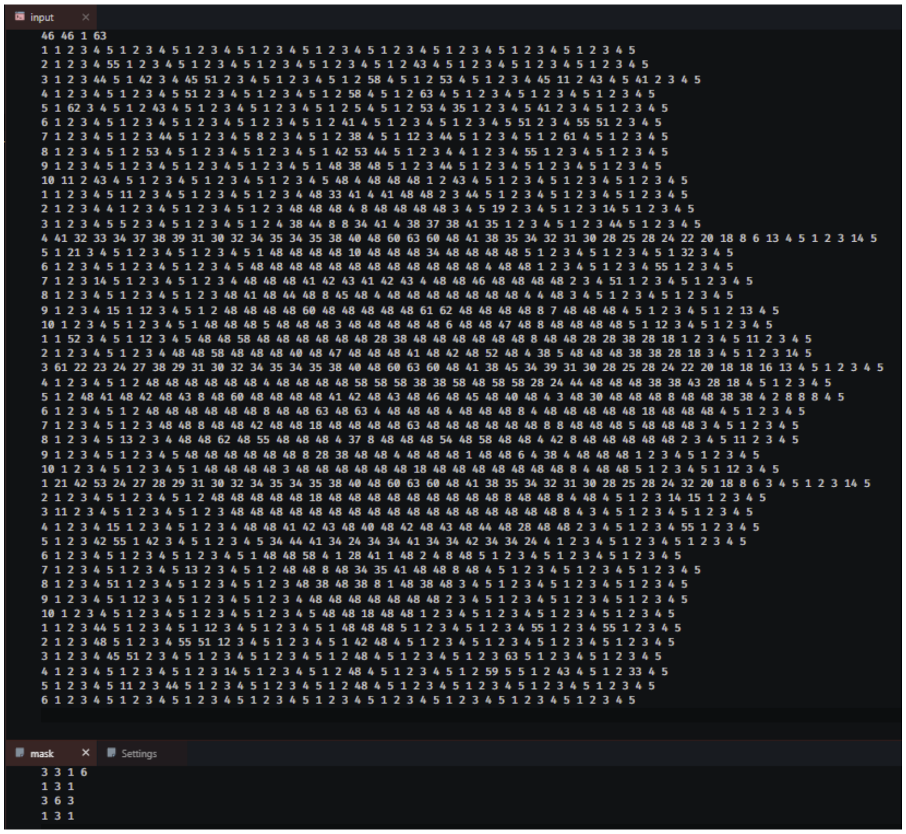
Similar to compute Median and Gaussian

```
1. newMin ← 999
2. newMax ← 0
3. For i = 1 < rows +1:
4. For j = 2 < cols +1:
    a. Avg ← 0
    b. loadNeighbor1DAry(i,j)
    c. Avg ← sumOf(neighborArray) / 9
    d. averageArray[i][j] =  Avg
    e. Check if there is newMin or newMax
```

```
/*
      Computer Vision
      Project 2
      Created by Adrian Noa

      usage:

java adrian_noa_main.java input mask 38 inputImg avgOut AvgThreshold MedianOut
MedianTreshold GaussOut GaussThreshold

      Change the value of third parameter(38) to change the Treshold Value
*/
```

```java
import java.io.*;
import java.util.Scanner;
import java.util.Arrays;

public class CVProject2 {

    public static void main(String[] args) throws IOException{
        if (args.length != 10){
            System.out.println("Not enough of arguments");
            return;
        }
        int threshold = Integer.parseInt(args[2]);
        try(
            Scanner inFile = readFile(args[0]);
            Scanner maskFile = readFile(args[1]);
            BufferedWriter inputImg = createFile(args[3]);
            BufferedWriter AvgOut = createFile(args[4]);
            BufferedWriter AvgThreshold = createFile(args[5]);
            BufferedWriter MedianOut = createFile(args[6]);
            BufferedWriter MedianThreshold = createFile(args[7]);
            BufferedWriter GaussOut = createFile(args[8]);
            BufferedWriter GaussThreshold = createFile(args[9]);
        ){
            ImageProcessing imgProcessing = new ImageProcessing(inFile, maskFile, threshold);
            load(imgProcessing);

            imgProcessing.imgReformat(imgProcessing.mirrorFramedAry, imgProcessing.minVal,
imgProcessing.maxVal, inputImg);

            imgProcessing.computeAvg();
            imgProcessing.imgReformat(imgProcessing.avgAry, imgProcessing.newMin, imgProcessing.newMax,
AvgOut);
            imgProcessing.threshold(imgProcessing.avgAry, imgProcessing.thrAry);
            imgProcessing.imgReformat(imgProcessing.thrAry, imgProcessing.newMin, imgProcessing.newMax,
AvgThreshold);

            imgProcessing.computeMedian();
            imgProcessing.imgReformat(imgProcessing.medianAry, imgProcessing.newMin, imgProcessing.newMax,
MedianOut);
            imgProcessing.threshold(imgProcessing.medianAry, imgProcessing.thrAry);
            imgProcessing.imgReformat(imgProcessing.thrAry, imgProcessing.newMin, imgProcessing.newMax,
MedianThreshold);

            imgProcessing.computeGauss();
            imgProcessing.imgReformat(imgProcessing.gaussAry, imgProcessing.newMin, imgProcessing.newMax,
GaussOut);
            imgProcessing.threshold(imgProcessing.gaussAry, imgProcessing.thrAry);
            imgProcessing.imgReformat(imgProcessing.thrAry, imgProcessing.newMin, imgProcessing.newMax,
GaussThreshold);
        }
    }

    public static void load(ImageProcessing imgProcessing){
        imgProcessing.allocate();
        imgProcessing.loadMask();
        imgProcessing.loadMask1DAry();
        imgProcessing.loadImage();
        imgProcessing.mirrorFraming();
    }

    public static Scanner readFile(String s) throws IOException{
        return new Scanner(new BufferedReader(new FileReader(s)));
    }

    public static BufferedWriter createFile(String s) throws IOException{
        return new BufferedWriter(new FileWriter(s));
    }
}

public class ImageProcessing {
        Scanner inFile, maskFile;
        int numRows, numCols, minVal, maxVal; // image header
        int maskRows, maskCols, maskMin, maskMax; //mask header
        int newMin, newMax, maskWeight;
        int thrVal;  // threshold value
        boolean includeZero;
        int[][] mirrorFramedAry, avgAry, medianAry, gaussAry, thrAry, mask2DAry;
        int[] neighbor1DAry, mask1DAry;

        ImageProcessing(Scanner inFile, Scanner maskFile, int threshold){
            this.inFile = inFile;
            this.maskFile = maskFile;
            this.numRows = inFile.nextInt();
            this.numCols = inFile.nextInt();
            this.minVal = inFile.nextInt();
            this.maxVal = inFile.nextInt();
            this.maskRows = maskFile.nextInt();
            this.maskCols = maskFile.nextInt();
            this.maskMin = maskFile.nextInt();
            this.maskMax = maskFile.nextInt();
            this.thrVal = threshold;
            this.includeZero = false; // change to false to Visualize treshold output files (no zeros only 1s)
        }

        public void threshold(int[][] in, int[][]out){
            newMin = 0;
            newMax = 1;
            for(int i=1; i<numRows+2; i++){
                for(int j=1; j<numCols+2; j++){
                    if (in[i][j] >= thrVal)
                        out[i][j] = 1;
                    else
                        out[i][j] = 0;
                }
            }
        }

        public void imgReformat(int[][] inAry, int newMin, int newMax, BufferedWriter outImg) throws
IOException{

            outImg.write(Integer.toString(numRows)+" ");
            outImg.write(Integer.toString(numCols)+" ");
            outImg.write(Integer.toString(newMin)+" ");
            outImg.write(Integer.toString(newMax)+"\n");

            int maxWidth = Integer.toString(newMax).length();
            String s;
            for (int i = 1; i<numRows+1; i++){
                for (int j = 1; j<numCols+1; j++){
                    s = Integer.toString(inAry[i][j]);
                    if(includeZero==true)
                        outImg.write(s);
                    else{
                        if(inAry[i][j]!=0)
                            outImg.write(s);
                        else outImg.write(" ");

                    }
                    int pixelWidth = s.length();
                    while(pixelWidth <= maxWidth){
                        outImg.write(" ");
                        pixelWidth++;
                    }
                }
                outImg.write("\n");
            }
        }

        public void mirrorFraming(){
            mirrorFramedAry[0][0] = mirrorFramedAry[2][2];
            mirrorFramedAry[0][numCols+1] = mirrorFramedAry[2][numCols-1];
            mirrorFramedAry[numRows+1][0] = mirrorFramedAry[numRows-1][2];
            mirrorFramedAry[numRows+1][numCols+1] = mirrorFramedAry[numRows-1][numCols-1];
            for(int i=1; i<numCols+1; i++){
                mirrorFramedAry[0][i] = mirrorFramedAry[2][i];
                mirrorFramedAry[numRows+1][i] = mirrorFramedAry[numRows-1][i];
            }
            for(int i=1; i<numRows+1; i++){
                mirrorFramedAry[i][0] = mirrorFramedAry[i][2];
                mirrorFramedAry[i][numCols+1] = mirrorFramedAry[i][numCols-1];
            }
        }

        public void loadImage(){
            for(int i=1; i<numRows+1; i++){
                for(int j=1; j<numCols+1; j++){
                    mirrorFramedAry[i][j] = inFile.nextInt();
                }
            }
        }

        public void loadMask(){
            for(int i=0; i<maskRows; i++){
                for(int j=0; j<maskCols; j++){
                    mask2DAry[i][j] = maskFile.nextInt();
                }
            }
        }

        public void loadMask1DAry(){
            this.maskWeight = 0;
            for(int i=0; i<maskRows; i++){
                for(int j=0; j<maskCols; j++){
                    mask1DAry[maskCols*i+j] = mask2DAry[i][j];
                    this.maskWeight += mask2DAry[i][j];
                }
            }
        }

        public void loadNeighbor1DAry(int r, int c){
            for(int i=0; i<3; i++){
                for(int j=0; j<3; j++){
                    neighbor1DAry[ (i*3) + j ] = mirrorFramedAry[r-1+i][c-1+j];
                }
            }
        }

        public void sort(int[] a){
            Arrays.sort(a);
        }

        public void computeAvg(){
            newMin = 9999;
            newMax = 0;
            for(int i=1; i<numRows+1; i++){
                for(int j=1; j<numCols+1; j++){
                    int avg = 0;
                    loadNeighbor1DAry(i,j);
                    for(int pixel:neighbor1DAry)avg+=pixel;
                    avgAry[i][j] = avg/9;
                    if (newMin > avgAry[i][j])
                        newMin = avgAry[i][j];
                    if (newMax < avgAry[i][j])
                        newMax = avgAry[i][j];
                }
            }
        }

        public void computeMedian(){
            newMin = 9999;
            newMax = 0;

            for(int i=1; i<numRows+1; i++){
                for(int j=1; j<numCols+1; j++){
                    loadNeighbor1DAry(i,j);
                    sort(neighbor1DAry);
                    medianAry[i][j] = neighbor1DAry[4];
                    if (newMin > medianAry[i][j])
                        newMin = medianAry[i][j];
                    if (newMax < medianAry[i][j])
                        newMax = medianAry[i][j];
                }
            }
        }

        public void computeGauss(){
            newMin = 9999;
            newMax = 0;
            for(int i=1; i<numRows+1; i++){
                for(int j=1; j<numCols+1; j++){
                    loadNeighbor1DAry(i,j);
                    gaussAry[i][j] = convolution(neighbor1DAry, mask1DAry);
                    if (newMin > gaussAry[i][j])
                        newMin = gaussAry[i][j];
                    if (newMax < gaussAry[i][j])
                        newMax = gaussAry[i][j];
                }
            }
        }

        public int convolution(int[] a, int[] mask){
            int result = 0;
            for(int i=0; i<9; i++){
                result += neighbor1DAry[i] * mask1DAry[i];
            }
            return result/maskWeight;
        }

        public void allocate(){
            mirrorFramedAry = new int[this.numRows+2][this.numCols+2];
            avgAry = new int[this.numRows+2][this.numCols+2];
            medianAry = new int[this.numRows+2][this.numCols+2];
            gaussAry = new int[this.numRows+2][this.numCols+2];
            thrAry = new int[this.numRows+2][this.numCols+2];
            mask2DAry = new int[this.maskRows][this.maskCols];
            mask1DAry = new int[this.maskRows*this.maskCols];
            neighbor1DAry = new int[9];
        }

    }
```

Input files(image and mask)



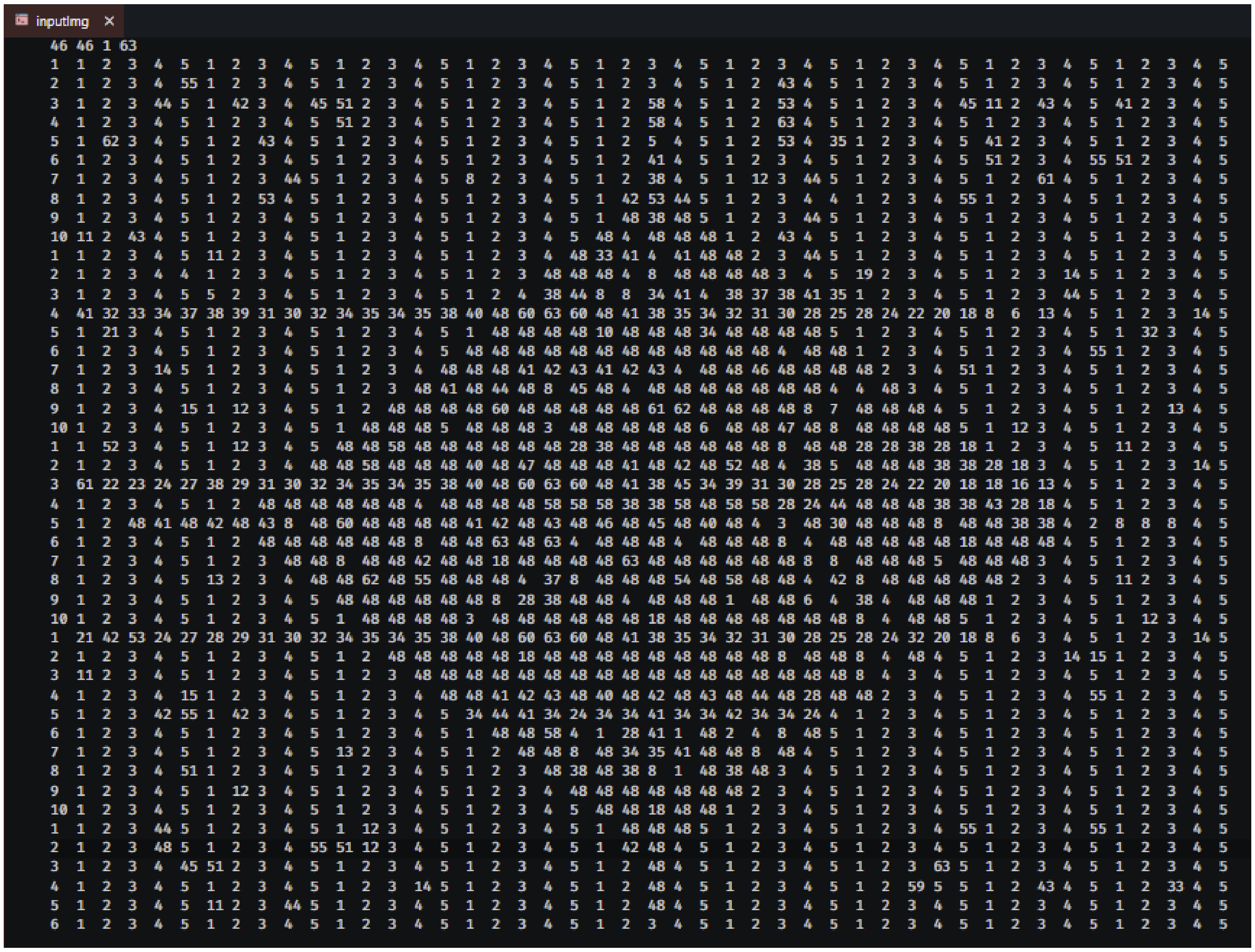
input

```
46 46 1 63
1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 44 5 1 42 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 58 4 5 1 2 53 4 5 1 2 3 4 45 11 2 43 4 5 41 2 3 4 5
4 1 2 3 4 5 1 2 3 4 5 51 2 3 4 5 1 2 3 4 5 1 2 58 4 5 1 2 63 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
5 1 62 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 5 4 5 1 2 53 4 35 1 2 3 4 5 41 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 41 4 5 1 2 3 4 5 1 2 3 4 5 51 2 3 4 55 51 2 3 4 5
7 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5 8 2 3 4 5 1 2 38 4 5 1 12 3 44 5 1 2 3 4 5 1 2 61 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 53 4 5 1 2 3 4 5 1 2 3 4 5 1 42 53 44 5 1 2 3 4 4 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 38 48 5 1 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
10 11 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 4 48 48 48 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 4 5 11 2 3 4 5 1 2 3 4 5 1 2 3 4 48 33 41 4 41 48 48 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 4 1 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 4 8 48 48 48 48 3 4 5 19 2 3 4 5 1 2 3 14 5 1 2 3 4 5
3 1 2 3 4 5 5 2 3 4 5 1 2 3 4 5 1 2 4 38 44 8 8 34 41 4 38 37 38 41 35 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5
4 41 32 33 34 37 38 39 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 22 20 18 8 6 13 4 5 1 2 3 14 5
5 1 21 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 10 48 48 48 34 48 48 48 48 5 1 2 3 4 5 1 2 3 4 5 1 32 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 48 48 48 48 48 48 4 48 48 1 2 3 4 5 1 2 3 4 55 1 2 3 4 5
7 1 2 3 14 5 1 2 3 4 5 1 2 3 4 48 48 48 41 42 43 41 42 43 4 48 48 46 48 48 48 48 2 3 4 51 1 2 3 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 3 4 5 1 2 3 48 41 48 44 48 8 45 48 4 48 48 48 48 48 48 48 4 4 48 3 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 15 1 12 3 4 5 1 2 48 48 48 48 60 48 48 48 48 48 61 62 48 48 48 48 8 7 48 48 48 4 5 1 2 3 4 5 1 2 13 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 5 48 48 48 3 48 48 48 48 48 6 48 48 47 48 8 48 48 48 48 5 1 12 3 4 5 1 2 3 4 5
1 1 52 3 4 5 1 12 3 4 5 48 48 58 48 48 48 48 48 48 28 38 48 48 48 48 48 48 8 48 48 28 28 38 28 18 1 2 3 4 5 11 2 3 4 5
2 1 2 3 4 5 1 2 3 4 48 48 58 48 48 48 40 48 47 48 48 48 41 48 42 48 52 48 4 38 5 48 48 48 38 38 28 18 3 4 5 1 2 3 14 5
3 61 22 23 24 27 38 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 45 34 39 31 30 28 25 28 24 22 20 18 18 16 13 4 5 1 2 3 4 5
4 1 2 3 4 5 1 2 48 48 48 48 48 48 4 48 48 48 48 58 58 58 38 38 58 48 58 58 28 24 44 48 48 48 38 38 43 28 18 4 5 1 2 3 4 5
5 1 2 48 41 48 42 48 43 8 48 60 48 48 48 48 41 42 48 43 48 46 48 45 48 40 48 4 3 48 30 48 48 48 8 48 48 38 38 4 2 8 8 8 4 5
6 1 2 3 4 5 1 2 48 48 48 48 48 48 8 48 48 63 48 63 4 48 48 48 4 48 48 48 8 4 48 48 48 48 48 18 48 48 48 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 48 48 8 48 48 42 48 48 18 48 48 48 48 63 48 48 48 48 48 48 8 8 48 48 48 5 48 48 48 3 4 5 1 2 3 4 5
8 1 2 3 4 5 13 2 3 4 48 48 62 48 55 48 48 48 4 37 8 48 48 48 54 48 58 48 48 4 42 8 48 48 48 48 48 2 3 4 5 11 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 8 28 38 48 48 4 48 48 48 1 48 48 6 4 38 4 48 48 48 1 2 3 4 5 1 2 3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 3 48 48 48 48 48 48 18 48 48 48 48 48 48 48 8 4 48 48 5 1 2 3 4 5 1 12 3 4 5
1 21 42 53 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 32 20 18 8 6 3 4 5 1 2 3 14 5
2 1 2 3 4 5 1 2 3 4 5 1 2 48 48 48 48 48 18 48 48 48 48 48 48 48 48 48 8 48 48 8 4 48 4 5 1 2 3 14 15 1 2 3 4 5
3 11 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 4 3 4 5 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 15 1 2 3 4 5 1 2 3 4 48 48 41 42 43 48 40 48 42 48 43 48 44 48 28 48 48 2 3 4 5 1 2 3 4 55 1 2 3 4 5
5 1 2 3 42 55 1 42 3 4 5 1 2 3 4 5 34 44 41 34 24 34 34 41 34 34 42 34 34 24 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 58 4 1 28 41 1 48 2 4 8 48 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 4 5 13 2 3 4 5 1 2 48 48 8 48 34 35 41 48 48 8 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
8 1 2 3 4 51 1 2 3 4 5 1 2 3 4 5 1 2 3 48 38 48 38 8 1 48 38 48 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 2 3 4 48 48 48 48 48 48 48 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 18 48 48 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 44 5 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 48 48 48 5 1 2 3 4 5 1 2 3 4 55 1 2 3 4 55 1 2 3 4 5
2 1 2 3 48 5 1 2 3 4 55 51 12 3 4 5 1 2 3 4 5 1 42 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 63 5 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 5 1 2 3 4 5 1 2 3 14 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 59 5 5 1 2 43 4 5 1 2 33 4 5
5 1 2 3 4 5 11 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

mask Settings

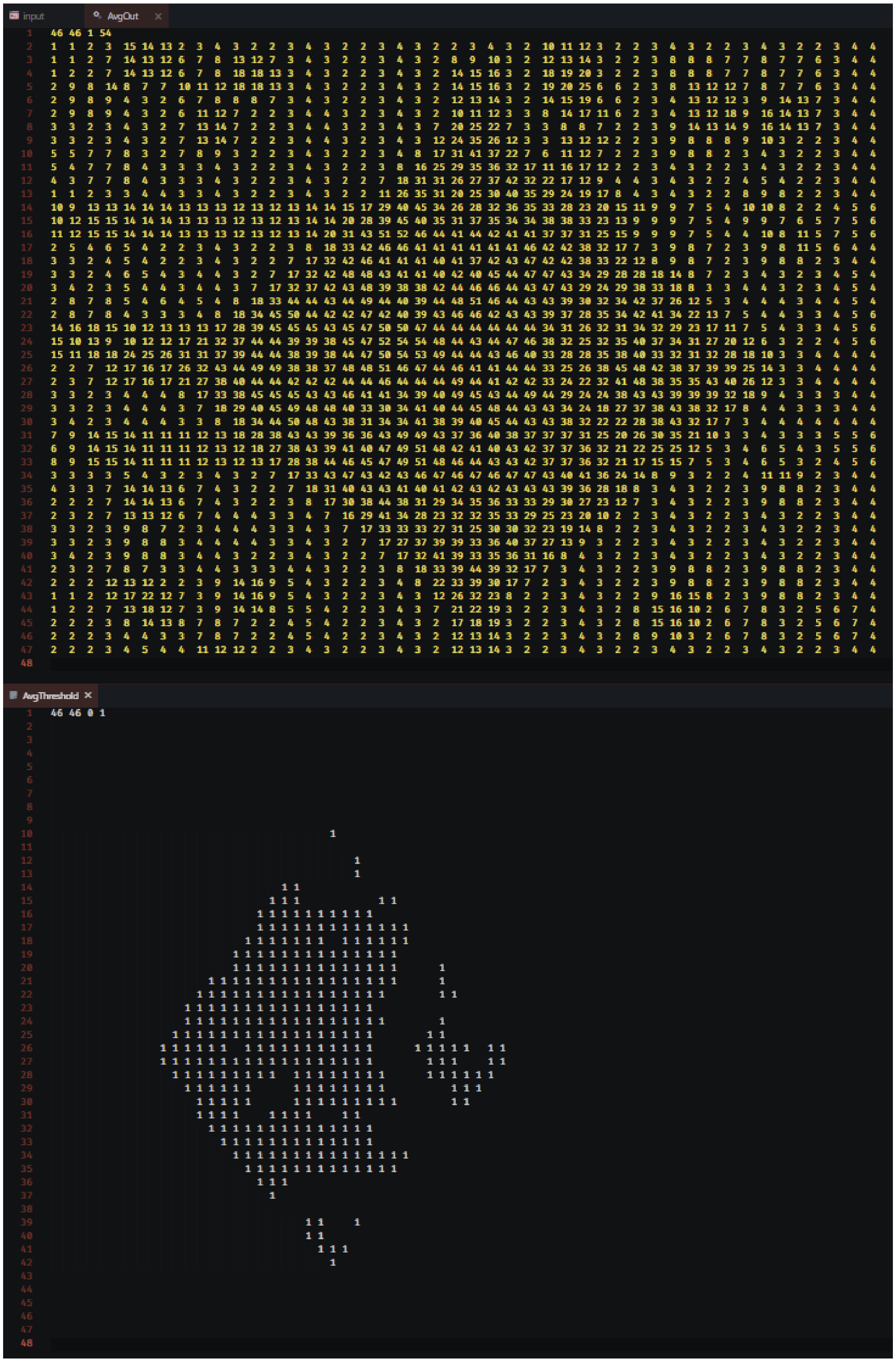
```
3 3 1 6
1 3 1
3 6 3
1 3 1
```

Reformatted Input Image



```
46 46 1 63
1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 44 5 1 42 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 58 4 5 1 2 53 4 5 1 2 3 4 45 11 2 43 4 5 41 2 3 4 5
4 1 2 3 4 5 1 2 3 4 5 51 2 3 4 5 1 2 3 4 5 1 2 58 4 5 1 2 63 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
5 1 62 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 5 4 5 1 2 53 4 35 1 2 3 4 5 41 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 41 4 5 1 2 3 4 5 1 2 3 4 5 51 2 3 4 55 51 2 3 4 5
7 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5 8 2 3 4 5 1 2 38 4 5 1 12 3 44 5 1 2 3 4 5 1 2 61 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 53 4 5 1 2 3 4 5 1 2 3 4 5 1 42 53 44 5 1 2 3 4 4 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 38 48 5 1 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
10 11 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 4 48 48 48 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 4 5 11 2 3 4 5 1 2 3 4 5 1 2 3 4 48 33 41 4 41 48 48 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 4 1 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 4 8 48 48 48 48 3 4 5 19 2 3 4 5 1 2 3 14 5 1 2 3 4 5
3 1 2 3 4 5 5 2 3 4 5 1 2 3 4 5 1 2 4 38 44 8 8 34 41 4 38 37 38 41 35 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5
4 41 32 33 34 37 38 39 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 22 20 18 8 6 13 4 5 1 2 3 14 5
5 1 21 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 10 48 48 48 34 48 48 48 48 5 1 2 3 4 5 1 2 3 4 5 1 32 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 48 48 48 48 48 48 4 48 48 1 2 3 4 5 1 2 3 4 55 1 2 3 4 5
7 1 2 3 14 5 1 2 3 4 5 1 2 3 4 48 48 48 41 42 43 41 42 43 4 48 48 46 48 48 48 48 2 3 4 51 1 2 3 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 3 4 5 1 2 3 48 41 48 44 48 8 45 48 4 48 48 48 48 48 48 48 4 4 48 3 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 15 1 12 3 4 5 1 2 48 48 48 48 60 48 48 48 48 48 61 62 48 48 48 48 8 7 48 48 48 4 5 1 2 3 4 5 1 2 13 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 5 48 48 48 3 48 48 48 48 48 6 48 48 47 48 8 48 48 48 48 5 1 12 3 4 5 1 2 3 4 5
1 1 52 3 4 5 1 12 3 4 5 48 48 58 48 48 48 48 48 48 28 38 48 48 48 48 48 48 8 48 48 28 28 38 28 18 1 2 3 4 5 11 2 3 4 5
2 1 2 3 4 5 1 2 3 4 48 48 58 48 48 48 40 48 47 48 48 48 41 48 42 48 52 48 4 38 5 48 48 48 38 38 28 18 3 4 5 1 2 3 14 5
3 61 22 23 24 27 38 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 45 34 39 31 30 28 25 28 24 22 20 18 18 16 13 4 5 1 2 3 4 5
4 1 2 3 4 5 1 2 48 48 48 48 48 48 4 48 48 48 48 58 58 58 38 38 58 48 58 58 28 24 44 48 48 48 38 38 43 28 18 4 5 1 2 3 4 5
5 1 2 48 41 48 42 48 43 8 48 60 48 48 48 48 41 42 48 43 48 46 48 45 48 40 48 4 3 48 30 48 48 48 8 48 48 38 38 4 2 8 8 8 4 5
6 1 2 3 4 5 1 2 48 48 48 48 48 48 8 48 48 63 48 63 4 48 48 48 4 48 48 48 8 4 48 48 48 48 48 18 48 48 48 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 48 48 8 48 48 42 48 48 18 48 48 48 48 63 48 48 48 48 48 48 8 8 48 48 48 5 48 48 48 3 4 5 1 2 3 4 5
8 1 2 3 4 5 13 2 3 4 48 48 62 48 55 48 48 48 4 37 8 48 48 48 54 48 58 48 48 4 42 8 48 48 48 48 48 2 3 4 5 11 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 8 28 38 48 48 4 48 48 48 1 48 48 6 4 38 4 48 48 48 1 2 3 4 5 1 2 3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 3 48 48 48 48 48 48 18 48 48 48 48 48 48 48 8 4 48 48 5 1 2 3 4 5 1 12 3 4 5
1 21 42 53 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 32 20 18 8 6 3 4 5 1 2 3 14 5
2 1 2 3 4 5 1 2 3 4 5 1 2 48 48 48 48 48 18 48 48 48 48 48 48 48 48 48 8 48 48 8 4 48 4 5 1 2 3 14 15 1 2 3 4 5
3 11 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 4 3 4 5 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 15 1 2 3 4 5 1 2 3 4 48 48 41 42 43 48 40 48 42 48 43 48 44 48 28 48 48 2 3 4 5 1 2 3 4 55 1 2 3 4 5
5 1 2 3 42 55 1 42 3 4 5 1 2 3 4 5 34 44 41 34 24 34 34 41 34 34 42 34 34 24 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 58 4 1 28 41 1 48 2 4 8 48 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 4 5 13 2 3 4 5 1 2 48 48 8 48 34 35 41 48 48 8 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
8 1 2 3 4 51 1 2 3 4 5 1 2 3 4 5 1 2 3 48 38 48 38 8 1 48 38 48 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 2 3 4 48 48 48 48 48 48 48 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 18 48 48 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 44 5 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 48 48 48 5 1 2 3 4 5 1 2 3 4 55 1 2 3 4 55 1 2 3 4 5
2 1 2 3 48 5 1 2 3 4 55 51 12 3 4 5 1 2 3 4 5 1 42 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 63 5 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 5 1 2 3 4 5 1 2 3 14 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 59 5 5 1 2 43 4 5 1 2 33 4 5
5 1 2 3 4 5 11 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

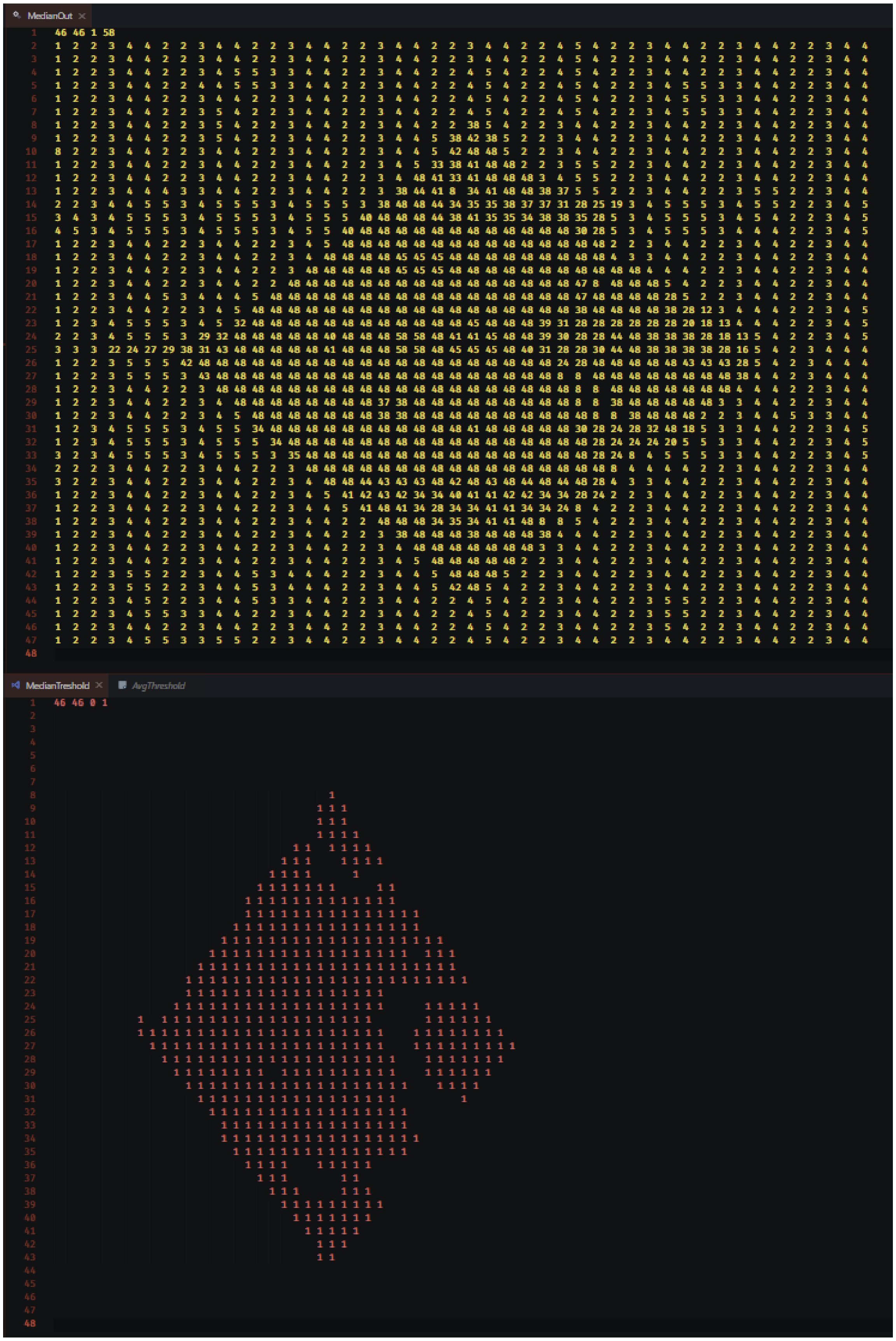
Reformatted Average and Binary Average Treshold



```
46 46 1 54
1 1 2 3 15 14 13 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 10 11 12 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
1 1 2 7 14 13 12 6 7 8 13 12 7 3 4 3 2 2 3 4 3 2 8 9 10 3 2 12 13 14 3 2 2 3 8 8 8 7 7 8 7 7 6 3 4 4
1 2 2 7 14 13 12 6 7 8 18 18 13 3 4 3 2 2 3 4 3 2 14 15 16 3 2 18 19 20 3 2 2 3 8 8 8 7 7 8 7 7 6 3 4 4
2 9 8 14 8 7 7 10 11 12 18 18 13 3 4 3 2 2 3 4 3 2 14 15 16 3 2 19 20 25 6 6 2 3 8 13 12 12 7 8 7 7 6 3 4 4
2 9 8 9 4 3 2 6 7 8 8 8 7 3 4 3 2 2 3 4 3 2 12 13 14 3 2 14 15 19 6 6 2 3 4 13 12 12 3 9 14 13 7 3 4 4
2 9 8 9 4 3 2 6 11 12 7 2 2 3 4 4 3 2 3 4 3 2 10 11 12 3 3 8 14 17 11 6 2 3 4 13 12 18 9 16 14 13 7 3 4 4
3 3 2 3 4 3 2 7 13 14 7 2 2 3 4 4 3 2 3 4 3 7 20 25 22 7 3 3 8 8 7 2 2 3 9 14 13 14 9 16 14 13 7 3 4 4
3 3 2 3 4 3 2 7 13 14 7 2 2 3 4 4 3 2 3 4 3 12 24 35 26 12 3 3 13 12 12 2 2 3 9 8 8 8 9 10 3 2 2 3 4 4
5 5 7 7 8 3 2 7 8 9 3 2 2 3 4 3 2 2 3 4 8 17 31 41 37 22 7 6 11 12 7 2 2 3 9 8 8 2 3 4 3 2 2 3 4 4
5 4 7 7 8 4 3 3 3 4 3 2 2 3 4 3 2 2 3 8 16 25 29 35 36 32 17 11 16 17 12 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
4 3 7 7 8 4 3 3 3 4 3 2 2 3 4 3 2 2 7 18 31 31 26 27 37 42 32 22 17 12 9 4 4 3 4 3 2 2 4 5 4 2 2 3 4 4
1 1 2 3 3 4 4 3 3 4 3 2 2 3 4 3 2 2 11 26 35 31 20 25 30 40 35 29 24 19 17 8 4 3 4 3 2 2 8 9 8 2 2 3 4 4
10 9 13 13 14 14 14 13 13 13 12 13 12 13 14 14 15 17 29 40 45 34 26 28 32 36 35 33 28 23 20 15 11 9 9 7 5 4 10 10 8 2 2 4 5 6
10 12 15 15 14 14 14 13 13 13 12 13 12 13 14 14 20 28 39 45 40 35 31 37 35 34 34 38 38 33 23 13 9 9 9 7 5 4 9 9 7 6 5 7 5 6
11 12 15 15 14 14 14 13 13 13 12 13 12 13 14 20 31 43 51 52 46 44 41 44 42 41 41 37 37 31 25 15 9 9 9 7 5 4 4 10 8 11 5 7 5 6
2 5 4 6 5 4 2 2 3 4 3 2 2 3 8 18 33 42 46 46 41 41 41 41 41 41 46 42 42 38 32 17 7 3 9 8 7 2 3 9 8 11 5 6 4 4
3 3 2 4 5 4 2 2 3 4 3 2 2 7 17 32 42 46 41 41 41 40 41 37 42 43 47 42 42 38 33 22 12 8 9 8 7 2 3 9 8 8 2 3 4 4
3 3 2 4 6 5 4 3 4 4 3 2 7 17 32 42 48 48 43 41 41 40 42 40 45 44 47 47 43 34 29 28 28 18 14 8 7 2 3 4 3 2 3 4 5 4
3 4 2 3 5 4 4 3 4 4 3 7 17 32 37 42 43 48 39 38 38 42 44 46 46 44 43 47 43 29 24 29 38 33 18 8 3 3 4 4 3 2 3 4 5 4
2 8 7 8 5 4 6 4 5 4 8 18 33 44 44 43 44 49 44 40 39 44 48 51 46 44 43 43 39 30 32 34 42 37 26 12 5 3 4 4 4 3 4 4 5 4
2 8 7 8 4 3 3 3 4 8 18 34 45 50 44 42 42 47 42 40 39 43 46 46 42 43 43 39 37 28 35 34 42 41 34 22 13 7 5 4 4 3 3 4 5 6
14 16 18 15 10 12 13 13 13 17 28 39 45 45 45 43 45 47 50 50 47 44 44 44 44 44 44 34 31 26 32 31 34 32 29 23 17 11 7 5 4 3 3 4 5 6
15 10 13 9 10 12 12 17 21 32 37 44 44 39 39 38 45 47 52 54 54 48 44 43 44 47 46 38 32 25 32 35 40 37 34 31 27 20 12 6 3 2 2 4 5 6
15 11 18 18 24 25 26 31 31 37 39 44 44 38 39 38 44 47 50 54 53 49 44 44 43 46 40 33 28 28 35 38 40 33 32 31 32 28 18 10 3 3 4 4 4 4
2 2 7 12 17 16 17 26 32 43 44 49 49 38 38 37 48 48 51 46 47 44 46 41 41 44 44 33 25 26 38 45 48 42 38 37 39 39 25 14 3 3 4 4 4 4
2 3 7 12 17 16 17 21 27 38 40 44 44 42 42 42 44 44 46 44 44 44 49 44 41 42 42 33 24 22 32 41 48 38 35 35 43 40 26 12 3 3 4 4 4 4
3 3 2 3 4 4 4 8 17 33 38 45 45 45 43 43 46 41 41 34 39 40 49 45 43 44 49 44 29 24 24 38 43 43 39 39 39 32 18 9 4 3 3 3 4 4
3 3 2 3 4 4 4 3 7 18 29 40 45 49 48 48 40 33 30 34 41 40 44 45 48 44 43 43 34 24 18 27 37 38 43 38 32 17 8 4 4 3 3 3 4 4
3 4 2 3 4 4 4 3 3 8 18 34 44 50 48 43 38 31 34 34 41 38 39 40 45 44 43 43 38 32 22 22 28 38 43 32 17 7 3 4 4 4 4 4 4 4
7 9 14 15 14 11 11 11 12 13 18 28 38 43 43 39 36 36 43 49 49 43 37 36 40 38 37 37 37 31 25 20 26 30 35 21 10 3 3 4 3 3 3 5 5 6
6 9 14 15 14 11 11 11 12 13 12 18 27 38 43 39 41 40 47 49 51 48 42 41 40 43 42 37 37 36 32 21 22 25 25 12 5 3 4 6 5 4 3 5 5 6
8 9 15 15 14 11 11 11 12 13 12 13 17 28 38 44 46 45 47 49 51 48 46 44 43 43 42 37 37 36 32 21 17 15 15 7 5 3 4 6 5 3 2 4 5 6
3 3 3 3 5 4 3 2 3 4 3 2 7 17 33 43 47 43 42 43 46 47 46 47 46 47 47 43 40 41 36 24 14 8 9 3 2 2 4 11 11 9 2 3 4 4
4 3 3 7 14 14 13 6 7 4 3 2 2 7 18 31 40 43 43 41 40 41 42 43 42 43 43 43 39 36 28 18 8 3 4 3 2 2 3 9 8 8 2 3 4 4
2 2 2 7 14 14 13 6 7 4 3 2 2 3 8 17 30 38 44 38 31 29 34 35 36 33 33 29 30 27 23 12 7 3 4 3 2 2 3 9 8 8 2 3 4 4
2 3 2 7 13 13 12 6 7 4 4 4 3 3 4 7 16 29 41 34 28 23 32 32 35 33 29 25 23 20 10 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
3 3 2 3 9 8 7 2 3 4 4 4 3 3 4 3 7 17 33 33 33 27 31 25 30 30 32 23 19 14 8 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
3 3 2 3 9 8 8 3 4 4 4 4 3 3 4 3 2 7 17 27 37 39 39 33 36 40 37 27 13 9 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
3 4 2 3 9 8 8 3 4 4 3 2 2 3 4 3 2 2 7 17 32 41 39 33 35 36 31 16 8 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
2 3 2 7 8 7 3 3 4 4 3 3 3 4 4 3 2 2 3 8 18 33 39 44 39 32 17 7 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4 4
2 2 2 12 13 12 2 2 3 9 14 16 9 5 4 3 2 2 3 4 8 22 33 39 30 17 7 2 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4 4
1 1 2 12 17 22 12 7 3 9 14 16 9 5 4 3 2 2 3 4 3 12 26 32 23 8 2 2 3 4 3 2 2 9 16 15 8 2 3 9 8 8 2 3 4 4
1 2 2 7 13 18 12 7 3 9 14 14 8 5 5 4 2 2 3 4 3 7 21 22 19 3 2 2 3 4 3 2 8 15 16 10 2 6 7 8 3 2 5 6 7 4
2 2 2 3 8 14 13 8 7 8 7 2 2 4 5 4 2 2 3 4 3 2 17 18 19 3 2 2 3 4 3 2 8 15 16 10 2 6 7 8 3 2 5 6 7 4
2 2 2 3 4 4 3 3 7 8 7 2 2 4 5 4 2 2 3 4 3 2 12 13 14 3 2 2 3 4 3 2 8 9 10 3 2 6 7 8 3 2 5 6 7 4
2 2 2 3 4 5 4 4 11 12 12 2 2 3 4 3 2 2 3 4 3 2 12 13 14 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 4
```

```
46 46 0 1








                                              1

                                                  1
                                                  1
                                      1 1
                                    1 1 1             1 1
                                  1 1 1 1 1 1 1 1 1 1
                                  1 1 1 1 1 1 1 1 1 1 1 1 1
                                1 1 1 1 1 1 1   1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1       1
                          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1       1
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1
                    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1 1
                  1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1 1       1 1 1 1 1   1 1
                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1 1 1     1 1
                    1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1       1 1 1 1 1 1
                      1 1 1 1 1 1       1 1 1 1 1 1 1 1           1 1 1
                        1 1 1 1 1       1 1 1 1 1 1 1 1 1         1 1
                        1 1 1 1     1 1 1 1     1 1
                          1 1 1 1 1 1 1 1 1 1 1 1 1 1
                            1 1 1 1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                                1 1 1 1 1 1 1 1 1 1 1 1 1
                                  1 1 1
                                    1

                                          1 1     1
                                          1 1
                                            1 1 1
                                              1





```

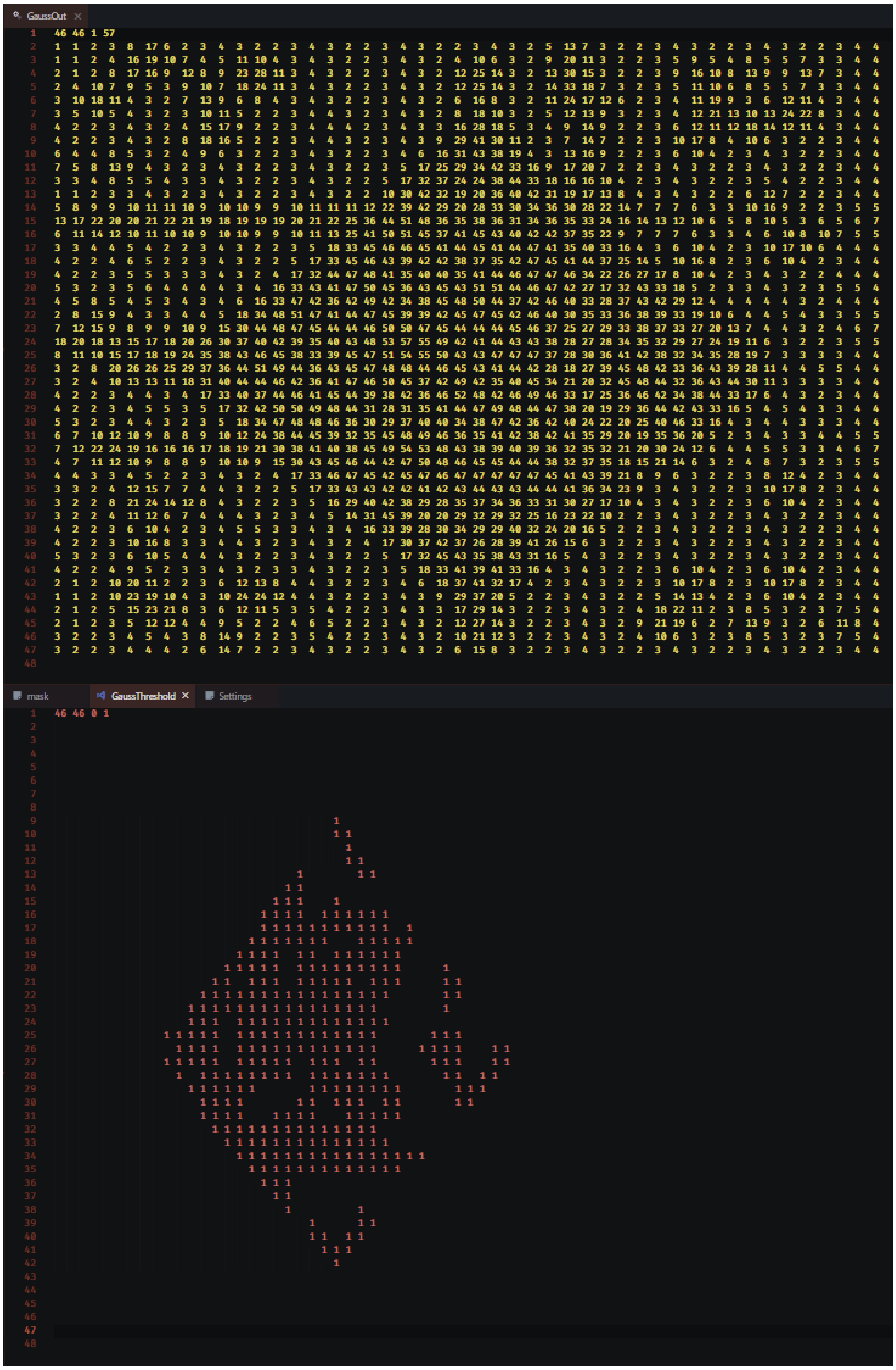
Reformatted Median and Binary Median Treshold



```
46 46 1 58
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  4  4  5  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4  2  2  38 5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4  5  38 42 38 5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
8  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  5  42 48 48 5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  33 38 41 48 48 2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  48 41 33 41 48 48 48 3  4  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  4  3  3  4  4  2  2  3  4  4  2  2  3  38 44 41 8  34 41 48 48 38 37 5  5  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4
2  2  3  4  4  5  5  3  4  5  5  5  3  4  5  5  5  3  38 48 48 44 34 35 35 38 37 37 31 28 25 19 3  4  5  5  5  3  4  5  5  2  2  3  4  5
3  4  3  4  5  5  5  3  4  5  5  5  3  4  5  5  5  40 48 48 48 44 38 41 35 35 34 38 38 35 28 5  3  4  5  5  5  3  4  5  4  2  2  3  4  5
4  5  3  4  5  5  5  3  4  5  5  5  3  4  5  5  40 48 48 48 48 48 48 48 48 48 48 48 48 30 28 5  3  4  5  5  5  3  4  4  4  2  2  3  4  5
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  48 48 48 48 45 45 45 48 48 48 48 48 48 48 48 48 4  3  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  48 48 48 48 48 45 45 45 48 48 48 48 48 48 48 48 48 48 48 4  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 47 8  48 48 48 5  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  5  3  4  4  4  5  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 47 48 48 48 48 28 5  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 38 48 48 48 48 38 28 12 3  4  4  4  2  2  3  4  5
1  2  3  4  5  5  5  3  4  5  32 48 48 48 48 48 48 48 48 48 48 48 48 45 48 48 48 39 31 28 28 28 28 28 28 20 18 13 4  4  4  2  2  3  4  5
2  2  3  4  5  5  5  3  29 32 48 48 48 48 48 40 48 48 48 58 58 48 41 41 45 48 48 39 30 28 28 44 48 38 38 38 28 18 13 5  4  2  2  3  4  5
3  3  3  22 24 27 29 38 31 43 48 48 48 48 48 41 48 48 48 58 58 48 45 45 45 48 40 31 28 28 30 44 48 38 38 38 38 28 16 5  4  2  3  4  4  4
1  2  2  3  5  5  5  42 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 24 28 48 48 48 48 48 43 43 43 28 5  4  2  3  4  4  4
1  2  2  3  5  5  5  3  43 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  8  48 48 48 48 48 48 48 48 38 4  4  2  3  4  4  4
1  2  2  3  4  4  2  2  3  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  8  48 48 48 48 48 48 48 4  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  48 48 48 48 48 48 48 48 37 38 48 48 48 48 48 48 48 48 48 8  8  38 48 48 48 48 48 3  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 38 38 48 48 48 48 48 48 48 48 48 48 48 8  8  38 48 48 48 2  2  3  4  4  5  3  3  4  4
1  2  3  4  5  5  5  3  4  5  5  34 48 48 48 48 48 48 48 48 48 48 48 41 48 48 48 48 48 30 28 24 28 32 48 18 5  3  3  4  4  2  2  3  4  5
1  2  3  4  5  5  5  3  4  5  5  5  34 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 28 24 24 24 20 5  5  3  3  4  4  2  2  3  4  5
3  2  3  4  5  5  5  3  4  5  5  5  3  35 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 28 24 8  4  5  5  5  3  3  4  4  2  2  3  4  5
2  2  2  3  4  4  2  2  3  4  4  2  2  3  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  4  4  4  4  2  2  3  4  4  2  2  3  4  4
3  2  2  3  4  4  2  2  3  4  4  2  2  3  4  48 48 44 43 43 43 48 42 48 43 48 44 48 44 48 28 4  3  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  41 42 43 42 34 34 40 41 41 42 42 34 34 28 24 2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  5  41 48 41 34 28 34 34 41 41 34 34 24 8  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  48 48 48 34 35 34 41 41 48 8  8  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  38 48 48 48 38 48 48 48 38 4  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  48 48 48 48 48 48 48 3  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  5  5  2  2  3  4  4  5  3  4  4  4  2  2  3  4  4  5  48 48 48 5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  5  5  2  2  3  4  4  5  3  4  4  4  2  2  3  4  4  5  42 48 5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  5  2  2  3  4  4  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4
1  2  2  3  4  5  5  3  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4

```

```
46 46 0 1






                                              1
                                            1 1 1
                                            1 1 1
                                            1 1 1 1
                                        1 1   1 1 1 1
                                      1 1 1     1 1 1 1
                                    1 1 1 1       1
                                  1 1 1 1 1 1 1       1 1
                                1 1 1 1 1 1 1 1 1 1 1 1 1
                                1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                            1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   1 1 1
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1       1 1 1 1 1
              1   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1 1 1 1 1 1
              1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1
                1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1
                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1
                    1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     1 1 1 1
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1           1
                          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                            1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                            1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                                1 1 1 1     1 1 1 1 1
                                  1 1 1         1 1
                                    1 1 1       1 1 1
                                      1 1 1 1 1 1 1 1 1
                                        1 1 1 1 1 1 1
                                          1 1 1 1 1
                                            1 1 1
                                            1 1




```

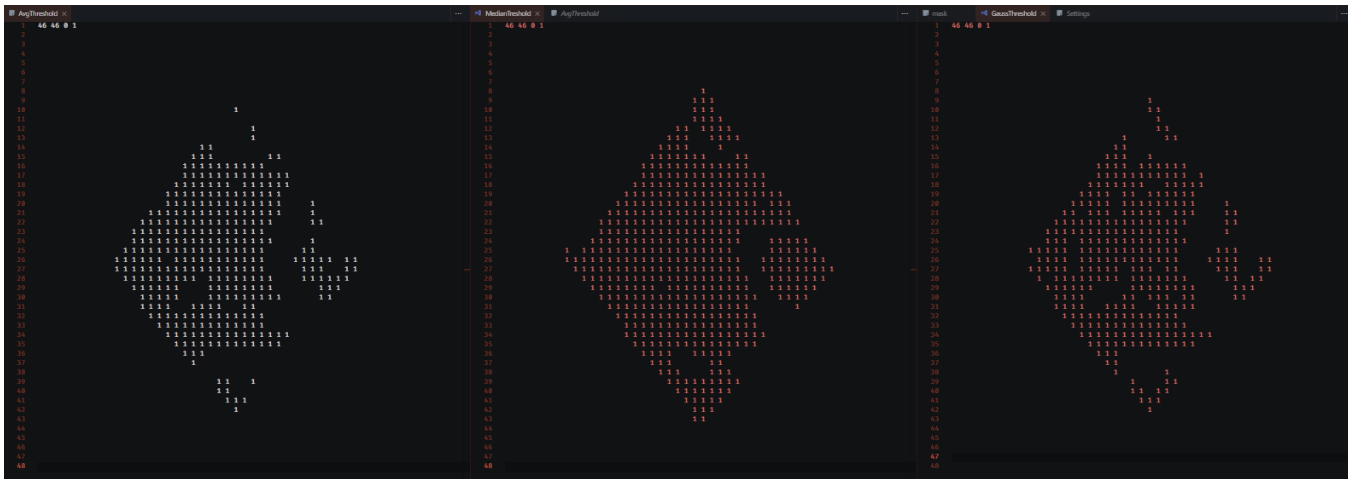
Reformatted Gaussian and Binary Gaussian Treshold



```
46 46 1 57
1  1  2  3  8  17 6  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  5  13 7  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
1  1  2  4  16 19 10 7  4  5  11 10 4  3  4  3  2  2  3  4  3  2  4  10 6  3  2  9  20 11 3  2  2  3  5  9  5  4  8  5  5  7  3  3  4  4
2  1  2  8  17 16 9  12 8  9  23 28 11 3  4  3  2  2  3  4  3  2  12 25 14 3  2  13 30 15 3  2  2  3  9  16 10 8  13 9  9  13 7  3  4  4
2  4  10 7  9  5  3  9  10 7  18 24 11 3  4  3  2  2  3  4  3  2  12 25 14 3  2  14 33 18 7  3  2  3  5  11 10 6  8  5  5  7  3  3  4  4
3  10 18 11 4  3  2  7  13 9  6  8  4  3  4  3  2  2  3  4  3  2  6  16 8  3  2  11 24 17 12 6  2  3  4  11 19 9  3  6  12 11 4  3  4  4
3  5  10 5  4  3  2  3  10 11 5  2  2  3  4  4  3  2  3  4  3  2  8  18 10 3  2  5  12 13 9  3  2  3  4  12 21 13 10 13 24 22 8  3  4  4
4  2  2  3  4  3  2  4  15 17 9  2  2  3  4  4  4  2  3  4  3  3  16 28 18 5  3  4  9  14 9  2  2  3  6  12 11 12 18 14 12 11 4  3  4  4
4  2  2  3  4  3  2  8  18 16 5  2  2  3  4  4  3  2  3  4  3  9  29 41 30 11 2  3  7  14 7  2  2  3  10 17 8  4  10 6  3  2  2  3  4  4
6  4  4  8  5  3  2  4  9  6  3  2  2  3  4  3  2  2  3  4  6  16 31 43 38 19 4  3  13 16 9  2  2  3  6  10 4  2  3  4  3  2  2  3  4  4
7  5  8  13 9  4  3  2  3  4  3  2  2  3  4  3  2  2  3  5  17 25 29 34 42 33 16 9  17 20 7  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
3  3  4  8  5  5  4  3  3  4  3  2  2  3  4  3  2  2  5  17 32 37 24 24 38 44 33 18 16 16 10 4  2  3  4  3  2  2  3  5  4  2  2  3  4  4
1  1  2  3  3  4  3  2  3  4  3  2  2  3  4  3  2  2  10 30 42 32 19 20 36 40 42 31 19 17 13 8  4  3  4  3  2  2  6  12 7  2  2  3  4  4
5  8  9  9  10 11 11 10 9  10 10 9  9  10 11 11 11 12 22 39 42 29 20 28 33 30 34 36 30 28 22 14 7  7  7  6  3  3  10 16 9  2  2  3  5  5
13 17 22 20 20 21 22 21 19 18 19 19 19 20 21 22 25 36 44 51 48 36 35 38 36 31 34 36 35 33 24 16 14 13 12 10 6  5  8  10 5  3  6  5  6  7
6  11 14 12 10 11 10 10 9  10 10 9  9  10 11 13 25 41 50 51 45 37 41 45 43 40 42 42 37 35 22 9  7  7  7  6  3  3  4  6  10 8  10 7  5  5
3  3  4  4  5  4  2  2  3  4  3  2  2  3  5  18 33 45 46 46 45 41 44 45 41 44 47 41 35 40 33 16 4  3  6  10 4  2  3  10 17 10 6  4  4  4
4  2  2  4  6  5  2  2  3  4  3  2  2  5  17 33 45 46 43 39 42 42 38 37 35 42 47 45 41 44 37 25 14 5  10 16 8  2  3  6  10 4  2  3  4  4
4  2  2  3  5  5  3  3  3  4  3  2  4  17 32 44 47 48 41 35 40 40 35 41 44 46 47 47 46 34 22 26 27 17 8  10 4  2  3  4  3  2  2  4  4  4
5  3  2  3  5  6  4  4  4  4  3  4  16 33 43 41 47 50 45 36 43 45 43 51 51 44 46 47 42 27 17 32 43 33 18 5  2  3  3  4  3  2  3  5  5  4
4  5  8  5  4  5  3  4  3  4  6  16 33 47 42 36 42 49 42 34 38 45 48 50 44 37 42 46 40 33 28 37 43 42 29 12 4  4  4  4  4  3  2  4  4  4
2  8  15 9  4  3  3  4  4  5  18 34 48 51 47 41 44 47 45 39 39 42 45 47 45 42 46 40 30 35 33 36 38 39 33 19 10 6  4  4  5  4  3  3  5  5
7  12 15 9  8  9  9  10 9  15 30 44 48 47 45 44 44 46 50 50 47 45 44 44 44 45 46 37 25 27 29 33 38 37 33 27 20 13 7  4  4  3  2  4  6  7
18 20 18 13 15 17 18 20 26 30 37 40 42 39 35 40 43 48 53 57 55 49 42 41 44 43 43 38 28 27 28 34 35 32 29 27 24 19 11 6  3  2  2  3  5  5
8  11 10 15 17 18 19 24 35 38 43 46 45 38 33 39 45 47 51 54 55 50 43 43 47 47 47 37 28 30 36 41 42 38 32 34 35 28 19 7  3  3  3  3  4  4
3  2  8  20 26 26 25 29 37 36 44 51 49 44 36 43 45 47 48 48 44 46 45 43 41 44 42 28 18 27 39 45 48 42 33 36 43 39 28 11 4  4  5  5  4  4
3  2  4  10 13 13 11 18 31 40 44 44 46 42 36 41 47 46 50 45 37 42 49 42 35 40 45 34 21 20 32 45 48 44 32 36 43 44 30 11 3  3  3  3  4  4
4  2  2  3  4  4  3  4  17 33 40 37 44 46 41 45 44 39 38 42 36 46 52 48 42 46 49 46 33 17 25 36 46 42 34 38 44 33 17 6  4  3  2  3  4  4
4  2  2  3  4  5  5  3  5  17 32 42 50 50 49 48 44 31 28 31 35 41 44 47 49 48 44 47 38 20 19 29 36 44 42 43 33 16 5  4  5  4  3  3  4  4
5  3  2  3  4  4  3  2  3  5  18 34 47 48 48 46 36 30 29 37 40 40 34 38 47 42 36 42 40 24 22 20 25 40 46 33 16 4  3  4  4  3  3  3  4  4
6  7  10 12 10 9  8  8  9  10 12 24 38 44 45 39 32 35 45 48 49 46 36 35 41 42 38 42 41 35 29 20 19 35 36 20 5  2  3  4  3  3  4  4  5  5
7  12 22 24 19 16 16 16 17 18 19 21 30 38 41 40 38 45 49 54 53 48 43 38 39 40 39 36 32 35 32 21 20 30 24 12 6  4  4  5  5  3  3  4  6  7
4  7  11 12 10 9  8  8  9  10 10 9  15 30 43 45 46 44 42 47 50 48 46 45 45 44 44 38 32 37 35 18 15 21 14 6  3  2  4  8  7  3  2  3  5  5
4  4  3  3  4  5  2  2  3  4  3  2  4  17 33 46 47 45 42 45 47 46 47 47 47 47 47 45 41 43 39 21 8  9  6  3  2  2  3  8  12 4  2  3  4  4
3  3  2  4  12 15 7  7  4  4  3  2  2  5  17 33 43 43 42 42 41 42 43 44 43 43 44 44 41 36 34 23 9  3  4  3  2  2  3  10 17 8  2  3  4  4
3  2  2  8  21 24 14 12 8  4  3  2  2  3  5  16 29 40 42 38 29 28 35 37 34 36 33 31 30 27 17 10 4  3  4  3  2  2  3  6  10 4  2  3  4  4
3  2  2  4  11 12 6  7  4  4  4  3  2  3  4  5  14 31 45 39 20 20 29 32 29 32 25 16 23 22 10 2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
4  2  2  3  6  10 4  2  3  4  5  5  3  3  4  3  4  16 33 39 28 30 34 29 29 40 32 24 20 16 5  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
4  2  2  3  10 16 8  3  3  4  4  3  2  3  4  3  2  4  17 30 37 42 37 26 28 39 41 26 15 6  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
5  3  2  3  6  10 5  4  4  4  3  2  2  3  4  3  2  2  5  17 32 45 43 35 38 43 31 16 5  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
4  2  2  4  9  5  2  3  3  4  3  2  3  3  4  3  2  2  3  5  18 33 41 39 41 33 16 4  3  4  3  2  2  3  6  10 4  2  3  6  10 4  2  3  4  4
2  1  2  10 20 11 2  2  3  6  12 13 8  4  4  3  2  2  3  4  6  18 37 41 32 17 4  2  3  4  3  2  2  3  10 17 8  2  3  10 17 8  2  3  4  4
1  1  2  10 23 19 10 4  3  10 24 24 12 4  4  3  2  2  3  4  3  9  29 37 20 5  2  2  3  4  3  2  2  5  14 13 4  2  3  6  10 4  2  3  4  4
2  1  2  5  15 23 21 8  3  6  12 11 5  3  5  4  2  2  3  4  3  3  17 29 14 3  2  2  3  4  3  2  4  18 22 11 2  3  8  5  3  2  3  7  5  4
2  1  2  3  5  12 12 4  4  9  5  2  2  4  6  5  2  2  3  4  3  2  12 27 14 3  2  2  3  4  3  2  9  21 19 6  2  7  13 9  3  2  6  11 8  4
3  2  2  3  4  5  4  3  8  14 9  2  2  3  5  4  2  2  3  4  3  2  10 21 12 3  2  2  3  4  3  2  4  10 6  3  2  3  8  5  3  2  3  7  5  4
3  2  2  3  4  4  4  2  6  14 7  2  2  3  4  3  2  2  3  4  3  2  6  15 8  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4

```

```
46 46 0 1







                                              1
                                              1 1
                                                1
                                                1 1
                                        1         1 1
                                      1 1
                                    1 1 1     1
                                  1 1 1 1   1 1 1 1 1 1
                                  1 1 1 1 1 1 1 1 1 1 1   1
                                1 1 1 1 1 1 1     1 1 1 1 1
                              1 1 1 1   1 1   1 1 1 1 1 1
                            1 1 1 1 1   1 1 1 1 1 1 1 1 1       1
                          1 1   1 1 1   1 1 1 1 1   1 1 1       1 1
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1         1 1
                      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1           1
                      1 1 1   1 1 1 1 1 1 1 1 1 1 1 1 1
                  1 1 1 1 1   1 1 1 1 1 1 1 1 1 1 1 1         1 1 1
                    1 1 1 1   1 1 1 1 1 1 1 1 1 1 1 1       1 1 1 1     1 1
                  1 1 1 1 1   1 1 1 1 1   1 1 1   1 1         1 1 1     1 1
                    1   1 1 1 1 1 1 1 1   1 1 1 1 1 1 1         1 1   1 1
                      1 1 1 1 1 1         1 1 1 1 1 1 1 1         1 1 1
                        1 1 1 1         1 1   1 1 1   1 1         1 1
                        1 1 1 1     1 1 1 1     1 1 1 1 1
                          1 1 1 1 1 1 1 1 1 1 1 1 1 1
                            1 1 1 1 1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                                1 1 1 1 1 1 1 1 1 1 1 1 1
                                  1 1 1
                                    1 1
                                      1           1
                                          1       1 1
                                          1 1   1 1
                                            1 1 1
                                              1






```

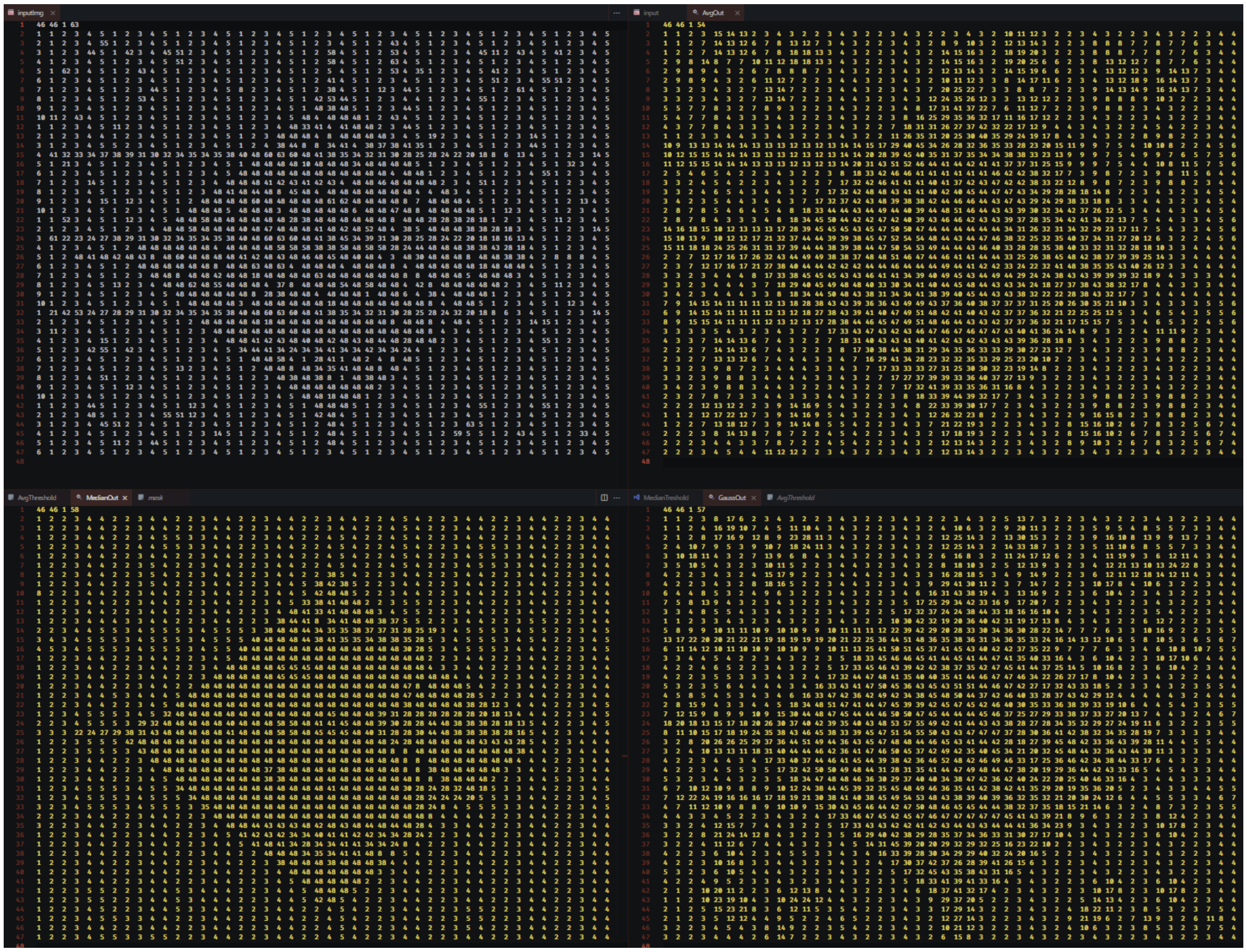
Comparison(Treshold)



```
46 46 0 1
```

```
46 46 0 1
```

```
46 46 0 1
```



```
46 46 1 63
```

```
46 46 1 54
```

```
46 46 1 58
```

```
46 46 1 57
```