Project 1: Histogram

Student: Simon Kong

Project Due date: 9/8/2021

Algorithm Steps:

****

I. main (...)

****

step 0: inFile ← open input file use args [0]

outFile1, outFile2, outFile3, outFile4 ← open using args[2] to args[5]

thrVal get from args[1]

step 1: numRows, numCols, minVal, maxVal ←read from inFile

step 2: histAry ← dynamically allocate and initialize to 0

step 3: ComputeHist (...)

step 4: printHist(outFile1)

Step 5: dispHist (outFile2)

step 6: close inFile

reopen inFile

Step 7: outFile3 "The threshold value uses is "

outFile3 ← thrVal

outFile4 ← "The threshold value uses is "

outFile4 ← thrVal

Step 8: threshold (inFile, outFile3, outFile4, thrVal)

step 9: close all files

****

II. printHist (output)

****

Step 0: Input <- given grey-scale image

Output <- open output files

Step 1: numRows, numCals, minVal, maxVal ← get from input

Hist[maxVal + 1] ←dynamically allocate array

Step 2: process the input file from Left →Right; Top→Bottom

Pixel P(x,y).value ←read from input

Hist[P(x,y).value]++

Step 3: repeat step 2 until the file is empty

Step 4: output ← histogram array to output file

Step 5: close input file and output file

****

II. dispHist (output)

****

Step 0: Input <- given grey-scale image

Output <- open output files

Step 1: numRows, numCals, minVal, maxVal ← get from input

Hist[maxVal + 1] ←dynamically allocate array

Step 2: process the input file from Left →Right; Top→Bottom

Pixel P(x,y).value ←read from input

Hist[P(x,y).value]++

Step 3: repeat step 2 until the file is empty

Step 4: output ← histogram array to output file

Step 5: append + to output by looping hist[index] times

Step 6: close input file and output file

****

III. threshold (inFile, outFile3, outFile4, thrVal)

****

Step 0: minVal ← 0

maxVal ← 1

Step 1: outFile3, outFile4 ← output numRows, numCols, minVal and maxVal

Step 2: pixelVal← read from inFile one integer at a time

Step 3: if pixelVal >= thrVal{

outFile3 ← write 1 follows by a blank

outFile4 ← write 1 follows by a blank

}else{

outFile3 ← write 0 follows by a blank

outFile4 ← write . follows by a blank

}

Step 4: repeat step 2 to 3 until the inFile is empty

Source Code

Main Class

```
package project_1;
import java.io.*;
public class Main{
    public static void main(String[] args) throws IOException {

        if(args.length != 6) {
            System.out.println("Invalid number of arguments.");
            System.exit(0);
        }

        try {
            Integer.parseInt(args[1]);
        }catch(Exception e){
            System.out.println("Invalid threshold.");
            System.exit(0);
        }

//Initialize variables
        String inputName = args[ 0 ];
        String threshold = args[ 1 ];
        String output_1 = args[ 2 ];
        String output_2 = args[ 3 ];
        String output_3 = args[ 4 ];
        String output_4 = args[ 5 ];
        FileReader inputReader = null ;
        BufferedReader buffInReader = null ;
        Scanner input = null ;
        FileWriter outputWriter = null ;
        BufferedWriter output = null ;
        FileWriter outputWriter_2 = null;
        BufferedWriter output2 = null;
```

```
        try{
//          Open input
            inputReader = new FileReader( inputName ) ;
            buffInReader = new BufferedReader( inputReader) ;
            input = new Scanner( buffInReader ) ;

//          Read input
            int numRows = 0 ;
            if( input.hasNextInt() ) numRows = input.nextInt() ;
            int numCols = 0 ;
            if( input.hasNextInt() ) numCols = input.nextInt() ;
            int minVal = 0 ;
            if( input.hasNextInt() ) minVal = input.nextInt() ;
            int maxVal = 0 ;
            if( input.hasNextInt() ) maxVal = input.nextInt() ;
            Image readObj = new Image( numRows, numCols, minVal, maxVal ) ;

            for( int i = 0 ; i < readObj.numRows ; ++i ) {
                for( int j = 0 ; j < readObj.numCols ; ++j ) {
                    if( input.hasNextInt() ) {
                        readObj.computeHist(input.nextInt());
                    }
                    else{ System.out.println( "Corrupted input data" ) ; System.exit(0); }
                }
            }

            readObj.printHist(output_1);
            readObj.dispHist(output_2);

//          Close input
            input.close();

//          Reopen input
            inputReader = new FileReader( inputName ) ;
            buffInReader = new BufferedReader( inputReader) ;
            input = new Scanner( buffInReader ) ;


//          Output 3
            outputWriter = new FileWriter( output_3 ) ;
            output = new BufferedWriter( outputWriter ) ;
            output.write("The threshold value used is " + threshold + "\n");
```

```
//          Output 4
            outputWriter_2 = new FileWriter(output_4);
            output2 = new BufferedWriter( outputWriter_2 );
            output2.write("The threshold value used is " + threshold + "\n");

            output.flush();
            output2.flush();
            readObj.threshold(inputName, output_3, output_4, threshold );

        }finally {
            if( input != null ) input.close() ;
            if( output != null) output.close();
            if( output2 != null) output2.close();
        }
    }
}
```

Image Class

Constructor

```
public class Image {
    public int numRows=0 , numCols=0 , minVal=0 , maxVal=0 ;
    public int[] histAry;

    public Image( int rows , int cols , int min , int max ){
        this.numRows = rows ;
        this.numCols = cols ;
        this.minVal = min ;
        this.maxVal = max ;
        this.histAry = new int[ this.maxVal + 1 ]; //initialized to zero by default
    }
```

compHist()

```
public void computeHist(int pixel){
    this.histAry[pixel]++;
}
```

printHist()

```
public void printHist(String output_name) throws IOException  {
    FileWriter outputWriter = null ;
    BufferedWriter output = null ;
    try {
        outputWriter = new FileWriter( output_name ) ;
        output = new BufferedWriter( outputWriter ) ;
        output.write( Integer.toString( this.numRows ) + " " ) ;
        output.write( Integer.toString( this.numCols ) + " " ) ;
        output.write( Integer.toString( this.minVal ) + " " ) ;
        output.write( Integer.toString( this.maxVal ) + "\n" ) ;
        for( int i = 0 ; i< this.histAry.length ; ++i ){
            output.write( Integer.toString(i) + " " + Integer.toString(this.histAry[i]) + "\n" );
        }

    }finally {
        if( output != null ) output.close() ;
    }

}
```

dispHist()

```
public void dispHist(String output_name) throws IOException {
    FileWriter outputWriter = null ;
    BufferedWriter output = null ;

    try {
        outputWriter = new FileWriter( output_name ) ;
        output = new BufferedWriter( outputWriter ) ;
        output.write( Integer.toString( this.numRows ) + " " ) ;
        output.write( Integer.toString( this.numCols ) + " " ) ;
        output.write( Integer.toString( this.minVal ) + " " ) ;
        output.write( Integer.toString( this.maxVal ) + "\n" ) ;
        for( int i = 0 ; i< this.histAry.length ; ++i ){
            output.write( Integer.toString(i) + " (" + Integer.toString(this.histAry[i]) + "):");
            for(int j = 0; j < this.histAry[i]; j++) {
                output.write("+");
            }
            output.write("\n");
        }

    }finally {
        if( output != null ) output.close();
    }
}
```

Threshold()

```
public void threshold(String input_name, String output_1, String output_2, String threshold) throws IOException {
    this.minVal = 0;
    this.maxVal = 1;

    FileWriter outputWriter = null ;
    FileWriter outputWriter_2 = null ;
    BufferedWriter bufferWriter = null ;
    BufferedWriter bufferWriter_2 = null;
    FileReader inputReader = null ;
    BufferedReader buffInReader = null ;
    Scanner input = null ;

    try {
        outputWriter = new FileWriter( output_1, true ) ;
        bufferWriter = new BufferedWriter( outputWriter ) ;

        outputWriter_2 = new FileWriter( output_2, true ) ;
        bufferWriter_2 = new BufferedWriter( outputWriter_2);

/       create output headers
        bufferWriter.write( Integer.toString( this.numRows ) + " " ) ;
        bufferWriter.write( Integer.toString( this.numCols ) + " " ) ;
        bufferWriter.write( Integer.toString(this.minVal) + " " ) ;
        bufferWriter.write( Integer.toString(this.maxVal) + "\n" ) ;
        bufferWriter_2.write( Integer.toString( this.numRows ) + " " ) ;
        bufferWriter_2.write( Integer.toString( this.numCols ) + " " ) ;
        bufferWriter_2.write( Integer.toString(this.minVal) + " " ) ;
        bufferWriter_2.write( Integer.toString(this.maxVal) + "\n" ) ;

/   read input
        inputReader = new FileReader( input_name ) ;
        buffInReader = new BufferedReader( inputReader) ;
        input = new Scanner( buffInReader ) ;

/   skip header
        for(int i = 0; i < 4; i++) {
            if( input.hasNextInt() ) input.nextInt() ;
        }


/   read pixels and write output
        for( int i = 0 ; i < this.numRows ; ++i ) {
            for( int j = 0 ; j < this.numCols ; ++j ) {
                if( input.hasNextInt() ) {
                    if( input.nextInt() >= Integer.parseInt(threshold) ) {
                        bufferWriter.write("1 ");
                        bufferWriter_2.write("1 ");
                    }else {
                        bufferWriter.write("0 ");
                        bufferWriter_2.write(". ");
                    }
                }
                else{ System.out.println( "Corrupted input data" ) ; System.exit(0); }
            }
            bufferWriter.write("\n");
            bufferWriter_2.write("\n");
        }


    }finally {
        if( input != null ) input.close();
        if( bufferWriter != null ) bufferWriter.close();
        if( bufferWriter_2 != null ) bufferWriter_2.close();

    }

}
```

Output 1

```
46 46 1 63
0 0
1 277
2 278
3 270
4 319
5 278
6 7
7 6
8 35
9 4
10 5
11 7
12 8
13 6
14 9
15 3
16 3
17 0
18 12
19 1
20 3
21 4
22 7
23 3
24 7
25 3
26 0
27 3
28 15
29 3
30 7
31 7
32 7
33 2
34 10
35 10
36 0
37 0
38 25
39 1
40 7
41 19
42 18
43 18
44 13
45 8
46 2
47 2
48 313
49 0
50 0
51 8
52 2
53 1
54 2
55 11
56 0
57 0
58 25
59 0
60 9
61 1
62 2
63 10
```

Output 2

46 46 1 63

0 (0):

1 (277):+++

2 (278):++

3 (270):++

4 (319):+++

5 (278):++

6 (7):+++++++

7 (6):++++++

8 (35):+++++++++++++++++++++++++++++++++++

9 (4):++++

10 (5):+++++

11 (7):+++++++

12 (8):++++++++

13 (6):++++++

14 (9):+++++++++

15 (3):+++

16 (3):+++

17 (0):

18 (12):++++++++++++

19 (1):+

20 (3):+++

21 (4):++++

22 (7):+++++++

23 (3):+++

24 (7):+++++++

25 (3):+++

26 (0):

27 (3):+++

28 (15):+++++++++++++++

29 (3):+++

30 (7):+++++++

31 (7):+++++++

32 (7):+++++++

33 (2):++

34 (10):++++++++++

35 (10):++++++++++

36 (0):

37 (0):

38 (25):+++++++++++++++++++++++++

39 (1):+

40 (7):+++++++

41 (19):+++++++++++++++++++

42 (18):++++++++++++++++++

43 (18):++++++++++++++++++

44 (13):+++++++++++++

45 (8):++++++++

46 (2):++

47 (2):++

48 (313):+++

49 (0):

50 (0):

51 (8):++++++++

52 (2):++

53 (1):+

54 (2):++

55 (11):+++++++++++

56 (0):

57 (0):

58 (25):+++++++++++++++++++++++++

59 (0):

60 (9):+++++++++

61 (1):+

62 (2):++

63 (10):++++++++++

Output 3

```
The threshold value used is 20
46 46 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 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 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 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 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 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 1 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 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 1 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 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 1 1 0 0 1 1 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 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 0 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 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 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 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 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
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 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 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 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 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 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
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 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 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 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 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 1 1 0 1 0 0 0 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 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 1 0 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 0 0 0 0 0 0 1 1 1 1 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 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 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 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 1 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 1 1 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 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 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 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 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
```

Output 4

```
The threshold value used is 20
46 46 0 1
```