********** **COVER PAGE** **********

Class: CV
Name: Loyd Flores
Project: Real Time Programming Exam – Question 1
Project Name: Deepest Concavity for automatic threshold selection
Language: Java
Due Date: 10/29/2024
Submit Date: 10/29/2024

********** SOURCE CODE **********

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

public class FloresL_Q1_Main {

    class Concavity{
        // HEADERS
        private int numRows;
        private int numCols;
        private int minVal;
        private int maxVal;

        // Points of Histogram && SLOPE
        private int x1;
        private int y1;
        private int x2;
        private int y2;
        private double m;   // slope
        private double b;   // intercept

        // HISTOGRAM
        private int bestThrVal; // auto selected
        private int histHeight; // Largest hist[i]

        // ARRAYS
        private int[] histAry;      // 1D (size of maxVal + 1) = zero
        private int[] lineAry;      // 1D (size of maxvVal +1) = zero
        private String[][] graphAry;   // 2D Char array ([maxVal + 1][histHeight + 1]) = blank

        public Concavity(int rows, int cols, int min, int max){
            this.numRows = rows;
            this.numCols = cols;
            this.minVal = min;
            this.maxVal = max;
            this.bestThrVal = 0;
            this.histHeight = 0;

            this.histAry = new int[maxVal + 1];
            this.lineAry = new int[maxVal + 1];

            this.histAry = new int[maxVal + 1];
            this.lineAry = new int[maxVal + 1];
            this.graphAry = new String[maxVal  + 1][histHeight + 1];
        }//end-constructor

        public int loadHist(String inFile){
            int maxHistVal = 0;
            try{
                BufferedReader readInfile = new BufferedReader(new FileReader(inFile));
                String[] header_inFile = readInfile.readLine().trim().split("\\s+");
                numRows = Integer.parseInt(header_inFile[0]);
                numCols = Integer.parseInt(header_inFile[1]);
                minVal = Integer.parseInt(header_inFile[2]);
                maxVal = Integer.parseInt(header_inFile[3]);

                String line;
                while ((line = readInfile.readLine()) != null) {
                    String[] parts = line.trim().split("\\s+");
                    int index = Integer.parseInt(parts[0]);
                    int value = Integer.parseInt(parts[1]);

                    // Check if index is within bounds
                    if (index >= 0 && index <= maxVal) {
                        histAry[index] = value;

                        // Keep track of the maximum histogram value
                        if (value > maxHistVal) {
                            maxHistVal = value;
                        }
                    } else {
                        System.err.println("Error: Index " + index + " out of bounds");
                    }
```

```
        } // end-while loop
    }catch (IOException e){
        System.err.println("ERROR: " + e.getMessage());
    }
    this.histHeight = maxHistVal;
    this.graphAry = new String[maxVal + 1][histHeight + 1];
    return maxHistVal;
}



public void printHist(int[] Histary, String logFileName) {
    try (BufferedWriter logFile = new BufferedWriter(new FileWriter(logFileName))) {
        logFile.write("** Below is the input histogram **\n");

        for (int i = 0; i <= maxVal; i++) {
            logFile.write(i + " " + Histary[i] + "\n");
        }

        logFile.write("\n");
    } catch (IOException e) {
        System.err.println("Error writing to file: " + e.getMessage());
    }
}


public void dispHist(String histGraphFileName) {
    try (BufferedWriter histGraphFile = new BufferedWriter(new FileWriter(histGraphFileName, true));) {

        histGraphFile.write("** Below is the graphic display with + of the input histogram ** \n");
        histGraphFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");

        for (int i = 0; i <= maxVal; i++) {
            histGraphFile.write(i + " (" + histAry[i] + "): ");
            for (int j = 0; j < histAry[i]; j++) {
                histGraphFile.write("+");
            }
            histGraphFile.write("\n");

        }


    } catch (IOException e) {
        System.err.println("Error writing to file: " + e.getMessage());
    }
} // end dispHist method


public int deepestConcavity(int x1, int y1, int x2, int y2, int[] histAry, int[] lineAry) {
    // Step 0
    double thr = 0;
    this.m = (double)(y2-y1) / (double)(x2 - x1);
    this.b = (double)y1 - (m * (double)x1);
    int maxGap = 0;
    double first = x1;
    double second = x2;
    double x = first;

    while (x <= second) {
        int y = (int)(m * x + b);
        lineAry[(int)x] = y;

        int gap = Math.abs(histAry[(int)x] - y);

        if (gap > maxGap) {
            maxGap = gap;
            thr = x;
        }


        x++;
    }

    this.bestThrVal = (int)thr;
    return (int)thr;
```

```
    }

    public void zero1Darray(int[] ary, int nRows){
        for (int i = 0; i < nRows; i++){
            ary[i] = 0;
        }
    }

    public void print1Darray(int[] ary, int nRows){
        for (int i = 0; i < nRows; i++){
            System.out.print(ary[i] + " ");
        }
    }

    public void blank2DAry(String[][] ary, int nRows, int nCols){
        for (int i = 0; i < ary.length; i++){
            for (int j = 0; j < ary[i].length; j++){
                ary[i][j] = "_";
            }
        }
    }

    public void print2Darray(String[][] ary, int nRows, int nCols){
        for (int j = 0; j < ary[0].length; j++) {
            System.out.printf("%2d ", j);
        }
        System.out.println();


        for (int i = 0; i < ary.length; i++){
            System.out.printf("%2d ", i);
            for (int j = 0; j < ary[i].length; j++){
                System.out.printf("%2s ", ary[i][j]);
            }
            System.out.println();
        }
    }

    public void printLine(int[] lineAry, String concavityFileName) {
        try (BufferedWriter concavityFile = new BufferedWriter(new FileWriter(concavityFileName))) {
            concavityFile.write("** Below is the computed line array **\n");

            for (int i = 0; i <= maxVal; i++) {
                concavityFile.write(i + " " + lineAry[i] + "\n");
            }

            concavityFile.write("\n");
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    public void plotGapGraph(int[] histAry, int[] lineAry, String[][] graphAry) {
        int index = 0;

        while (index <= maxVal) {
            for (int i = 0; i < histAry[index]; i++) {
                graphAry[index][i] = "+";
            }

            if (lineAry[index] > 0) {
                plotOneRowGap(index, histAry, lineAry, graphAry);
            }

            index++;
        }
    }

    public void plotOneRowGap(int index, int[] histAry, int[] lineAry, String[][] graphAry) {
        int j = histAry[index];
        while (j < lineAry[index]) {
            graphAry[index][j] = "=";
            j++;
        }

        if (lineAry[index] > 0 && lineAry[index] - 1 < graphAry[index].length) {
```

```
            graphAry[index][lineAry[index] - 1] = "@";
        }

        if (index == bestThrVal) {
            for (int k = 0; k < 4; k++) {
                if (lineAry[index] + k < graphAry[index].length) {
                    graphAry[index][lineAry[index] + k] = "<";
                }
            }
        }
    }

    public void outputGraphAry(String[][] graphAry, String concavityFileName) {
        try (BufferedWriter concavityFile = new BufferedWriter(new FileWriter(concavityFileName, true))) { // true for append mode
            concavityFile.write("\n** Below is the graphic display of histAry with + on the gaps with = and line pts with @**\n");

            concavityFile.write("    ");
            for (int j = 0; j < graphAry[0].length; j++) {
                concavityFile.write(String.format("%2d ", j));
            }
            concavityFile.write("\n");

            for (int i = 0; i < graphAry.length; i++) {
                concavityFile.write(String.format("%2d ", i));
                for (int j = 0; j < graphAry[i].length; j++) {
                    concavityFile.write(String.format("%2s ", graphAry[i][j]));
                }
                concavityFile.write("\n");
            }
            concavityFile.write("\n");

        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

}//end-concavity


public static void main(String[] args) {
    // STEP 0
    String inputHist = args[0];
    String twoPoints = args[1];
    String histFile = args[2];
    String concavityFile = args[3];

    int numRows = 0;
    int numCols = 0;
    int minVal = 0;
    int maxVal = 0;
    int x1 = 0;
    int y1 = 0;
    int x2 = 0;
    int y2 = 0;
    int histHeigh = 0;
    int bestThrVal;
    // STEP 1
    try{
        BufferedReader hist = new BufferedReader(new FileReader(inputHist));
        String[] header_hist = hist.readLine().trim().split("\\s+");
        numRows = Integer.parseInt(header_hist[0]);
        numCols = Integer.parseInt(header_hist[1]);
        minVal = Integer.parseInt(header_hist[2]);
        maxVal = Integer.parseInt(header_hist[3]);

        BufferedReader twopts = new BufferedReader(new FileReader(twoPoints));
        String[] header_twopts = twopts.readLine().trim().split("\\s+");

        x1 = Integer.parseInt(header_twopts[0]);
        y1 = Integer.parseInt(header_twopts[1]);
        x2 = Integer.parseInt(header_twopts[2]);
        y2 = Integer.parseInt(header_twopts[3]);
    }catch (IOException e){
        System.err.println("ERROR: " + e.getMessage());
    }
```

```
        System.out.println("Histogram Header: " + numRows + " " + numCols + " " + minVal + " " + maxVal );
        System.out.println("Two Points Header: " + x1 + " " + y1 + " " + x2 + " " + y2 + "\n");
        // Instantiate Concavity
        Concavity concavity = new FloresL_Q1_Main().new Concavity(numRows, numCols, minVal, maxVal);
        System.out.println("Zeroed out histAry");
        // Instantiate Histary
        //concavity.zero1Darray(concavity.histAry, numRows);
        //concavity.print1Darray(concavity.histAry, numRows);
        System.out.println();System.out.println();
        // Instantiate lineAry
        System.out.println("Zeroed out lineAry");
        concavity.zero1Darray(concavity.lineAry, numRows);
        //concavity.print1Darray(concavity.histAry, numRows);
        // Obtain Hist Height
        histHeigh = concavity.loadHist(inputHist);
        System.out.println();System.out.println();
        System.out.println("Hist height: " + histHeigh);
        // Instantiate graphAry
        concavity.blank2DAry(concavity.graphAry, numRows, numCols);
        //concavity.print2Darray(concavity.graphAry, numRows, numCols);

        // Step 2
        concavity.printHist(concavity.histAry, histFile);
        concavity.dispHist(histFile);

        // Step 3
        bestThrVal = concavity.deepestConcavity(x1, y1, x2, y2, concavity.histAry, concavity.lineAry);
        System.out.println("Best Thr: " + bestThrVal);
        concavity.printLine(concavity.lineAry, concavityFile);

        try (BufferedWriter concavityOutFile = new BufferedWriter(new FileWriter(concavityFile, true))) {
            concavityOutFile.write("** Below is the best threshold produced by the deepest concavity method  **\n" + bestThrVal + "\n");
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }

        // Step 4
        concavity.plotGapGraph(concavity.histAry, concavity.lineAry, concavity.graphAry);
        concavity.outputGraphAry(concavity.graphAry, concavityFile);
    }//end-main

}//end-FloresL_Q1_Main


/**
 COMPILE : javac FloresL_Q1_Main.java
 RUN     : java FloresL_Q1_Main hist1.txt hist1_2pts.txt histFile.txt concavityFile.txt
 */
```

********** **histFile for hist1 and hist1_2pts** **********

** Below is the input histogram **

0 10
1 14
2 17
3 20
4 22
5 31
6 28
7 33
8 45
9 56
10 70
11 90
12 120
13 150
14 192
15 210
16 192
17 172
18 132
19 100
20 89
21 78
22 42
23 20
24 18
25 10
26 9
27 8
28 8
29 7
30 6
31 5
32 4
33 4
34 6
35 8
36 10
37 12
38 22
39 26
40 40
41 45
42 72
43 80
44 90
45 100
46 120
47 150
48 188
49 190
50 170
51 140
52 120
53 110
54 90
55 80
56 70
57 60
58 30
59 20
60 12
61 9
62 8
63 6

** Below is the graphic display with + of the input histogram **

```
64 64 0 63
0 (10): ++++++++++
1 (14): ++++++++++++++
2 (17): +++++++++++++++++
3 (20): ++++++++++++++++++++
4 (22): ++++++++++++++++++++++
5 (31): +++++++++++++++++++++++++++++++
6 (28): ++++++++++++++++++++++++++++
7 (33): +++++++++++++++++++++++++++++++++
8 (45): +++++++++++++++++++++++++++++++++++++++++++++
9 (56): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
11 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
12 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
13 (150): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
14 (192): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
15 (210): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
16 (192): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
17 (172): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
18 (132): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
19 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20 (89): +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
21 (78): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
22 (42): ++++++++++++++++++++++++++++++++++++++++++
23 (20): ++++++++++++++++++++
24 (18): ++++++++++++++++++
25 (10): ++++++++++
26 (9): +++++++++
27 (8): ++++++++
28 (8): ++++++++
29 (7): +++++++
30 (6): ++++++
31 (5): +++++
32 (4): ++++
33 (4): ++++
34 (6): ++++++
35 (8): ++++++++
36 (10): ++++++++++
37 (12): ++++++++++++
38 (22): ++++++++++++++++++++++
39 (26): ++++++++++++++++++++++++++
40 (40): ++++++++++++++++++++++++++++++++++++++++
41 (45): +++++++++++++++++++++++++++++++++++++++++++++
42 (72): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
43 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
44 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
45 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
46 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
47 (150): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
48 (188): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
49 (190): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
50 (170): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
51 (140): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
52 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
53 (110): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
54 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
55 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
56 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
57 (60): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
58 (30): ++++++++++++++++++++++++++++++
59 (20): ++++++++++++++++++++
60 (12): ++++++++++++
61 (9): +++++++++
62 (8): ++++++++
63 (6): ++++++
```

****************************** concavityFile for hist1 and hist1_2pts *******************************

** Below is the computed line array **
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 210
16 209
17 208
18 208
19 207
20 207
21 206
22 205
23 205
24 204
25 204
26 203
27 202
28 202
29 201
30 201
31 200
32 200
33 199
34 198
35 198
36 197
37 197
38 196
39 195
40 195
41 194
42 194
43 193
44 192
45 192
46 191
47 191
48 190
49 190
50 0
51 0
52 0
53 0
54 0
55 0
56 0
57 0
58 0
59 0
60 0
61 0
62 0
63 0

** Below is the best threshold produced by the deepest concavity method **
32

**** Below is the graphic display of histAry with + on the gaps with = and line pts with @****

Output on the doc is unreadable but I verified with the professor and she said I should get credit. I'll also attach a screenshot copy of the output