Student: Michael Grossman

Project Due Date: 02/21/2022

**Algorithms**

Algorithm Steps for Computing the deepestConcavity given a peak starting point (x1, y1), a 1D histogram representation histAry, a slope m to the second peak point (x2, y2), the y-intercept b for that line, and a 2d-array displayGraph to store the results:

1. Max ← 0
2. First ← x1
3. Second ← x2
4. X ← First
5. Thr ← First
6. While X <= Second:
7. Y ←m\*X + b
8. PlotOneRow(x, y, displayGraph)
9. Gap ←
10. If Gap > Max:
11. Max ←Gap
12. Thr ← X
13. End-If
14. X++
15. End-While-Loop
16. Return Thr

Algorithm Steps for PlotOneRow given an X value, Y value, and a 2D-array representation of the deepest concavity operations displayGraph, and a 1D array representation of a histogram histAry:

1. Index ←
2. Last ←
3. While Index <= Last:
4. displayGraph[X][Index] ← 3
5. Index++
6. End-While-Loop
7. displayGraph[X][histAry[X]] ← 1
8. displayGraph[X][Last] ← 2

**Main.cpp**

#include <iostream>

#include <fstream>

using namespace std;

class Concavity{

    public:

        //variables

        int numRows, numCols, maxVal, minVal; //image header specs

        int x1, y1, x2, y2; //histogram peaks

        double m, b; //the slope and y intercept of between the peaks

        int\* histAry; //stores the histogram for the image

        int maxHeight; //max height found in histAry

        int bestThrVal; //auto selected threshold value

        int\*\* displayGraph; //a graph representing our deepest concavity

        //constructor + destructor

        Concavity(int\* points, ifstream& input);

        ~Concavity();

        //class functions

        int loadHist(ifstream& input); //returns the maxHeight

        void printHist(ofstream& output); //prints histogram to output file

        int deepestConcavity(); //returns a proposed threshold value

        void plotOneRow(int x, int y, int\*\* display);

        void printGraph(ofstream& output); //print 2d graph after work is done

        int digit\_count(int input); //counts digits in a number

};

int main(int argc, char\*\* argv){

    //extract input and output files from command line arguments

    ifstream inFile1(argv[1]), inFile2(argv[2]);

    ofstream outFile1(argv[3]);

    //read in the first line of the points file where even indices are x's

    //and odd are y's

    int coords[4] = {0};

    for(int i = 0; i < 4; ++i){

        inFile2 >> coords[i];

    }

    //main algorithm, given a bimodal histogram and the peaks find the best

    //threshold value, graph everything, and output everything.

    Concavity concavity(coords, inFile1);

    concavity.printHist(outFile1);

    concavity.bestThrVal = concavity.deepestConcavity();

    outFile1 << "\nProposed best threshold value: " << concavity.bestThrVal;

    outFile1 << "\n";

    concavity.printGraph(outFile1);

    //close files

    inFile1.close();

    inFile2.close();

    outFile1.close();

    return 0;

}

//constructor, gives points of the peaks, calculates the slope and the

//intercept, calculates loads the histogram data and calculates the max

//histogram height. Initializes the displayGraph.

Concavity::Concavity(int\* points, ifstream& input){

    //adds in points where index 2\*i is x\_(i+1) and index 2\*i+1 is is y\_(i+1)

    x1 = points[0];

    y1 = points[1];

    x2 = points[2];

    y2 = points[3];

    //find the slope of the line between points 1 and 2

    m = 1.0\*(y2 - y1)/(x2 - x1);

    //finding the y-intercept, (y - y1) = m(x - x1) ->

    //y = m\*x - m\*x1 + y1 -> b = -m\*x1 + y1

    b = 1.0\*(-m\*x1 + y1);

    //loads the histogram and finds the max value stored

    maxHeight = loadHist(input);

    //initializes a dynamic array to hold the display for deepest

    //concavity

    int max1 = maxVal+1, max2 = maxHeight+1;

    displayGraph = new int\*[max1];

    for(int i = 0; i < max1; ++i){

        displayGraph[i] = new int[max2]();

    }

}

//deconstructor

Concavity::~Concavity(){

    //delete all dynamically allocated memory

    int max = maxVal+1;

    for(int i = 0; i < max; ++i){

        delete[] displayGraph[i];

    }

    delete[] displayGraph;

    delete[] histAry;

}

int Concavity::loadHist(ifstream& input){

    //load the first line / first four tokens which correspond to

    //this format

    input >> numRows;

    input >> numCols;

    input >> minVal;

    input >> maxVal;

    //allocate space for the array, initializing all to 0 and then

    //placing as we read

    histAry = new int[maxVal+1]();

    int index = 0, max = -1;

    for (int i = 0; i < maxVal; i++){

        input >> index;

        input >> histAry[index];

        //search for the max value stored in the histogram array

        if(histAry[i] > max) max = histAry[i];

    }

    //return the max value

    return max;

}

//print a visual 2d representation of the 1d histogram to a given file

void Concavity::printHist(ofstream& output){

    output << "Histogram Representation: \n";

    output << numRows << " " << numCols << " " << minVal

             << " " << maxVal << "\n";

    int arr\_end = maxVal + 1;

    //variables for use in generalizing the spacing

    int digits1 = digit\_count(maxVal), digits2 = digit\_count(maxHeight);

    int difference = 0;

    //loop through the histogram array

    for(int i = 0; i < arr\_end; ++i){

        output << i << " ";

        //generalized spacing between the index and count

        difference = digits1 - digit\_count(i);

        for(int j = 0; j < difference; ++j){

            output << " ";

        }

        output << "(" << histAry[i] << ") ";

        //generalized spacing between the count and it's representation

        difference = digits2 - digit\_count(histAry[i]);

        for(int j = 0; j < difference; ++j){

            output << " ";

        }

        output << ":";

        //represent the count in 2d

        for(int j = 0; j < histAry[i]; ++j){

            output << "+";

        }

        output << "\n";

    }

    //spec was changed from using the printHist func in project1 to the

    //dispHist function

    /\*

    output << numRows << " " << numCols << " " << minVal

            << " " << maxVal << "\n";

    int arr\_end = maxVal + 1, max\_space = digit\_count(maxVal);

    int current\_digits = 1 check = 10, calc = 0;

    for(int i = 0; i < arr\_end; ++i){

        output << i;

        if(i == check){

            check \*= 10;

            ++current\_digits;

        }

        calc = max\_space - current\_digits + 1;

        for(int j = 0; j < calc; ++j){

            output << " ";

        }

        output << histAry[i] << "\n";

    }

    \*/

}

//plots the concavity chart, finds the largest gap between line and

//histogram value, and returns thex value for that spot

int Concavity::deepestConcavity(){

    //initialize variables

    int max = 0, first = x1, second = x2, x = first, y, thr = first, gap = 0;

    //climb the line from peak to peak starting at the first peak

    //and calculate the distance to the histogram value at the spot

    while(x <= second){

        y = (int)(m\*x+b);

        //plots the information in the 2d representation of deepest

        //concavity calculation

        plotOneRow(x, y, displayGraph);

        //gap = |hist[x] - y| without the use of another library

        gap = (histAry[x] > y) ? histAry[x] - y : y - histAry[x];

        //keep track of the largest gap found between histogram values

        //and the line

        if (gap > max){

            max = gap;

            thr = x;

        }

        ++x;

    }

    return thr;

}

//adds one row to the 2d dynamic array representing the deepest concavity

//calculation

void Concavity::plotOneRow(int x, int y, int\*\* display){

    //determine if the line is above or below the current histogram value

    int index = (histAry[x] < y) ? histAry[x] : y;

    int last = (histAry[x] > y) ? histAry[x] : y;

    //record the space between the values in the 2d representation

    while(index <= last){

        displayGraph[x][index] = 3;

        ++index;

    }

    //record where the histogram and line points are for this row in the 2d

    //representation

    displayGraph[x][histAry[x]] = 1;

    displayGraph[x][last] = 2;

}

//prints a 2d graph representing the deepest concavity calculation to a given

//output file

void Concavity::printGraph(ofstream& output){

    //initialize variables and label graph

    int max1 = maxVal+1, max2 = maxHeight+1;

    output << "\nDeepest Concavity Graph: \n";

    //print the graph to the output file row by row with different values

    //depending on the points relation to the deepest concavity calculation

    for(int i = 0; i < max1; ++i){

        for(int j = 0; j < max2; ++j){

            if(displayGraph[i][j] == 0) output << " ";

            else if(displayGraph[i][j] == 1) output << "\*";

            else if(displayGraph[i][j] == 2) output << "+";

            else output << "=";

        }

        output << "\n";

    }

}

//helper function that counts the amount of digits in a number

//used to generalize spacing for prettier outputs

int Concavity::digit\_count(int input){

    int num\_digits = 0, temp = input;

    while(temp > 0){

        num\_digits++;

        temp /= 10;

    }

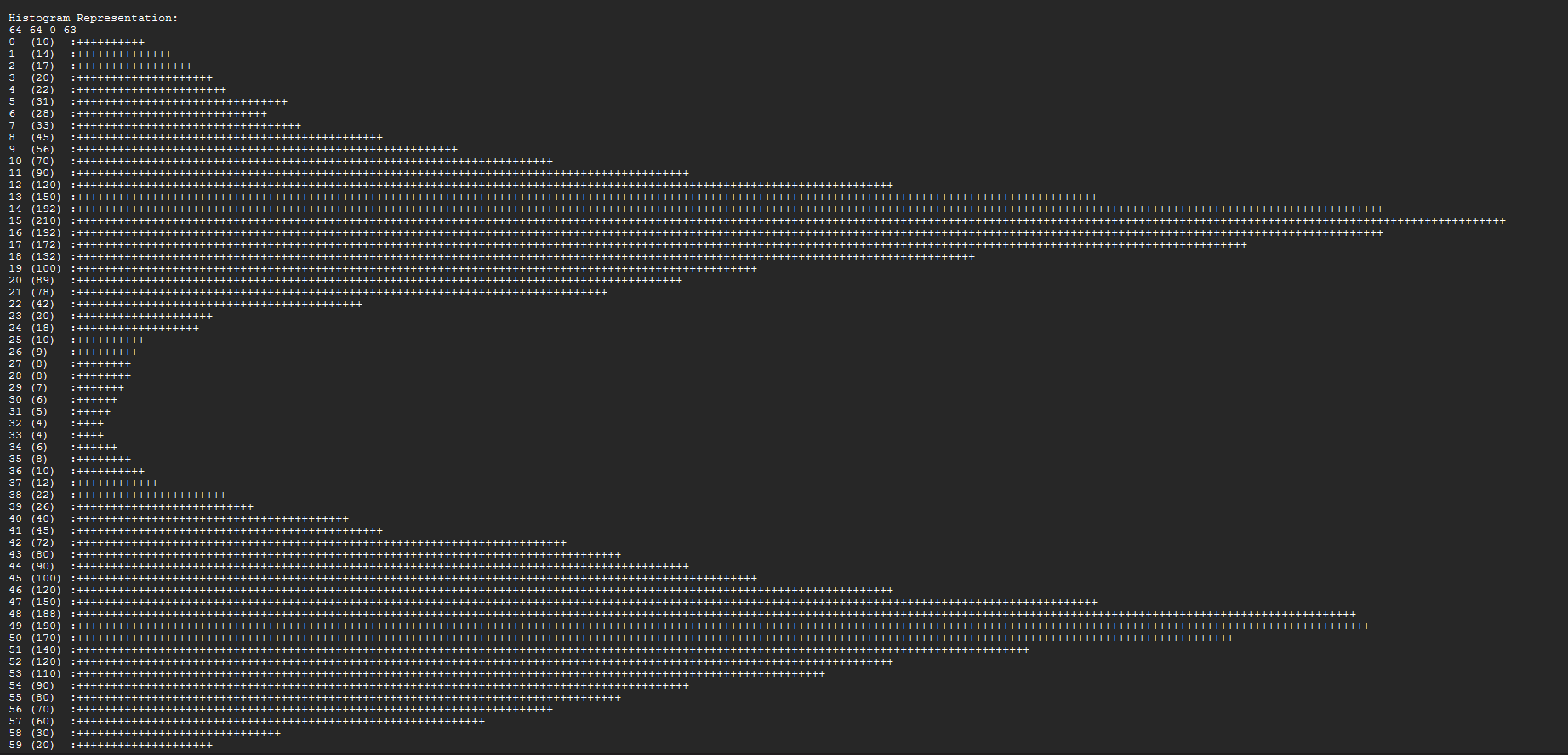
    //if num\_digits is 0 then we were given a 0, which has 1 digit

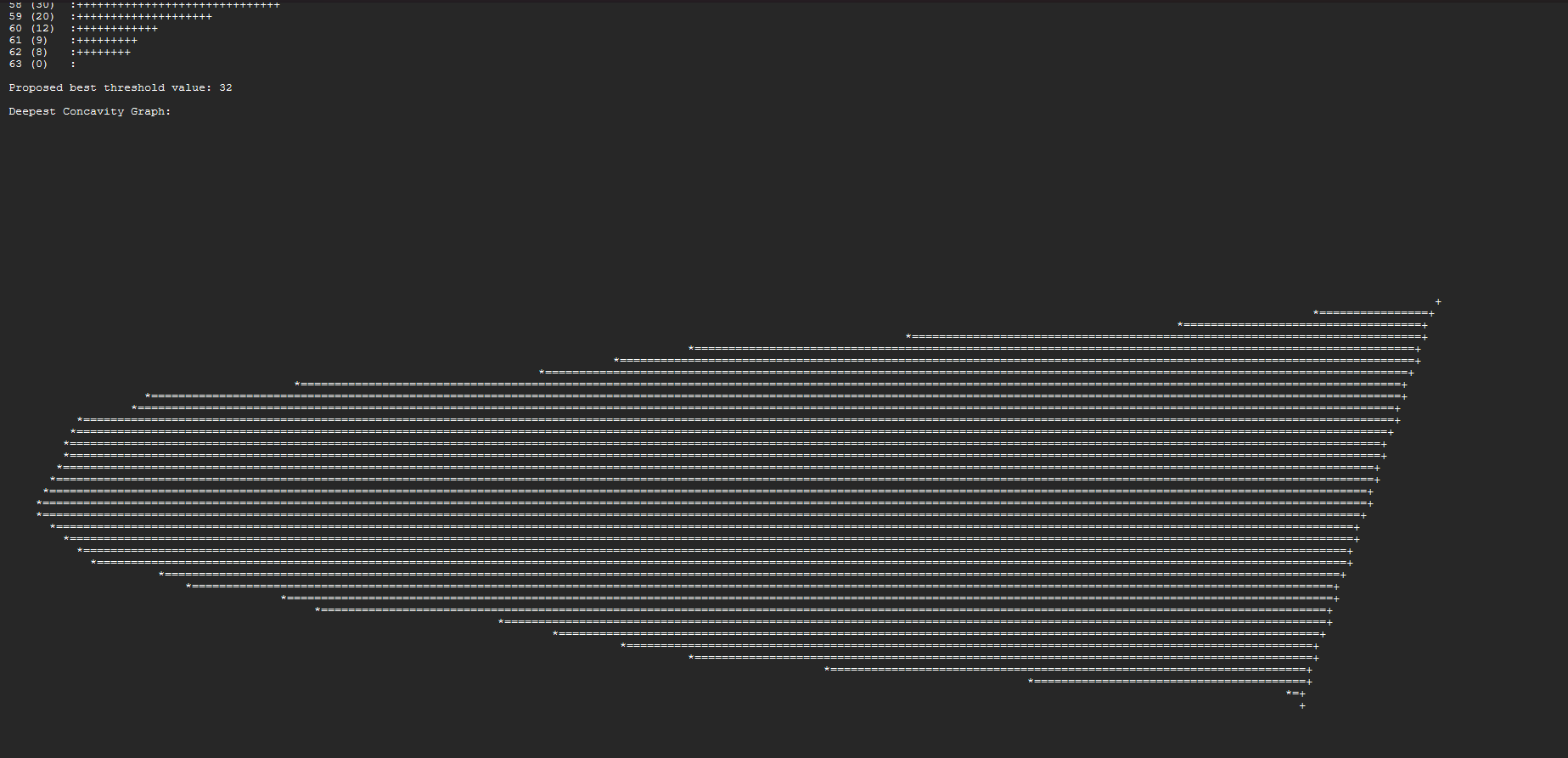
    if(num\_digits == 0) return 1;

    return num\_digits;

}

**Output for data1**





**Output for Data2**

