

**Student:** Pawan Bhatta

**Project Due Date:** 02/14/2021

### **Algorithm for Computing Histogram:**

- Step 1: Input ← given gray-scale image (input file)  
Output ← open output files (for histogram)
- Step 2: numRows, numCols, minVal, maxVal ← get from input  
hist[maxVal] ← dynamically allocate the hist array and initialize to 0
- Step 3: Process the input file from Left→Right , Top→Bottom  
Pixel P(x, y).value ← read from input  
hist[P(x, y).value] ++
- Step 4: Repeat step 3 until the file is empty
- Step 5: Output ← histogram array to output file
- Step 6: Close input file and output file

### **Algorithm for Applying Threshold Operation:**

- Step 1: minVal ← 0  
maxVal ← 1
- Step 2: outFile3, outFile4 ← output numRows, numCols, minVal, and maxVal
- Step 3: pixelVal ← read from inFile one integer at a time
- Step 4: if pixelVal ≥ thrVal  
    outFile3 ← write "1 "  
    outFile4 ← write "1 "  
else  
    outFile3 ← write "0 "  
    outFile4 ← write ". "
- Step 5: Repeat step 3 to 4 until the inFile is empty

## Source Code:

```
#include <iostream>
#include <fstream>
using namespace std;

const int MAX_PLUS = 70;

int toInt(string input)
{
    return stoi(input);
}

string getPlus(int numberOfPlus)
{
    string returnVal = "";
    if (numberOfPlus > MAX_PLUS)
    {
        numberOfPlus = MAX_PLUS;
    }
    for (int k = 0; k < numberOfPlus; k++)
    {
        returnVal = returnVal + "+";
    }
    return returnVal;
}

class Image
{
public:
    int numRows, numCols, minVal, maxVal;
    int *histAry;
    int thresholdValue;

    void computeHist(ifstream &input)
    {
        for (int i{0}; i < this->numRows; ++i)
        {
            for (int j{0}; j < this->numCols; ++j)
            {
                int pixelValue;
                input >> pixelValue;
                histAry[pixelValue]++;
            }
        }
    }

    void printHist(ofstream &output)
    {

```

```

        output << this->numRows << " " << this->numCols << " " << this->minVal << " " <<
this->maxVal << endl;

        for (int i = 0; i < maxVal + 1; ++i)
        {
            output << i << " " << histAry[i] << endl;
        }
    }

    void dispHist(ofstream &output)
    {
        output << numRows << " " << numCols << " " << minVal << " " << maxVal << " " <<
endl;
        for (int i = 0; i < maxVal + 1; ++i)
        {
            output << i << " "
                << "(" << histAry[i] << ") "
                << ": " << getPlus(histAry[i]) << endl;
        }
    }

    void threshold(ifstream &input, ofstream &output3, ofstream &output4, int thrVal)
    {
        Image binaryImg;
        binaryImg.minVal = 0;
        binaryImg.maxVal = 1;
        int anon;
        input >> binaryImg.numRows >> binaryImg.numCols >> anon >> anon;

        output3 << binaryImg.numRows << " " << binaryImg.numCols << " " << binaryImg.minVal
<< " " << binaryImg.maxVal << " " << endl;
        output4 << binaryImg.numRows << " " << binaryImg.numCols << " " << binaryImg.minVal
<< " " << binaryImg.maxVal << " " << endl;

        for (int i{0}; i < this->numRows; ++i)
        {
            for (int j{0}; j < this->numCols; ++j)
            {
                int pixelValue;
                input >> pixelValue;
                if (pixelValue >= thrVal)
                {
                    output3 << "1 ";
                    output4 << "1 ";
                }
                else
                {
                    output3 << "0 ";

```

```

        output4 << ". ";
    }
}
output3 << endl;
output4 << endl;
}
};

};

int main(int argc, const char *argv[])
{
    //READ
    string inputName = argv[1]; //(1) get the input file
    ifstream input;           //(2) instantiate the input
    input.open(inputName);     //(3) tell the input object

    //WRITES
    string outputName1{argv[3]}, outputName2{argv[4]}, outputName3{argv[5]},
outputName4{argv[6]};
    ofstream output1, output2, output3, output4;
    output1.open(outputName1);
    output2.open(outputName2);
    output3.open(outputName3);
    output4.open(outputName4);

    if (input.is_open())
    {

        if (output1.is_open() && output2.is_open() && output3.is_open() &&
output4.is_open())
        {
            Image img;
            input >> img.numRows >> img.numCols >> img.minVal >> img.maxVal;
            img.histAry = new int[img.maxVal + 1](); //dynamically allocated and
initialized to zero
            img.computeHist(input);
            img.printHist(output1);
            img.dispHist(output2);
            input.close();
            input.open(inputName);
            int thrVal = toInt(argv[2]);
            output3 << "The threshold value uses is " << thrVal << endl;
            output4 << "The threshold value uses is " << thrVal << endl;
            img.threshold(input, output3, output4, thrVal);
        }
        else
        {
            cout << "Error: Some output files couldnt be opened" << endl;
        }
    }
}

```

```
else
{
    cout << "Error: " << inputName << endl;
};

output1.close();
output2.close();
output3.close();
output4.close();
return 0;
}
```

Output outFile1 for data 1:

```
31 40 0 9
0 309
1 288
2 194
3 64
4 0
5 2
6 12
7 106
8 124
9 141
```

Output outFile2 for data 1:

```
31 40 0 9
0 (309) : +++++
1 (288) : +++++
2 (194) : +++++
3 (64)  : +++++
4 (0)   :
5 (2)   : ++
6 (12)  : +++++
7 (106) : +++++
8 (124) : +++++
9 (141) : +++++
```

Output outFile3 for data 1:

The threshold value uses is 5

31 40 0 1

[illegible]

Output outFile4 for data 1:

The threshold value uses is 5

31 40 0 1

[illegible]

Output outFile1 for data 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 outFile2 for data 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) :
```

46 46 0 1

Output outFile4 for data 2:

The threshold value uses is 38

46 46 0 1

[illegible]