Student: Simon Kong

Project Due date: 10/2/2021

Algorithm Steps:

Main

```
step 0: inFile, outFile1, outFile2 ← open via argv[]
step 1: numRows, numCols, minVal, maxVal ← read from inFile
step 2: offSet ← (int) (maxVal - minVal) / 10
        dividePt ← offSet
step 3: dynamically allocate histAry and GaussAry, and initialized to zero
        maxHeight ← loadHist (histAry, inFile)
Step 4: dynamically allocate all other arrays and initialized to zero
step 5: plotHistGraph (histGraph)
step 6: prettyPrint (histGraph, outFile1) // with caption
step 7: bestThrVal ← biMeanGauss (dividePt, outFile2)
        outFile1 ← output bestThrVal to outFile1 // with caption
step 8: bestFitPlot (bestThrVal) // plotting the result of Gaussian curves
prettyPrint(GaussGraph, outFile1) // with caption
step 9: prettyPrint(gapGraph, outFile1) // with caption
step 10: close all files
```

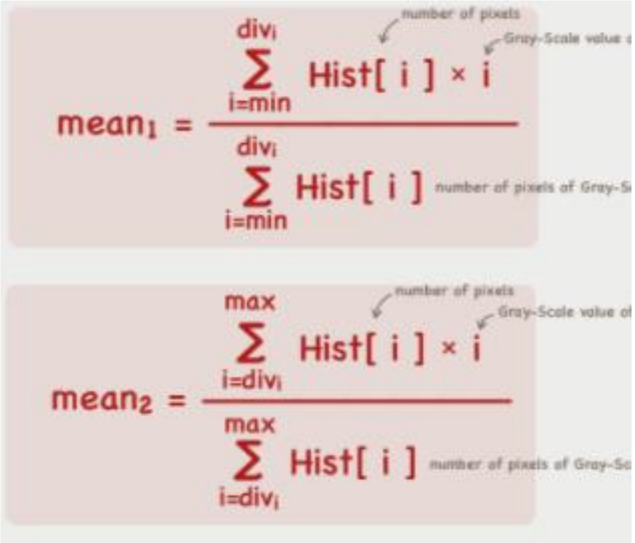
Bi-Mean Gauss

```
Step 0: (double) sum1
        (double) sum2
        (double) total
        (double) minSumDiff
        bestThr <- dividePt
        minSumDiff <- 999999.0 // a large value
Step 1: set1DZero (GaussAry) // reset for next computation
        set2DZero (GaussGraph)
        set2DZero (gapGraph)
step 2: sum1 <- fitGauss (0, dividePt, GaussAry, GaussGraph)
        // fitting the first Gaussian curve
Step 3: sum2 <- fitGauss (dividePt, maxVal, GaussAry, GaussGraph)
        // fit the second Gaussian curve
Step 4: total <- sum1 + sum2
        outFile2 <- print sum1, sum2, total with caption
Step 5: if total < minSumDiff
        minSumDiff <- total
        bestThr <- dividePt
Step 6: outFile2 <- print dividePt, minSumDiff and bestThr with caption
Step 7: dividePt ++
Step 8: prettyPrint (GaussGraph, outFile2)
Step 9: plotGaps (histAry, GaussGraph, gapGraph)
        prettyPrint (gapGraph, outFile3)
step 10: repeat step 1 to step 9 while dividePt < (maxVal – offSet)
step 11: return bestThr
```

fitGauss

```
Step 0: (double) mean
        (double) var
        (double) sum
        (double) Gval
        (double) maxGval
        sum ← 0.0
step 1: mean ← computeMean (leftIndex, rightIndex, maxHight)
        var ← computeVar (leftIndex, rightIndex, mean )
        outFile2← write leftIndex, rightIndex, mean, var with captions
Step 2: index ← leftIndex
Step 3: Gval ← modifiedGauss (x, mean, var, maxHight)
Step 4: sum += abs (Gval – (double)histAry[index])
Step 5: GaussAry[index] ← (int) Gval
        GaussGraph[index][(int) Gval] ← 1
Step 6: index ++
Step 7: repeat step 3 – step 6 while index <= rightIndex
Step 8: return sum
```

computeMean



$$\mathrm{mean}_1 = \frac{\sum_{i=min}^{div_i} \mathrm{Hist}[\ i\ ] \times i}{\sum_{i=min}^{div_i} \mathrm{Hist}[\ i\ ]}$$

number of pixels

Gray-Scale value o

number of pixels of Gray-S

$$\mathrm{mean}_2 = \frac{\sum_{i=div_i}^{max} \mathrm{Hist}[\ i\ ] \times i}{\sum_{i=div_i}^{max} \mathrm{Hist}[\ i\ ]}$$

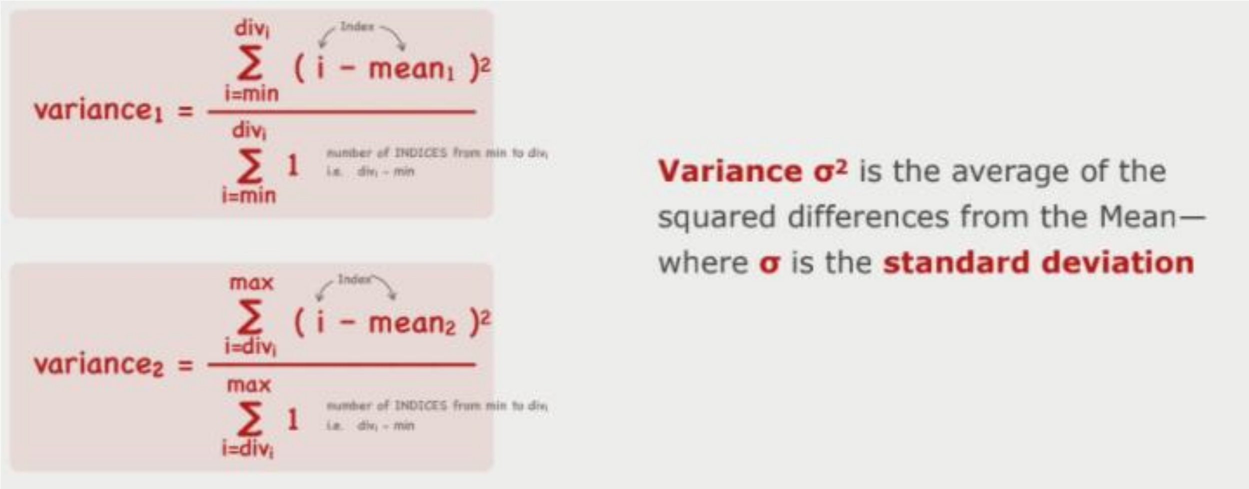
number of pixels

Gray-Scale value of

number of pixels of Gray-Sc

```
Step 0: maxHeight ← 0 // maxHight came via parameter, it is NOT local variable!
        sum ← 0
        numPixels ← 0
Step 1: index ← leftIndex
Step 2: sum += (hist[index] * index)
        numPixels += hist[index]
Step 3: if hist[index] > maxHeight
        maxHeight ← hist[index]
Step 4: index++
Step 5: repeat Step 2 to step 4 while index < rightIndex
Step 6: return (double)sum / (double) numPixels
```

computeVar



$$\text{variance}_1 = \frac{\sum_{i=\min}^{\text{div}_i} (\overset{\text{Index}}{i} - \text{mean}_1)^2}{\sum_{i=\min}^{\text{div}_i} 1}$$

number of INDICES from min to div$_i$ i.e. div$_i$ – min

$$\text{variance}_2 = \frac{\sum_{i=\text{div}_i}^{\max} (\overset{\text{Index}}{i} - \text{mean}_2)^2}{\sum_{i=\text{div}_i}^{\max} 1}$$

number of INDICES from min to div$_i$ i.e. div$_i$ – min

Variance σ² is the average of the squared differences from the Mean—where **σ** is the **standard deviation**

```
Step 0: sum □ 0.0
        numPixels □ 0
Step 1: index □ leftIndex
Step 2: sum += (double) hist [index] * ((double) index – mean)^2)
        numPixels += hist[index]
Step 3: index++
Step 4: repeat Step 2 to step 3 while index < rightIndex
Step 5: return (double) sum / (double) numPixels
```

ModifiedGauss

$$\text{Gaussian}_1(\ i\ ) = \text{height}_1 \times e^{-\frac{1}{2}\frac{(\ i-\text{mean}_1)^2}{\text{variance}_1}}$$

```
return (double) (maxHeight * exp( - ( (x-mean)^2 / (2*var) )
```

bestFitPlot

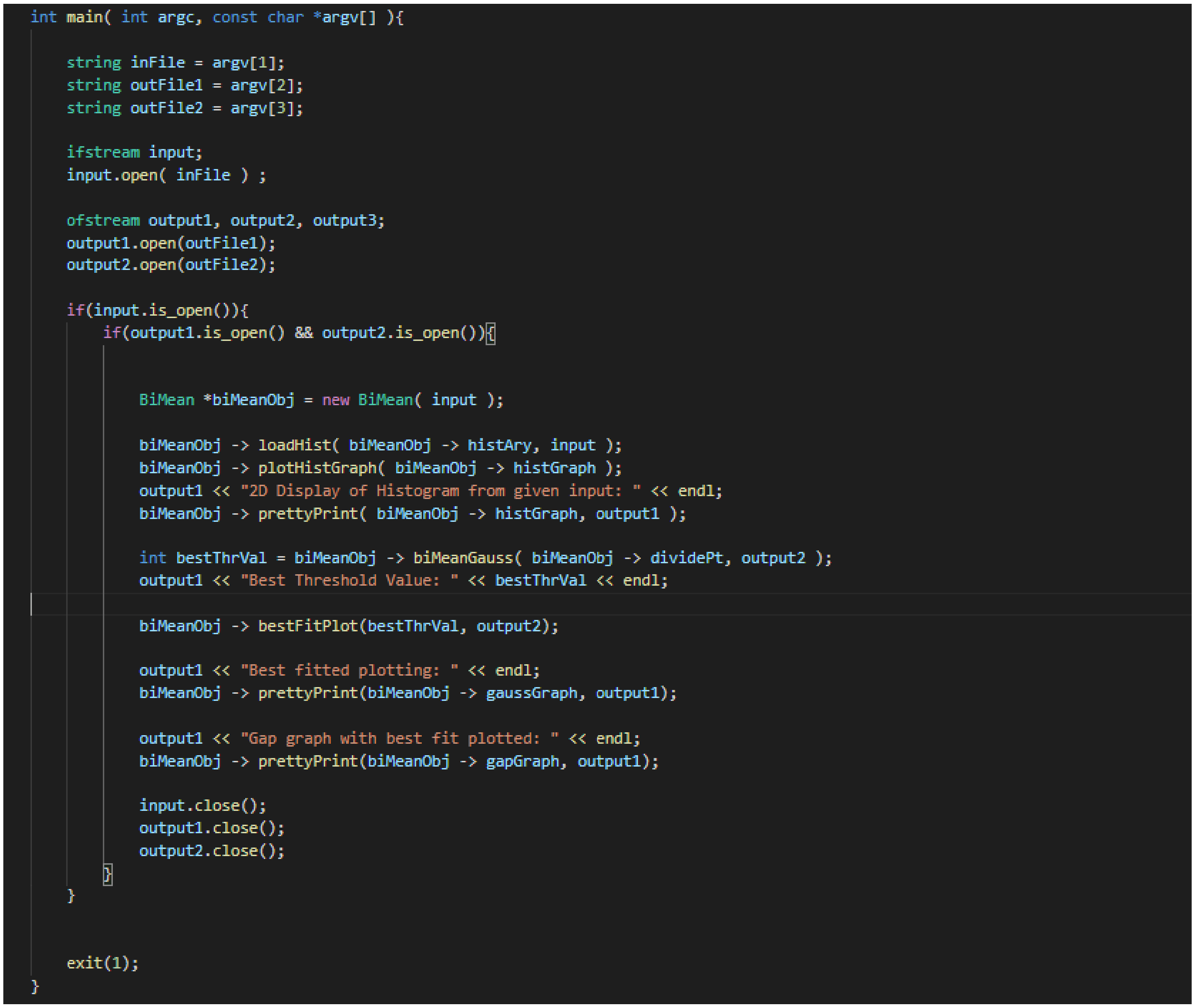
```
step 0: sum1 (double), sum2 (double)
Step 1: set1DZero(GaussAry)
        set2DZero(GaussGraph)
        set2DZero(gapGraph)
step 2: sum1 ← fitGauss(0, bestThrVal, GaussAry, GaussGraph)
Step 3: Sum2 ← fitGauss(bestThrVal, maxVal, GaussAry, GaussGraph)
Step 4: plotGaps(histAry, GaussGraph, gapGraph)
```

plotGaps

```
step 1: index ← minVal
step 2: first ← min(histAry[index], GaussAry[index])
        last ← max(histAry[index], GaussAry[index])
Step 3: gapGraph[index][first] ← 1
Step 4: first ++
Step 5: repeat step 3 to step 4 while first < last
Step 6: index ++
Step 7: repeat step 2 – step 6 while index < maxVal
```

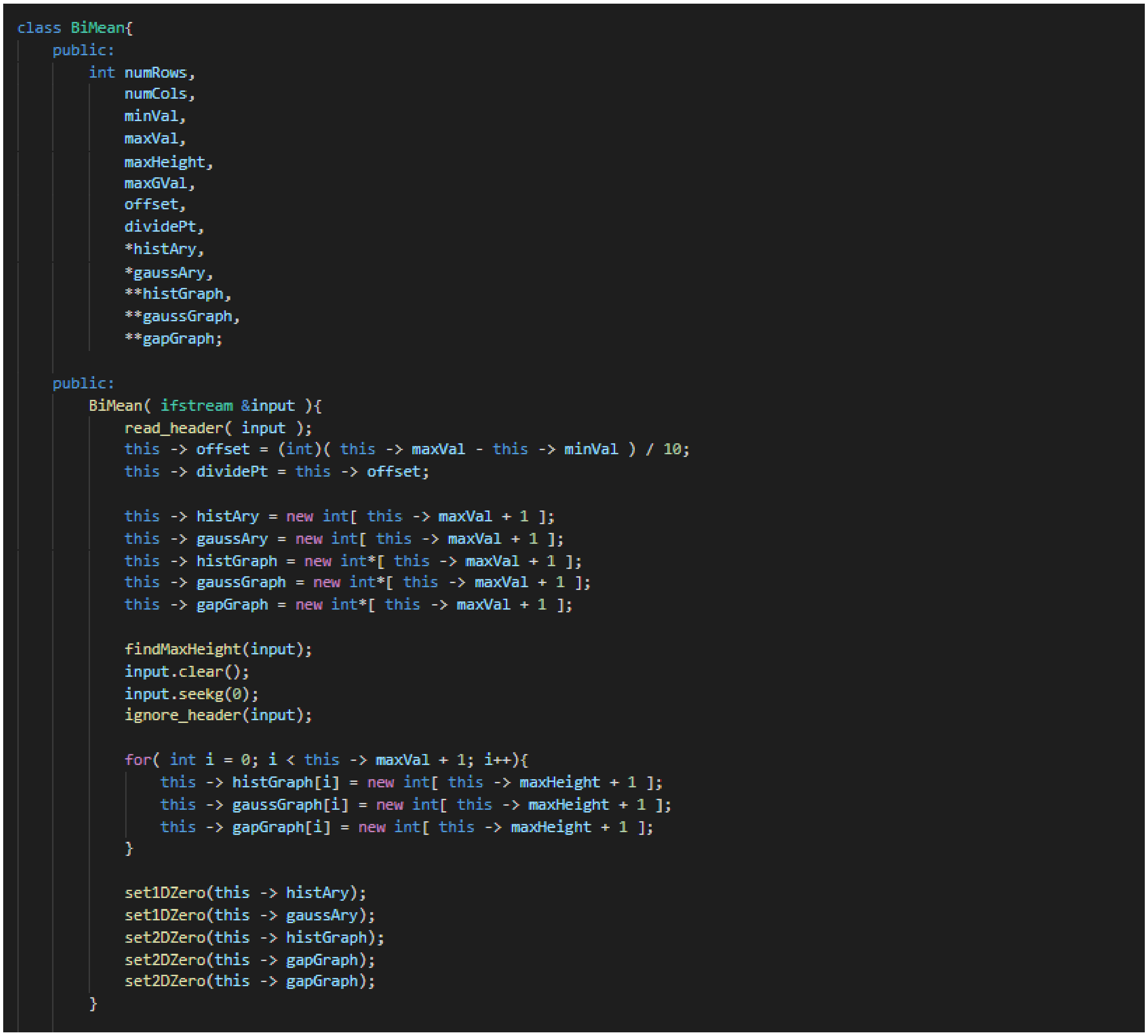
Source Code

Main Class



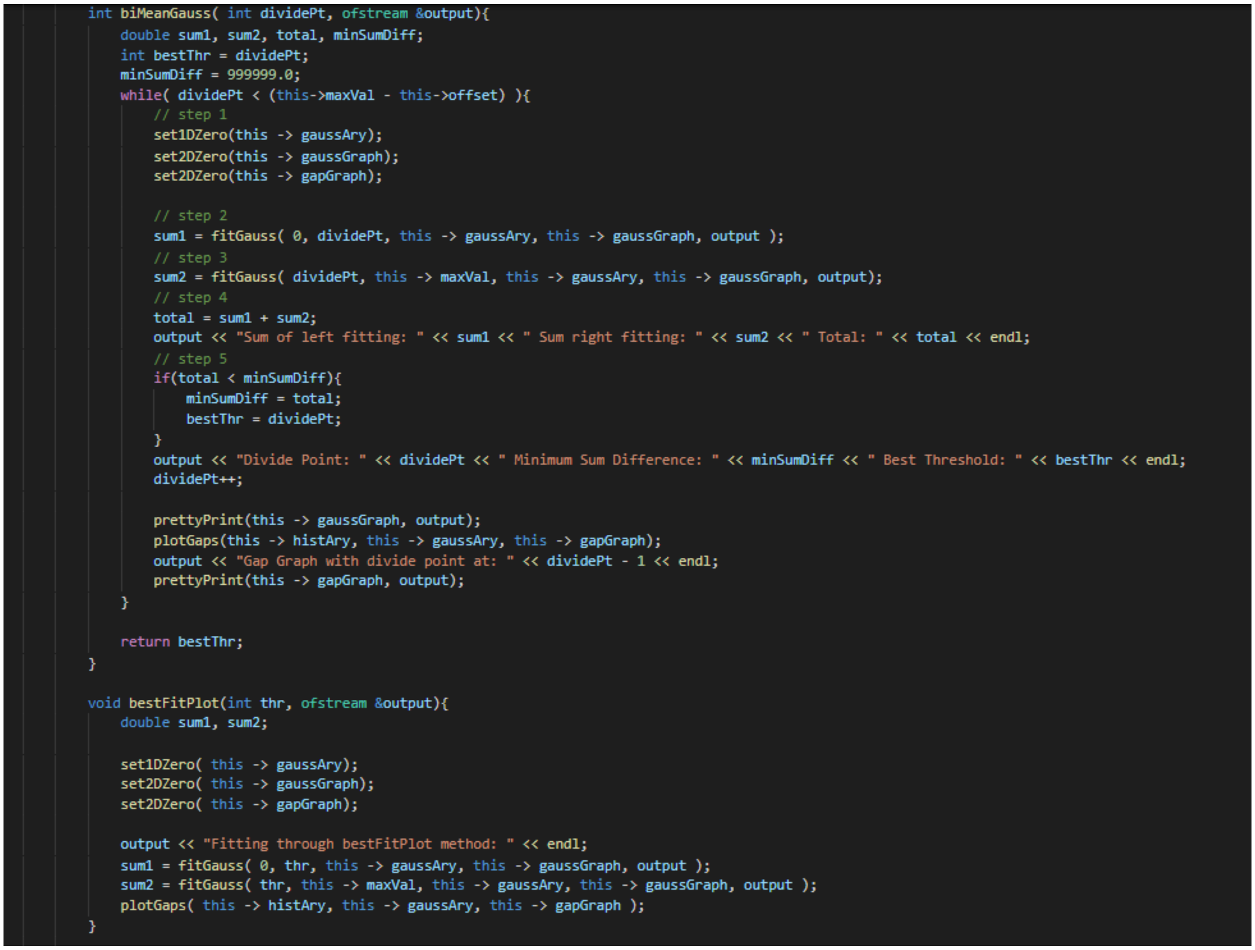
```
int main( int argc, const char *argv[] ){

    string inFile = argv[1];
    string outFile1 = argv[2];
    string outFile2 = argv[3];

    ifstream input;
    input.open( inFile ) ;

    ofstream output1, output2, output3;
    output1.open(outFile1);
    output2.open(outFile2);

    if(input.is_open()){
        if(output1.is_open() && output2.is_open()){


            BiMean *biMeanObj = new BiMean( input );

            biMeanObj -> loadHist( biMeanObj -> histAry, input );
            biMeanObj -> plotHistGraph( biMeanObj -> histGraph );
            output1 << "2D Display of Histogram from given input: " << endl;
            biMeanObj -> prettyPrint( biMeanObj -> histGraph, output1 );

            int bestThrVal = biMeanObj -> biMeanGauss( biMeanObj -> dividePt, output2 );
            output1 << "Best Threshold Value: " << bestThrVal << endl;

            biMeanObj -> bestFitPlot(bestThrVal, output2);

            output1 << "Best fitted plotting: " << endl;
            biMeanObj -> prettyPrint(biMeanObj -> gaussGraph, output1);

            output1 << "Gap graph with best fit plotted: " << endl;
            biMeanObj -> prettyPrint(biMeanObj -> gapGraph, output1);

            input.close();
            output1.close();
            output2.close();
        }
    }


    exit(1);
}
```

BiMean Class



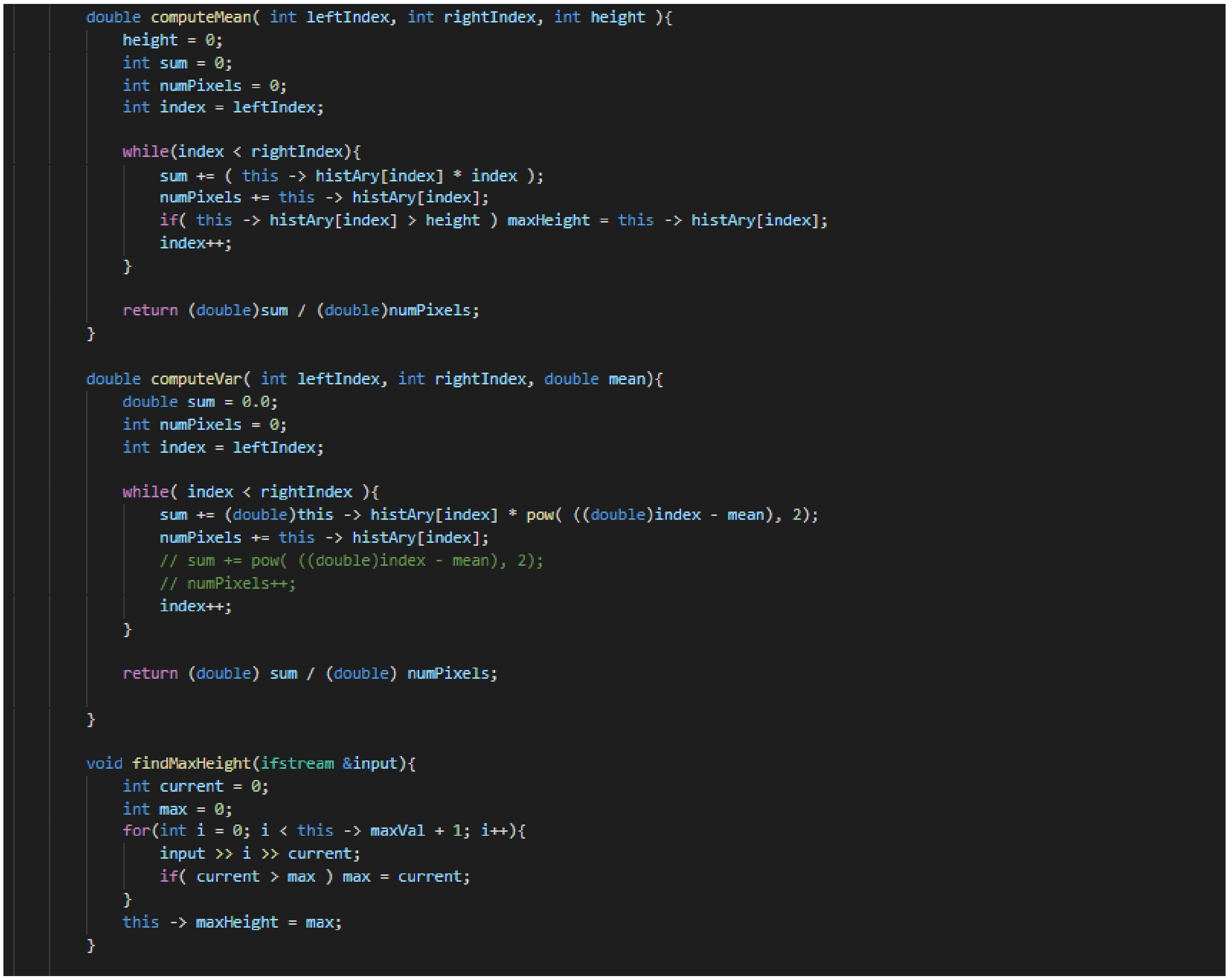
```
class BiMean{
    public:
        int numRows,
            numCols,
            minVal,
            maxVal,
            maxHeight,
            maxGVal,
            offset,
            dividePt,
            *histAry,
            *gaussAry,
            **histGraph,
            **gaussGraph,
            **gapGraph;

    public:
        BiMean( ifstream &input ){
            read_header( input );
            this -> offset = (int)( this -> maxVal - this -> minVal ) / 10;
            this -> dividePt = this -> offset;

            this -> histAry = new int[ this -> maxVal + 1 ];
            this -> gaussAry = new int[ this -> maxVal + 1 ];
            this -> histGraph = new int*[ this -> maxVal + 1 ];
            this -> gaussGraph = new int*[ this -> maxVal + 1 ];
            this -> gapGraph = new int*[ this -> maxVal + 1 ];

            findMaxHeight(input);
            input.clear();
            input.seekg(0);
            ignore_header(input);

            for( int i = 0; i < this -> maxVal + 1; i++){
                this -> histGraph[i] = new int[ this -> maxHeight + 1 ];
                this -> gaussGraph[i] = new int[ this -> maxHeight + 1 ];
                this -> gapGraph[i] = new int[ this -> maxHeight + 1 ];
            }

            set1DZero(this -> histAry);
            set1DZero(this -> gaussAry);
            set2DZero(this -> histGraph);
            set2DZero(this -> gapGraph);
            set2DZero(this -> gapGraph);
        }
```

BiMeanGauss & bestFitPlot



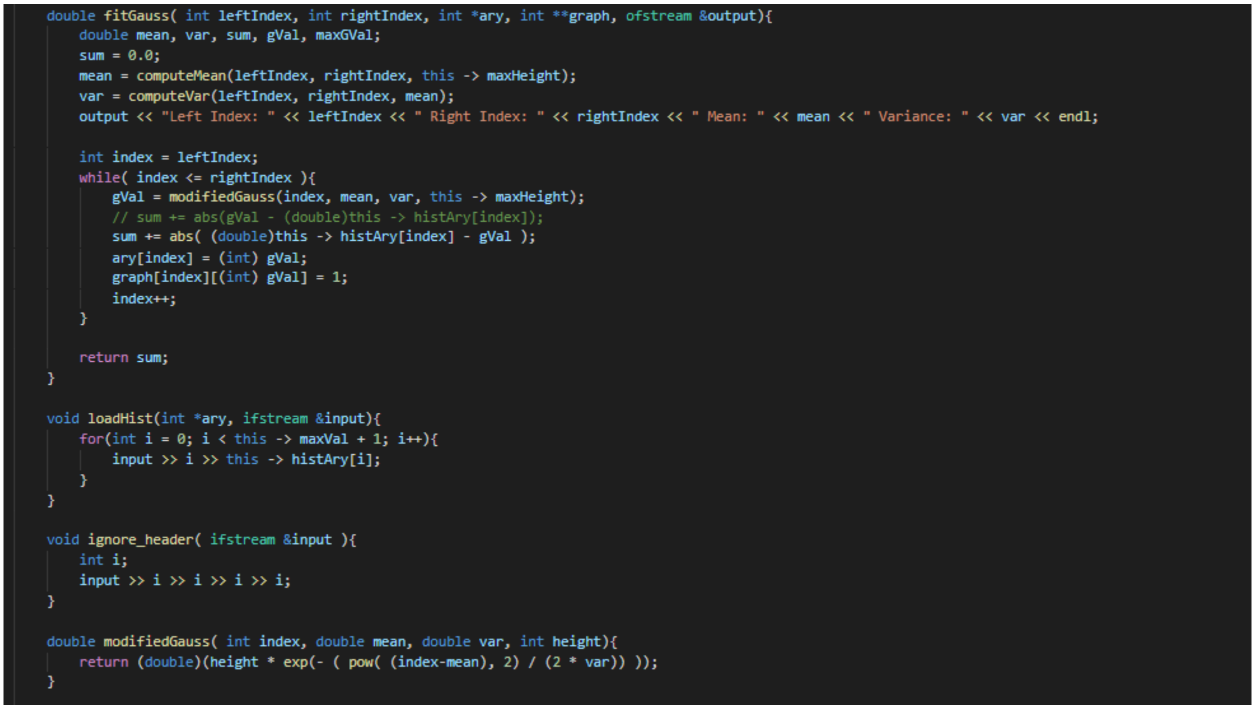
```cpp
int biMeanGauss( int dividePt, ofstream &output){
    double sum1, sum2, total, minSumDiff;
    int bestThr = dividePt;
    minSumDiff = 999999.0;
    while( dividePt < (this->maxVal - this->offset) ){
        // step 1
        set1DZero(this -> gaussAry);
        set2DZero(this -> gaussGraph);
        set2DZero(this -> gapGraph);

        // step 2
        sum1 = fitGauss( 0, dividePt, this -> gaussAry, this -> gaussGraph, output );
        // step 3
        sum2 = fitGauss( dividePt, this -> maxVal, this -> gaussAry, this -> gaussGraph, output);
        // step 4
        total = sum1 + sum2;
        output << "Sum of left fitting: " << sum1 << " Sum right fitting: " << sum2 << " Total: " << total << endl;
        // step 5
        if(total < minSumDiff){
            minSumDiff = total;
            bestThr = dividePt;
        }
        output << "Divide Point: " << dividePt << " Minimum Sum Difference: " << minSumDiff << " Best Threshold: " << bestThr << endl;
        dividePt++;

        prettyPrint(this -> gaussGraph, output);
        plotGaps(this -> histAry, this -> gaussAry, this -> gapGraph);
        output << "Gap Graph with divide point at: " << dividePt - 1 << endl;
        prettyPrint(this -> gapGraph, output);
    }

    return bestThr;
}

void bestFitPlot(int thr, ofstream &output){
    double sum1, sum2;

    set1DZero( this -> gaussAry);
    set2DZero( this -> gaussGraph);
    set2DZero( this -> gapGraph);

    output << "Fitting through bestFitPlot method: " << endl;
    sum1 = fitGauss( 0, thr, this -> gaussAry, this -> gaussGraph, output );
    sum2 = fitGauss( thr, this -> maxVal, this -> gaussAry, this -> gaussGraph, output );
    plotGaps( this -> histAry, this -> gaussAry, this -> gapGraph );
}
```

computeMean & computeVar



```
        double computeMean( int leftIndex, int rightIndex, int height ){
            height = 0;
            int sum = 0;
            int numPixels = 0;
            int index = leftIndex;

            while(index < rightIndex){
                sum += ( this -> histAry[index] * index );
                numPixels += this -> histAry[index];
                if( this -> histAry[index] > height ) maxHeight = this -> histAry[index];
                index++;
            }

            return (double)sum / (double)numPixels;
        }

        double computeVar( int leftIndex, int rightIndex, double mean){
            double sum = 0.0;
            int numPixels = 0;
            int index = leftIndex;

            while( index < rightIndex ){
                sum += (double)this -> histAry[index] * pow( ((double)index - mean), 2);
                numPixels += this -> histAry[index];
                // sum += pow( ((double)index - mean), 2);
                // numPixels++;
                index++;
            }

            return (double) sum / (double) numPixels;

        }

        void findMaxHeight(ifstream &input){
            int current = 0;
            int max = 0;
            for(int i = 0; i < this -> maxVal + 1; i++){
                input >> i >> current;
                if( current > max ) max = current;
            }
            this -> maxHeight = max;
        }
```

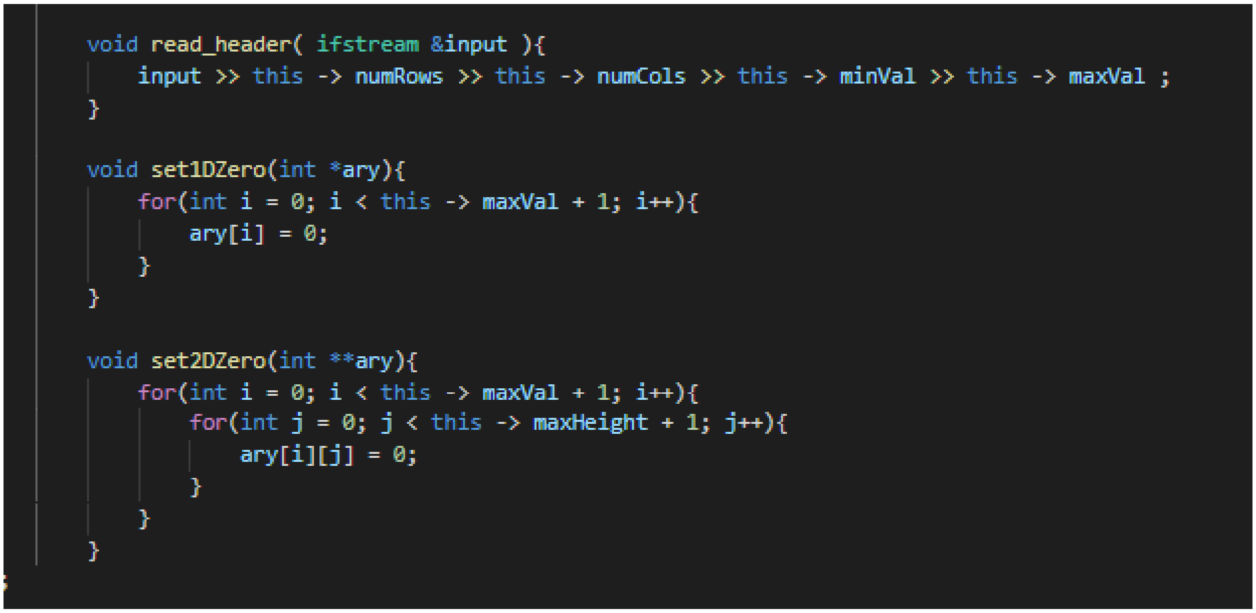
fitGauss & modifiedGauss



```cpp
double fitGauss( int leftIndex, int rightIndex, int *ary, int **graph, ofstream &output){
    double mean, var, sum, gVal, maxGVal;
    sum = 0.0;
    mean = computeMean(leftIndex, rightIndex, this -> maxHeight);
    var = computeVar(leftIndex, rightIndex, mean);
    output << "Left Index: " << leftIndex << " Right Index: " << rightIndex << " Mean: " << mean << " Variance: " << var << endl;

    int index = leftIndex;
    while( index <= rightIndex ){
        gVal = modifiedGauss(index, mean, var, this -> maxHeight);
        // sum += abs(gVal - (double)this -> histAry[index]);
        sum += abs( (double)this -> histAry[index] - gVal );
        ary[index] = (int) gVal;
        graph[index][(int) gVal] = 1;
        index++;
    }

    return sum;
}

void loadHist(int *ary, ifstream &input){
    for(int i = 0; i < this -> maxVal + 1; i++){
        input >> i >> this -> histAry[i];
    }
}

void ignore_header( ifstream &input ){
    int i;
    input >> i >> i >> i >> i;
}

double modifiedGauss( int index, double mean, double var, int height){
    return (double)(height * exp(- ( pow( (index-mean), 2) / (2 * var)) ));
}
```

plotGaps & plotHistGraph & prettyPrint

set1DZero & set2DZero

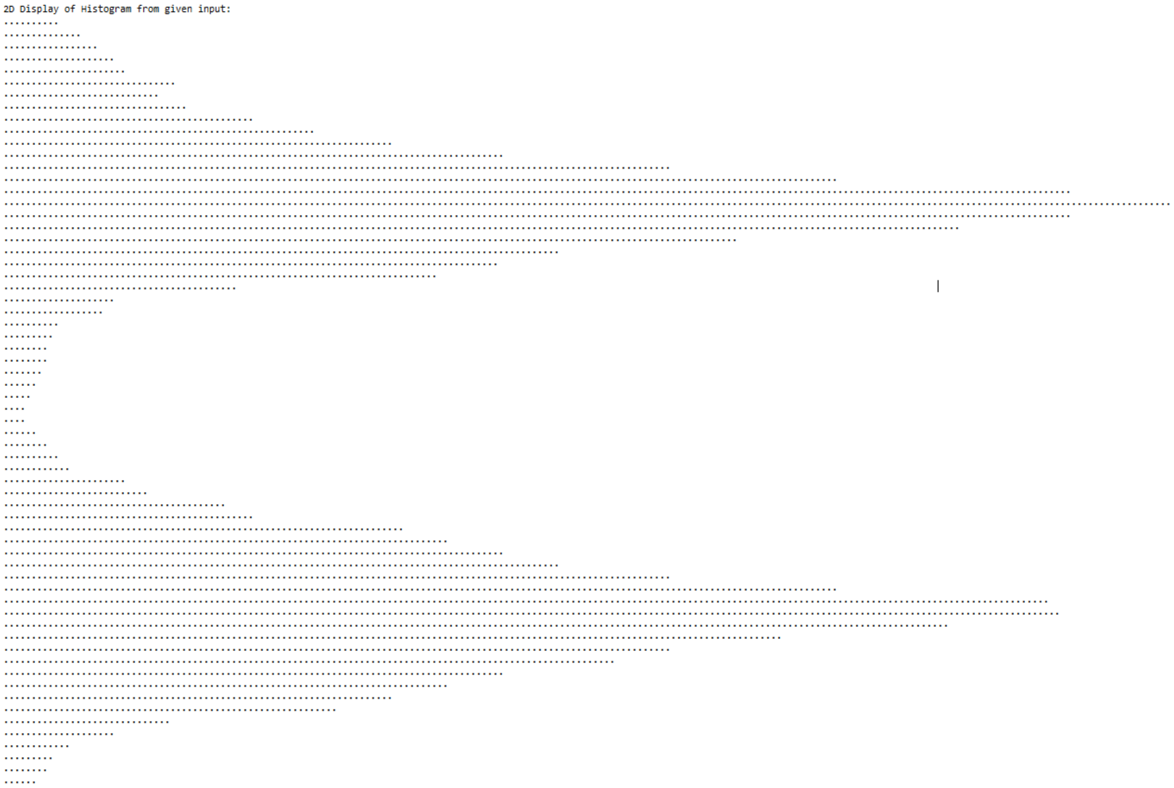


```
    void read_header( ifstream &input ){
        input >> this -> numRows >> this -> numCols >> this -> minVal >> this -> maxVal ;
    }

    void set1DZero(int *ary){
        for(int i = 0; i < this -> maxVal + 1; i++){
            ary[i] = 0;
        }
    }

    void set2DZero(int **ary){
        for(int i = 0; i < this -> maxVal + 1; i++){
            for(int j = 0; j < this -> maxHeight + 1; j++){
                ary[i][j] = 0;
            }
        }
    }
```

Outputs

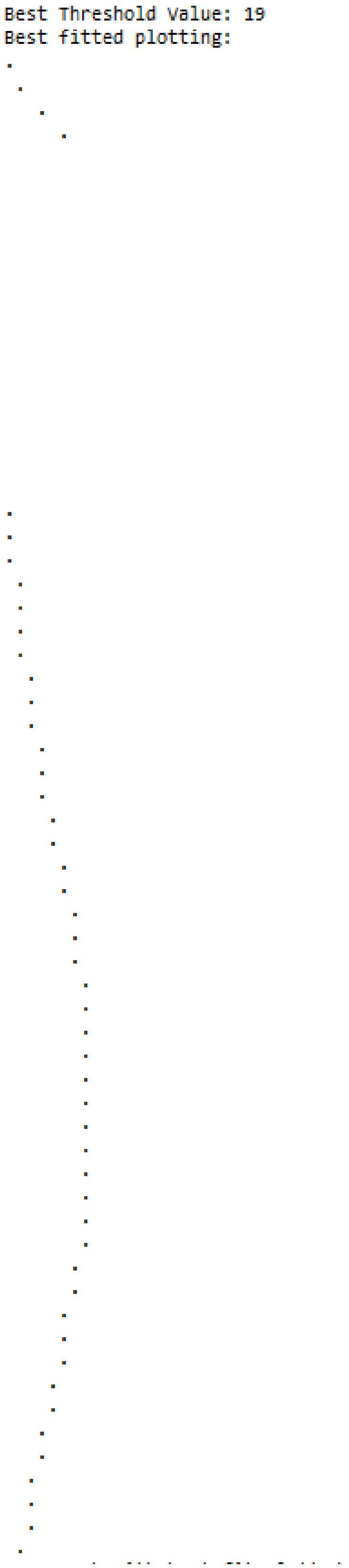
Data 1 Output 1:

Histogram



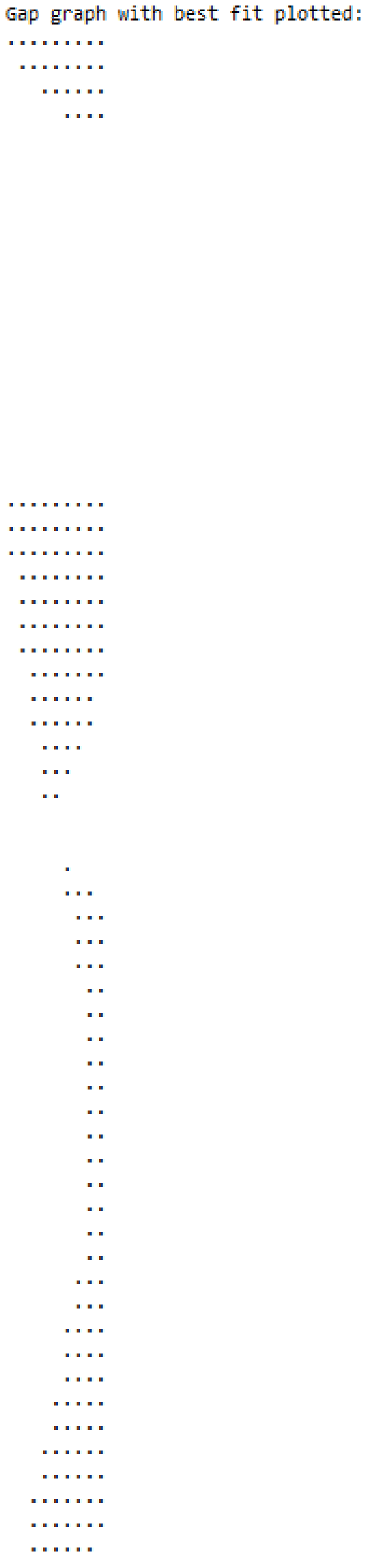
```
2D Display of Histogram from given input:
..........
..............
.................
....................
......................
...............................
............................
.................................
.............................................
........................................................
......................................................................
..........................................................................................
.......................................................................................................................
.....................................................................................................................................................
...............................................................................................................................................................................................
.................................................................................................................................................................................................................
...............................................................................................................................................................................................
...........................................................................................................................................................................
...................................................................................................................................
....................................................................................................
.........................................................................................
..............................................................................
..........................................
....................
..................
..........
.........
........
........
.......
......
.....
....
....
......
........
..........
............
......................
..........................
........................................
.............................................
........................................................................
................................................................................
..........................................................................................
....................................................................................................
.......................................................................................................................
.....................................................................................................................................................
...........................................................................................................................................................................................
.............................................................................................................................................................................................
.........................................................................................................................................................................
...........................................................................................................................................
.......................................................................................................................
.............................................................................................................
..........................................................................................
................................................................................
......................................................................
............................................................
..............................
....................
............
.........
........
......
```

Best Fitted Plots



```
Best Threshold Value: 19
Best fitted plotting:
.
 .
   .
     .















.
.
.
 .
 .
 .
 .
  .
  .
  .
   .
   .
   .
    .
    .
     .
     .
      .
      .
      .
       .
       .
       .
       .
       .
       .
       .
       .
       .
       .
       .
       .
      .
      .
     .
     .
     .
    .
    .
   .
   .
  .
  .
  .
 .
```

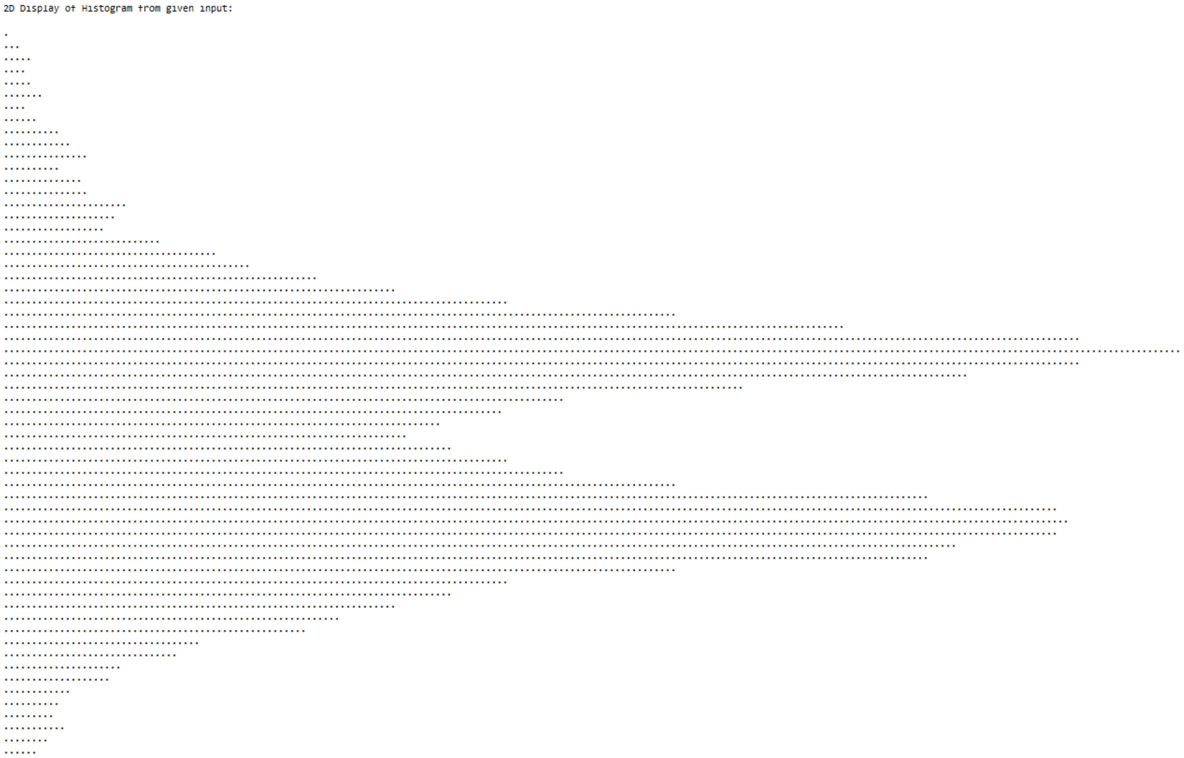
Gap Graph



```
Gap graph with best fit plotted:
.........
 ........
   ......
     ....
















.........
.........
.........
 ........
 ........
 ........
 ........
  .......
  ......
  ......
   ....
   ...
   ..


     .
     ...
      ...
      ...
      ...
       ..
       ..
       ..
       ..
       ..
       ..
       ..
       ..
       ..
       ..
       ..
       ..
      ...
      ...
     ....
     ....
     ....
    .....
    .....
   ......
   ......
  .......
  .......
  ......
```

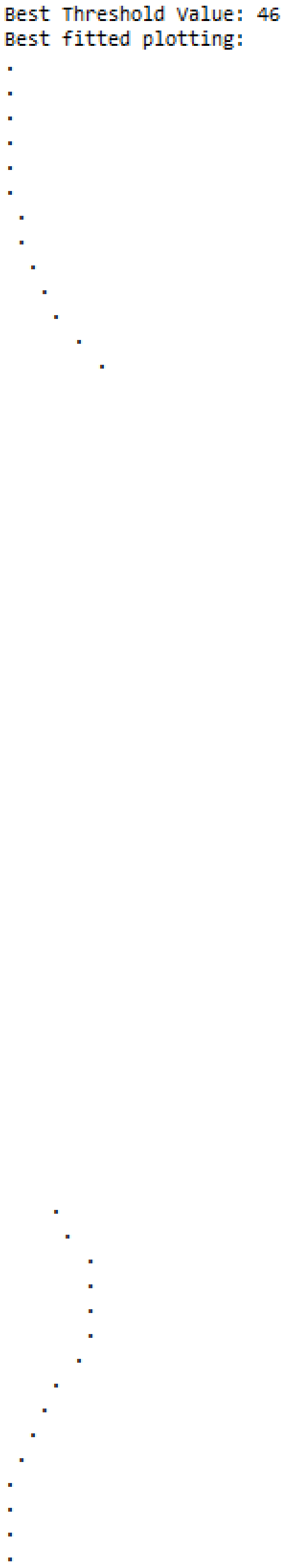
Data 2 Output 1

Histogram



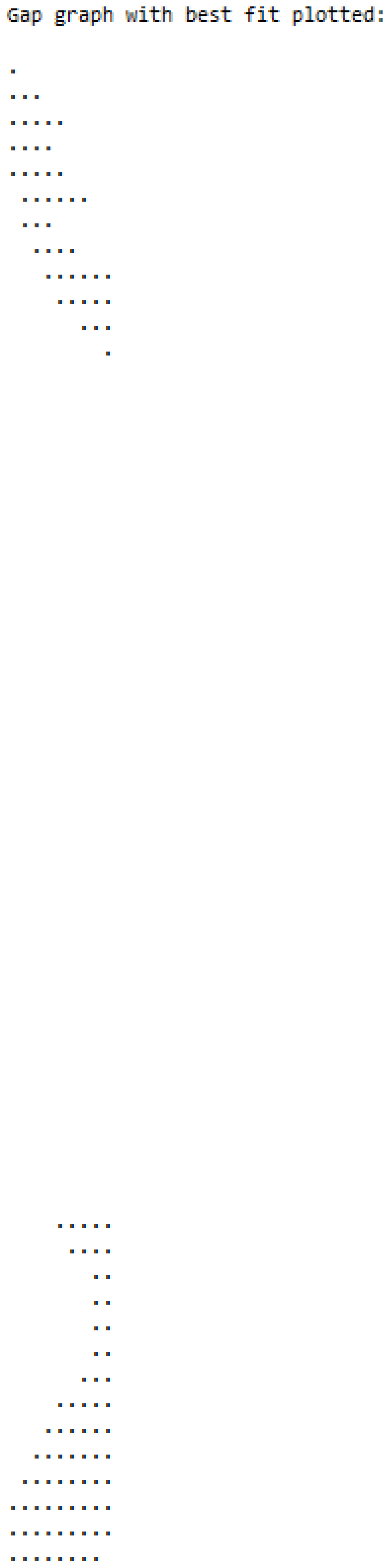
```
2D Display of Histogram from given input:

.
...
.....
....
.....
.......
....
......
..........
............
...............
..........
..............
...............
......................
....................
..................
............................
......................................
............................................
........................................................
......................................................................
..........................................................................................
........................................................................................................................
.....................................................................................................................................................
................................................................................................................................................................................................
..................................................................................................................................................................................................................
................................................................................................................................................................................................
............................................................................................................................................................................
...................................................................................................................................
....................................................................................................
.........................................................................................
..............................................................................
........................................................................
................................................................................
..........................................................................................
....................................................................................................
........................................................................................................................
.....................................................................................................................................................................
.............................................................................................................................................................................................
...............................................................................................................................................................................................
.............................................................................................................................................................................................
..........................................................................................................................................................................
.....................................................................................................................................................................
........................................................................................................................
..........................................................................................
................................................................................
......................................................................
............................................................
......................................................
...................................
...............................
.....................
...................
............
..........
.........
...........
........
......
```

Best Fitted Plot



```
Best Threshold Value: 46
Best fitted plotting:
.
.
.
.
.
.
 .
 .
  .
   .
    .
      .
        .

































    .
     .
       .
       .
       .
       .
      .
    .
   .
  .
 .
.
.
.
.
```

Gap Graph



```
Gap graph with best fit plotted:

.
...
.....
....
.....
 ......
 ...
  ....
   ......
    .....
      ...
        .



































    .....
     ....
       ..
       ..
       ..
       ..
      ...
    .....
   ......
  .......
 ........
.........
.........
........
```

Output 2 is hard to read on document, here is a link to my output files:

Google Drive of Data 1 Output 2

Google Drive of Data 2 Output 2