CV Exam 2: Decoding Java

**Student:** Pawan Bhatta

**Source Code for Run Length Encoding:**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdarg>

using namespace std;

class Encoding

{

public:

int numRows;

int numCols;

int minVal;

int maxVal;

int startRow;

int startCol;

int greyScale;

int length;

int \*\*img;

Encoding(ifstream &inputFile)

{

loadHeader(inputFile);

img = new int \*[numRows];

for (int i = 0; i < numRows; i++)

{

img[i] = new int[numCols];

}

loadImage(inputFile);

}

void loadHeader(ifstream &inputImg)

{

inputImg >> numRows >> numCols >> minVal >> maxVal;

}

void loadImage(ifstream &inputImg)

{

for (int i = 0; i < numRows; i++)

{

for (int j = 0; j < numCols; j++)

{

inputImg >> img[i][j];

}

}

}

void print2DArray()

{

for (int i = 0; i < numRows; i++)

{

for (int j = 0; j < numCols; j++)

{

cout << img[i][j];

}

cout << "\n";

}

}

void encodeOneRow(int row, ofstream &encodeFile, ofstream &debugFile)

{

startRow = row;

startCol = 0;

greyScale = img[row][0];

length = 1;

for (int i = 1; i < numCols; i++)

{

if (img[row][i] == greyScale)

{

length++;

}

else

{

encodeFile << row << " " << i << " " << greyScale << " " << length << " ";

encodeFile << "\n";

length = 1;

greyScale = img[row][i];

}

if (i == numCols - 1)

{

encodeFile << row << " " << i << " " << greyScale << " " << length << " ";

encodeFile << "\n";

length = 1;

greyScale = img[row][i];

}

}

}

};

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

{

string inputFileName = argv[1];

ifstream inputFile;

inputFile.open(inputFileName);

string encodeFileName = argv[2];

ofstream encodeFile;

encodeFile.open(encodeFileName);

string debugFileName = argv[3];

ofstream debugFile;

debugFile.open(debugFileName);

if (inputFile.is\_open() && debugFile.is\_open() && encodeFile.is\_open())

{

Encoding t(inputFile);

encodeFile << t.numRows << " " << t.numCols << " " << t.minVal << " " << t.maxVal << endl;

for (int r = 0; r < t.numRows; r++)

{

t.encodeOneRow(r, encodeFile, debugFile);

}

}

else

{

cout << "Error reading file." << endl;

}

inputFile.close();

debugFile.close();

encodeFile.close();

return 0;

}

**Output for Run Length Encoding:**

**Data\_1:**

Original Image

10 22 0 9

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4

4 0 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3 3 3 0 0 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 7 7

7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7

7 7 0 0 0 0 0 2 3 4 2 2 3 3 4 4 4 4 4 4 0 0

0 0 0 0 0 0 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Encoded/Compressed Image:

10 22 0 9

0 15 0 15

0 21 4 7

1 1 4 1

1 2 0 1

1 11 4 9

1 21 0 11

2 5 0 5

2 21 3 17

3 3 3 3

3 5 0 2

3 11 3 6

3 21 7 11

4 21 7 22

5 2 7 2

5 7 0 5

5 8 2 1

5 9 3 1

5 10 4 1

5 12 2 2

5 14 3 2

5 20 4 6

5 21 0 2

6 6 0 6

6 11 1 5

6 16 9 5

6 21 1 6

7 10 1 10

7 21 6 12

8 21 0 22

9 21 0 22

**Data\_2:**

Original Image

10 22 0 9

1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4

4 4 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 9 9 9 9 9 9 9 9 9 9 3 9 9 9 9 9 9

3 3 3 0 0 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 7 7

7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8

8 8 8 0 0 5 5 2 3 4 2 2 2 2 4 4 4 4 4 4 0 0

0 0 0 0 0 0 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Encoded Image:

10 22 0 9

0 5 1 5

0 15 0 10

0 21 4 7

1 11 4 11

1 21 0 11

2 5 0 5

2 15 9 10

2 16 3 1

2 21 9 6

3 3 3 3

3 5 0 2

3 11 3 6

3 21 7 11

4 13 7 13

4 21 8 9

5 3 8 3

5 5 0 2

5 7 5 2

5 8 2 1

5 9 3 1

5 10 4 1

5 14 2 4

5 20 4 6

5 21 0 2

6 6 0 6

6 11 1 5

6 16 9 5

6 21 1 6

7 10 1 10

7 21 6 12

8 21 0 22

9 21 0 22