CS 323\_33 Programming Language: C++

Project #6.2 Running Length Decoding

Andres Quintero

Due Date:

Soft copy: 2/29/2020

Hard copy: 2/29/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: nameEncodeFile 🡨 argv[1]

encodeFile 🡨 open (nameEncodeFile)

step 1: numRows, numCols, minVal, maxVal 🡨 Read from encodeFile

step 2: whichMethod 🡨 read from encodeFile // on the 2nd text line

step 3: nameEncodeFile 🡨 nameEncodeFile + “\_Decoded”

decodeFile 🡨open (nameDecodeFile)

step 4: decodeFile 🡨 output numRows, numCols, minVal, maxVal to decodeFile

step 5: case of whichMethod

case 1: decodeMethod1 (encodeFile, decodeFile)

case 4: decodeMethod4 (encodeFile, decodeFile)

default: error message

exit program

step 6: close all files

**decodeMethod4():**

Step 0: rowCount <- 0

colCount <- 0

Step 1: startRow, startCol, color, length <- read from encodeFile

Step 2:

If rowCount == numRows && colCount == numCols

for [0,length)

decodeFile <- outputColor

colCount++

if (colCount == numCols) // checking for end of column

decodeFile <- endl

colCount <- 0

rowCount++

Else //Print zeros instead

decodeFile <- 0

colCount++

if (colCount == numCols) // checking for end of column

decodeFile <- endl

colCount <- 0

rowCount++

Step 3: Repeat step 2 until eof (encodeFile)

Step 4: If still rows left to output

decodeFile <- 0

colCount++

if (colCount == numCols) // checking for end of column

decodeFile <- endl

colCount <- 0

rowCount++

until rowCount < numRows

**Source code:**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

// Prototypes

void decodeMethod1(fstream& encodeFile, fstream& decodeFile, int numRows, int numCols);

void decodeMethod4(fstream& encodeFile, fstream& decodeFile, int numRows, int numCols);

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

// 0

string nameEncodeFile = argv[1];

fstream encodeFile(argv[1]);

// cout << nameEncodeFile << endl; // DEBUGSTUFF

// 1

int numRows, numCols, minVal, maxVal, whichMethod;

encodeFile >> numRows;

encodeFile >> numCols;

encodeFile >> minVal;

encodeFile >> maxVal;

// 2

encodeFile >> whichMethod;

if(whichMethod != 1 && whichMethod !=4){

cout << "ERROR: data is not encoded properly" << endl;

exit(1);

}

// 3

int pos = nameEncodeFile.find(".txt");

nameEncodeFile = nameEncodeFile.substr(0,pos);

string nameDecodeFile = nameEncodeFile + "\_Decoded.txt";

fstream decodeFile(nameDecodeFile, fstream::out);

// 4

decodeFile << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;

// 5

if(whichMethod == 1){

decodeMethod1(encodeFile, decodeFile, numRows, numCols);

} else if (whichMethod == 4) {

decodeMethod4(encodeFile, decodeFile, numRows, numCols);

} else {

cout << "ERROR: data is not encoded properly" << endl;

exit(1);

}

// 6

encodeFile.close();

decodeFile.close();

return 0;

}

// My Functions

void decodeMethod4(fstream& encodeFile, fstream& decodeFile, int numRows, int numCols){

int rowCount = 0, colCount = 0;// these are the position of the 'read head'

int startRow, startCol, color, length;

encodeFile >> startRow;

encodeFile >> startCol;

encodeFile >> color;

encodeFile >> length;

// read from image, if row&col match print color for length run

// else print zero and move colCOunt++

while(!encodeFile.eof() && rowCount < numRows){

if(rowCount == startRow && colCount == startCol){

for(int i = 0; i < length; i++){

decodeFile << color << " ";

colCount++;

if(colCount == numCols){ // because we keep wrap around

decodeFile << endl;

colCount = 0;

rowCount++;

}

}

encodeFile >> startRow;

encodeFile >> startCol;

encodeFile >> color;

encodeFile >> length;

} else {

decodeFile << 0 << " ";

colCount++;

if(colCount == numCols){ // wrap around AKA newline

decodeFile << endl;

colCount = 0;

rowCount++;

}

}

}

//if eof but still some Rows left, fill with zero

while(rowCount < numRows){

decodeFile << 0 << " ";

colCount++;

if(colCount == numCols){ // because we keep wrap around

decodeFile << endl;

colCount = 0;

rowCount++;

}

}

}

void decodeMethod1(fstream& encodeFile, fstream& decodeFile, int numRows, int numCols){

// cout << "Calling decodeMethod1" << endl; // DEBUGSTUFF

int rowCount = 0, colCount = 0;

// 0

rowCount = 0;

colCount = 0; // here because we reset to zero when we are at the last column only;

int startRow, startCol, color, length;

while(!encodeFile.eof() && rowCount < numRows){

encodeFile >> startRow;

encodeFile >> startCol;

encodeFile >> color;

encodeFile >> length;

// cout << startRow << " " << startCol << " " << color << " " << length << endl; // DEBUGSTUFF

// 3

for(int i = 0; i < length; i++){

decodeFile << color << " ";

colCount++;

if(colCount == numCols){ // because we keep wrap around

decodeFile << endl;

colCount = 0;

rowCount++;

}

}

}

}

**Image1 files:**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**END OF IMAGE 1 FILES**

**Image 2 files:**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**END OF IMAGE 2 FILES**