CSCI 381 (Java)

Section 31

Project 7.2 Run Length Decode

Essam Yousry

Due Date of soft copy: 04/11/2018

Due Date of hard copy: 04/12/2018

Algorithm Steps for Run Length Decode (Main)

step 0: inFile  open input file

outFile  open a text file to write

initializing those needed in the data structures

step 1: Read the image header from inFile

Write the header to outFile

step 2: methodUsed <-- read from inFile

if methodUsed is not within 1 – 4

exit with error message

step 3: case of methodUsed

1: call deCodeMethod1 (inFile) // on your own

2: call deCodeMethod2 (inFile) // on your own

3: call deCodeMethod3 (inFile) // on your own

4: call deCodeMethod4 (inFile)) // on your own

step 4: closed all files

Source Code

import java.util.Scanner;

import java.io.FileReader;

import java.io.IOException;

import java.io.FileWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.PrintWriter;

import java.util.\*;

public class RunLengthDecode{

    public static void main(String [] args){

        File inFile = new File (args[0]);

        File outFile = new File (args[1]);

        int row, col, min, max, method;

        int [] vars = new int [4];

        try {

            PrintWriter pw = new PrintWriter(new FileWriter(outFile));

            Scanner sc = new Scanner(inFile);

            int c = 0;

            while (sc.hasNextInt() && c != 4){

                vars[c++] = sc.nextInt();

            }

            row = vars[0];

            col = vars[1];

            min = vars[2];

            max = vars[3];

            method = sc.nextInt();

            if ((method < 1) || (method > 4)){

                System.out.println("The number should be between 1 and 4");

                System.exit(0);

            }

            int [][] data = new int [row][col];

            switch (method){

                case 1: DecodeMethod1(data, row, col, min, max, outFile, sc);

                    break;

                case 2: DecodeMethod2(data, row, col, min, max, outFile, sc);

                    break;

                case 3: DecodeMethod3(data, row, col, min, max, outFile, sc);

                    break;

                case 4: DecodeMethod4(data, row, col, min, max, outFile, sc);

                    break;

            }

            pw.print(row);

            pw.print(' ');

            pw.print(col);

            pw.print(' ');

            pw.print(min);

            pw.print(' ');

            pw.print(max);

            pw.print(' ');

            pw.println();

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

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

                    pw.print(data[i][j]);

                    pw.print(' ');

                }

                pw.println();

            }

            pw.close();

        }

        catch (Exception e) {

            System.out.println("Error " + e);

        }

    }

    //decode non-zeros without wrap around

    public static void DecodeMethod1(int [][] array, int row, int col, int min, int max, File outFile, Scanner in){

        int i = 0;

        int j = 0;

        int count = 0;

        int val = 0;

        try{

            while(in.hasNextInt()){

                i = in.nextInt() - 1;

                j = in.nextInt() - 1;

                val = in.nextInt();

                count = in.nextInt();

                while (count != 0){

                    array[i][j] = val;

                    j++;

                    if (j == col){

                        i++;

                        j = 0;

                    }

                    count--;

                }

            }

        }

        catch (Exception e) {

            System.out.println("Error " + e);

        }

    }

    //decode non-zeros with wrap around

    public static void DecodeMethod2(int [][] array, int row, int col, int min, int max, File outFile, Scanner in){

        int i = 0;

        int j = 0;

        int count = 0;

        int val = 0;

        try{

            while(in.hasNextInt()){

                i = in.nextInt() - 1;

                j = in.nextInt() - 1;

                val = in.nextInt();

                count = in.nextInt();

                while (count != 0){

                    array[i][j] = val;

                    j++;

                    if (j == col){

                        i++;

                        j = 0;

                    }

                    count--;

                }

            }

        }

        catch (Exception e) {

            System.out.println("Error " + e);

        }

    }

    //decode zeros and non-zeros without wrap around

    public static void DecodeMethod3(int [][] array, int row, int col, int min, int max, File outFile, Scanner in){

        int i = 0;

        int j = 0;

        int count = 0;

        int val = 0;

        try{

            while(in.hasNextInt()){

                i = in.nextInt() - 1;

                j = in.nextInt() - 1;

                val = in.nextInt();

                count = in.nextInt();

                while (count != 0){

                    array[i][j] = val;

                    j++;

                    if (j == col){

                        i++;

                        j = 0;

                    }

                    count--;

                }

            }

        }

        catch (Exception e) {

            System.out.println("Error " + e);

        }

    }

    //decode zeros and non-zeros with wrap around

    public static void DecodeMethod4(int [][] array, int row, int col, int min, int max, File outFile, Scanner in){

        int i = 0;

        int j = 0;

        int count = 0;

        int val = 0;

        try{

            while(in.hasNextInt()){

                i = in.nextInt() - 1;

                j = in.nextInt() - 1;

                val = in.nextInt();

                count = in.nextInt();

                while (count != 0){

                    array[i][j] = val;

                    j++;

                    if (j == col){

                        i++;

                        j = 0;

                    }

                    count--;

                }

            }

        }

        catch (Exception e) {

            System.out.println("Error " + e);

        }

    }

}

Output







