CSCI 381 (Java)

Section 31

Project 3 Eight Connected Component

Essam Yousry

Due Date of soft copy: 02/20/2018

Due Date of hard copy: 02/22/2018

Algorithm Steps for EightCC:

Step 0: read the image header

dynamically allocate zeroFramedAry and all other arrays

Step 1: load the input image onto zeroFramedAry

Step 2: - 8ConnectCC\_Pass1

- prettyprint the result of pass1

- print EQAry // with index up to newLabel with proper caption

Step 3: - 8ConnectCC\_Pass2

- prettyprint the result of pass2

- print EQAry // with index up to newLabel with caption

Step 4: - manageEQAry

- print EQAry

Step 5: - 8ConnectCC\_Pass3

- prettyprint the result of pass3 of the connected c.c.

- Output the result of pass3 to outFile2 with updated image header

- print the properties of the connected c.c.

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;

public class EightCC{

    public static void main(String [] args){

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

        String filename = args[0];

        int dot = filename.indexOf(".");

        String fileNameWithoutExtension = filename.substring(0, dot);

        String out1 = fileNameWithoutExtension + "\_ThreePasses.txt";

        String out2 = fileNameWithoutExtension + "\_ImageFile.txt";

        String out3 = fileNameWithoutExtension + "\_CCProperty.txt";

        File outFile1 = new File (out1);

        File outFile2 = new File (out2);

        File outFile3 = new File (out3);

        int row, col, min, max;

        int [] neighborAry = new int [5];

        int [] vars = new int [4];

        int newMin, newMax;

        int newLabel = 0;

        int EQsize;

        try {

            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];

            EQsize = (row \* col) / 2;

            int [][] zeroFramedAry = new int [row + 2][col + 2];

            int [] EQAry = new int [EQsize];

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

                EQAry[i] = i;

            }

            loadImage(zeroFramedAry, sc, row, col);

            zeroFrame(zeroFramedAry, row, col);

            int x = ConnectCC\_Pass1(zeroFramedAry, EQAry, EQsize, row, col, newLabel, outFile1);

            ConnectCC\_Pass2(zeroFramedAry, EQAry, EQsize, row, col, x, outFile1);

            int [] y = manageEQAry(EQAry, x, outFile1);

            int numComp = y[0];

            ConnectCC\_Pass3(zeroFramedAry, EQAry, y, EQsize, row, col, numComp, outFile1, outFile2);

            printCCProperty(zeroFramedAry, EQAry, row, col, numComp, outFile3);

        }

        catch (Exception e) {

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

        }

    }

    public static int getMax (int[][] numbers, int row, int col){

        int max = numbers[1][1];

        for (int i = 1; i < row + 1; i++){

            for (int j = 1; j < col + 1; j++){

                if (numbers[i][j] > max) max = numbers[i][j];

            }

        }

        return max;

    }

    public static int getMin (int[][] numbers, int row, int col){

        int min = numbers[1][1];

        for(int i = 1; i < row + 1; i++){

            for (int j = 1; j < col + 1; j++){

                if (numbers[i][j] < min) min = numbers[i][j];

            }

        }

        return min;

    }

    public static int getMinComp(int [] array){

        int min = array[0];

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

            if (array[i] < min) min = array[i];

        }

        return min;

    }

    public static void loadImage(int [][] array, Scanner sc, int row, int col){

        for (int i = 1; i < row + 1; i++){

            for (int j = 1; j < col + 1; j++){

                array[i][j] = sc.nextInt();

            }

        }

    }

    public static void zeroFrame(int [][] array, int row, int col){

        for (int i = 1; i < row + 1; i++){

            for (int j = 1; j < col + 1; j++){

                array[0][j] = 0;

                array[row + 1][j] = 0;

                array[i][0] = 0;

                array[i][col + 1] = 0;

            }

        }

        array[0][0] = 0;

        array[0][col + 1] = 0;

        array[row + 1][0] = 0;

        array[row + 1][col + 1] = 0;

    }

    public static void loadNeighborsPass1(int [] neighborAry, int [][] mirrorFramedAry, int i, int j){

            neighborAry[0] = mirrorFramedAry[i - 1][j - 1];

            neighborAry[1] = mirrorFramedAry[i - 1][j];

            neighborAry[2] = mirrorFramedAry[i - 1][j + 1];

            neighborAry[3] = mirrorFramedAry[i][j - 1];

    }

    public static void loadNeighborsPass2(int [] neighborAry, int [][] mirrorFramedAry, int i, int j){

            neighborAry[0] = mirrorFramedAry[i][j];

            neighborAry[1] = mirrorFramedAry[i][j + 1];

            neighborAry[2] = mirrorFramedAry[i + 1][j - 1];

            neighborAry[3] = mirrorFramedAry[i + 1][j];

            neighborAry[4] = mirrorFramedAry[i + 1][j + 1];

    }

    public static int [] removeZerosPass1(int [] array, int size){

        int length = 0;

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

            if (array[i] != 0) length++;

        }

        int [] array2 = new int [length];

        int t = 0;

        for (int s = 0; s < array.length; s++){

            if(array[s] != 0){

                array2[t] = array[s];

                t++;

            }

        }

        return array2;

    }

    public static int [] removeZerosPass2(int [] array, int size){

        int length = 0;

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

            if (array[i] != 0) length++;

        }

        int [] array2 = new int [length];

        int t = 0;

        for (int s = 0; s < array.length; s++){

            if(array[s] != 0){

                array2[t] = array[s];

                t++;

            }

        }

        return array2;

    }

    public static boolean isUnique(int[] array, int num) {

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

            if (array[i] == num) {

                return false;

            }

        }

        return true;

    }

    public static int[] toUniqueArray(int[] array) {

        int[] temp = new int[array.length];

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

            temp[i] = -1;

        }

        int counter = 0;

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

            if (isUnique(temp, array[i]))

                temp[counter++] = array[i];

        }

        int[] uniqueArray = new int[counter];

        System.arraycopy(temp, 0, uniqueArray, 0, uniqueArray.length);

        return uniqueArray;

    }

public static int ConnectCC\_Pass1(int [][] array, int [] EQAry, int EQsize, int row, int col, int label, File out){

    try {

        int [] neighborAry = new int [4];

        int [] NoZeroAry;

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

        pw.print("PASS 1 Pretty Print");

        pw.println();

        pw.print("-------------------");

        pw.println();

        pw.println();

        for (int i = 1; i < row + 1; i++){

            for (int j = 1; j < col + 1; j++){

                loadNeighborsPass1(neighborAry, array, i, j);

                if (array[i][j] > 0){

                    //Case 1

                    boolean check = true;;

                    for(int d = 0; d < 4; d++){

                        if (neighborAry[d] != 0){

                            check = false;

                            break;

                        }

                    }

                    if (check == true) {

                        label++;

                        array[i][j] = label;

                    }

                    if (check != true){

                    //Case 2

                        NoZeroAry = removeZerosPass1(neighborAry, 4);

                        boolean test = true;

                        int first = NoZeroAry[0];

                        for (int s = 0; s < NoZeroAry.length; s++){

                            if (NoZeroAry[s] != first){

                                test = false;

                                break;

                            }

                        }

                        if (test == true) array[i][j] = first;

                        //Case 3

                        if ((check == false) && (test == false)) {

                            array[i][j] = getMinComp(NoZeroAry);

                            updateEQAry(EQAry, EQsize, array[i][j], getMinComp(NoZeroAry));

                        }

                    }

                }

            }

        }

        prettyPrint(array, row, col, pw);

        pw.println();

        pw.print("EQAry");

        pw.println();

        pw.print("-----");

        pw.println();

        for (int s = 1; s <= label; s++){

            pw.print(EQAry[s]);

            pw.print(' ');

        }

        pw.println();

        pw.println();

        pw.close();

        }

        catch (Exception e) {

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

        }

    return label;

    }

    public static void ConnectCC\_Pass2(int [][] array, int [] EQAry, int EQsize, int row, int col, int label, File out){

    try {

        int [] neighborAry = new int [5];

        int [] NoZeroAry;

        PrintWriter pw = new PrintWriter(new FileWriter(out, true));

        pw.print("--------------------------------------------------");

        pw.println();

        pw.println();

        pw.print("PASS 2 Pretty Print");

        pw.println();

        pw.print("-------------------");

        pw.println();

        pw.println();

        for (int i = row; i >= 1; i--){

            for (int j = col; j >= 1; j--){

                loadNeighborsPass2(neighborAry, array, i, j);

                if (array[i][j] > 0){

                    //Case 1

                    boolean check = true;

                    for(int d = 0; d < 5; d++){

                        if (neighborAry[d] != 0){

                            check = false;

                            break;

                        }

                    }

                    if (check != true){

                        NoZeroAry = removeZerosPass2(neighborAry, 5);

                        boolean test = true;

                        int first = NoZeroAry[0];

                        for (int s = 0; s < NoZeroAry.length; s++){

                            if (NoZeroAry[s] != first){

                                test = false;

                                break;

                            }

                        }

                        //if (test == true) array[i][j] = NoZeroAry[0];

                        //Case 3

                        if (test == false){

                            updateEQAry(EQAry, EQsize, array[i][j], getMinComp(NoZeroAry));

                            array[i][j] = getMinComp(NoZeroAry);

                        }

                    }

                }

            }

        }

        prettyPrint(array, row, col, pw);

        pw.println();

        pw.print("EQAry");

        pw.println();

        pw.print("-----");

        pw.println();

        for (int s = 1; s <= label; s++){

            pw.print(EQAry[s]);

            pw.print(' ');

        }

        pw.println();

        pw.println();

        pw.close();

    }

        catch (Exception e) {

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

        }

    }

    public static void ConnectCC\_Pass3(int [][] array, int [] EQAry, int [] originalArray, int EQsize, int row, int col, int count, File out, File out2){

        try{

            PrintWriter pw = new PrintWriter(new FileWriter(out, true));

            PrintWriter pw2 = new PrintWriter(new FileWriter(out2));

            pw.print("--------------------------------------------------");

            pw.println();

            pw.println();

            pw.print("PASS 3 Pretty Print");

            pw.println();

            pw.print("-------------------");

            pw.println();

            pw.println();

            int [] uniqueOriginalArray = toUniqueArray(originalArray);

            int [] uniqueEQAry = toUniqueArray(EQAry);

            for(int i = 1; i < row + 1; i++){

                for (int j = 1; j < col + 1; j++){

                    for(int d = 1; d < count; d++){

                        if (array[i][j] == uniqueOriginalArray[d]){

                            array[i][j] = uniqueEQAry[d];

                        }

                    }

                }

            }

            prettyPrint(array, row, col, pw);

            pw2.print(row);

            pw2.print(' ');

            pw2.print(col);

            pw2.print(' ');

            pw2.print(getMin(array, row, col));

            pw2.print(' ');

            pw2.print(getMax(array, row, col));

            pw2.println();

            for(int v = 1; v < row + 1; v++){

                for (int w = 1; w < col + 1; w++){

                    pw2.print(array[v][w]);

                    pw2.print(' ');

                }

                pw2.println();

            }

            pw.println();

            pw.print("EQAry");

            pw.println();

            pw.print("-----");

            pw.println();

            for (int s = 1; s <= count; s++){

                pw.print(uniqueEQAry[s]);

                pw.print(' ');

            }

            pw.println();

            pw.println();

            pw.close();

            pw2.close();

        }catch(Exception e){

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

            }

    }

    public static void printCCProperty(int [][] array, int [] EQAry, int row, int col, int count, File out){

        int [] uniqueEQAry = toUniqueArray(EQAry);

        int length = uniqueEQAry.length;

        int [] data = new int [length];

        try{

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

            pw.print(row);

            pw.print(' ');

            pw.print(col);

            pw.print(' ');

            pw.print(getMin(array, row, col));

            pw.print(' ');

            pw.print(getMax(array, row, col));

            pw.println();

            pw.print(count);

            pw.println();

            pw.println();

            for (int component = 1; component <= count; component++){

                int counter = 0;

                for (int i = 1; i < row + 1; i++){

                    for (int j = 1; j < col + 1; j++){

                        if (array[i][j] == uniqueEQAry[component]){

                            counter++;

                        }

                    }

                }

                data[component] = counter;

            }

            int ro = 1;

            int co = 1;

            for (int c = 1; c <= count; c++){

                pw.print("Label: ");

                pw.print(c);

                pw.println();

                pw.print("Pixels: ");

                pw.print(data[c]);

                pw.println();

                pw.println();

                int comp = c;

                while ((ro < row + 1) && (co < col + 1)){

                    pw.print("min row: ");

                    pw.print(ro);

                    pw.println();

                    pw.print("min col: ");

                    pw.print(co);

                    pw.println();

                    while(array[ro][co] == comp){

                        ro++;

                    }

                    while(array[ro][co] == comp){

                        co++;

                    }

                    pw.print("max row: ");

                    pw.print(ro);

                    pw.println();

                    pw.print("max col: ");

                    pw.print(co);

                    pw.println();

                    pw.println();

                    break;

                }

            }

            pw.close();

        }

        catch(Exception e){

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

        }

    }

    public static void prettyPrint(int [][] array, int row, int col, PrintWriter pw){

        for(int x = 1; x < row + 1; x++){

            for (int y = 1; y < col + 1; y++){

                if (array[x][y] > 0){

                    pw.print(array[x][y]);

                    pw.print(' ');

                }

                else {

                    pw.print(' ');

                    pw.print(' ');

                }

            }

            pw.println();

        }

    }

    public static void updateEQAry(int [] EQAry, int size, int position, int value){

        EQAry[position] = value;

    }

    public static int[] manageEQAry(int [] EQAry, int size, File out){

        int count = 0;

        int [] array2 = new int [size + 2];

        for (int i = 1; i <= size; i++) array2[i] = EQAry[i];

        try{

            PrintWriter pw = new PrintWriter(new FileWriter(out, true));

            pw.print("--------------------------------------------------");

            pw.println();

            pw.println();

            pw.print("ManageEQAry");

            pw.println();

            pw.print("-----------");

            pw.println();

            int counter = 1;

            for (int i = 1; i <= size; i++){

                if (EQAry[i] == i) {

                    EQAry[i] = ++count;

                }

                else {

                    EQAry[i] = EQAry[EQAry[i]];

                }

            }

            for (int j = 1; j <= size; j++){

                pw.print(EQAry[j]);

                pw.print(' ');

            }

            pw.println();

            pw.println();

            //pw.print(count);

            pw.close();

        }

        catch(Exception e){

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

        }

        array2[0] = count;

        return array2;

    }

}

Output













