CV  **ChainCode** Java

**Student:** Shuhua Song

**Project Due Date:**04/30/2021

1. Algorithm Steps for Chain-Code Algorithm (inArray, outAry):

Step 0: Image <- given a binary image (a Connected Component Box)

Output <- open Chain-Code output file

Step 1: scan image left->right & top->bottom

startingP(i,j) <- next pixel

Step 2: if startingP(i, j) > 0

startRow <- i

startCol <- j

gray-scale <- startingP(i, j)

currentP(i, j) <- startingP(i, j)

lastZero <- 4

output <- startRow, startCol, gray-scale

Step 2.1: repeat steps1 to 2 until startP(i, j) is found(If found break)

Step 4: direction <- lastZero + 1 // from 0-7 (++lastZert % 8)

Step 5: nextP(i, j) <- findNextPixel(direction, currentP(i, j))

Step 1: if nextP(i, j) ==0

direction = ++ direction%8

Step 2: repeat steps1 until nextP(I,j) > 0

Step 6: output <- direction

currentP(i, j) <- nextP(i, j)

lastZero <- zeroTable[direction-1]

Step 7: repeat step 4 to 6 until you reach the startingP(i, j)

**-The hand tracing of Img1CC and write the traced chain-code.**

Text

Description automatically generated

**- Source Code**

import java.io.\*;  
import java.util.Scanner;  
import java.util.regex.Pattern;  
  
public class Image{  
 int numRows, numCols;  
 int minVal, maxVal;  
 int[][] imageAry;  
 int[][] boundAry;  
 int[][] CCAry;  
  
 public Image() { }  
  
 public Image(int numRows, int numCols, int minVal, int maxVal) {  
 this.numRows = numRows;  
 this.numCols = numCols;  
 this.minVal = minVal;  
 this.maxVal = maxVal;  
  
 imageAry = new int[numRows + 2][numCols + 2];  
 boundAry = new int[numRows + 2][numCols + 2];  
 CCAry = new int[numRows + 2][numCols + 2]; //process the chain code of each C.C  
 }  
  
 public void zeroFramed(int[][] inArray) {  
 for (int i = 0; i < numRows + 2; i++) {  
 inArray[i][0] = 0;  
 inArray[i][numCols + 1] = 0;  
 }  
 for (int j = 0; j < numCols + 2; j++) {  
 inArray[0][j] = 0;  
 inArray[numRows + 1][j] = 0;  
 }  
 }  
  
 public void zero2DAry(int[][] inArray) {  
 for (int i = 0; i < inArray.length; i++) {  
 for (int j = 0; j < inArray[0].length; j++) {  
 inArray[i][j] = 0;  
 }  
 }  
 }  
  
 public void loadImage(Scanner inFile, int[][] inArray, BufferedWriter outFile) throws IOException {  
 for (int i = 1; i < numRows + 1; i++) {  
 for (int j = 1; j < numCols + 1; j++) {  
 inArray[i][j] = inFile.nextInt();  
 outFile.write(inArray[i][j] + " ");  
 }  
 outFile.write("\n");  
 }  
 }  
  
 class CCproperty{  
 //int numCC;  
 public int label, numPixels;  
 public int minRow, minCol;  
 public int maxRow, maxCol;  
 public CCproperty[] CCs;  
 public CCproperty(){  
 this.label = 0;  
 this.numPixels = 0;  
 this.minRow = 0;  
 this.minCol = 0;  
 this.maxRow = 0;  
 this.maxCol = 0;  
 }  
  
 public void clearCCAry(int[][] CCAry){  
 for (int i = 0; i < CCAry.length; i++) {  
 for (int j = 0; j < CCAry[0].length; j++) {  
 CCAry[i][j] = 0;  
 }  
 }  
 }  
 public void loadCCAry(int label, int[][] CCAry, int[][] imageAry){  
 for (int i = 0; i < CCAry.length; i++) {  
 for (int j = 0; j < CCAry[0].length; j++) {  
 if(imageAry[i][j]==label){  
 CCAry[i][j] = label;  
 }  
 }  
 }  
 }  
 }  
  
 class ChainCode{  
 class Point{  
 int row, col;  
 public Point(){}  
 public Point(int row, int col){  
 this.row = row;  
 this.col = col;  
 }  
 }  
 Point[] neighborCoord = new Point[8];  
 int[] zeroTable = new int[]{6, 0, 0, 2, 2, 4, 4, 6};  
 Point startP = null;  
 Point currentP = null; //current none zero border pixel  
 Point nextP = null; //next none-zero border pixel  
 int lastQ; //Range from 0 to 7; it is the direction of the last zero scanned from currentP  
 int nextQ; // the next scanning direction of currentP's neighbors  
 // to find nextP, range from 0 to 7, need to mod 8.  
 int PchainDir; // chain code direction from currentP to nextP  
  
 public ChainCode(){}  
 Point[] loadNeighbors(int i, int j, int[][] CCAry){  
 int k=0;  
 if(i>=1 && i<numRows+1 && j>=1 && j<numCols+1){  
 neighborCoord[k++] = new Point(i, j+1);  
 neighborCoord[k++] = new Point(i-1, j+1);  
 neighborCoord[k++] = new Point(i-1, j);  
 neighborCoord[k++] = new Point(i-1, j-1);  
 neighborCoord[k++] = new Point(i, j-1);  
 neighborCoord[k++] = new Point(i+1, j-1);  
 neighborCoord[k++] = new Point(i+1, j);  
 neighborCoord[k++] = new Point(i+1, j+1);  
 }  
 return neighborCoord;  
 }  
  
  
 public int findNextP(Point currentP, int nextQ, Point nextP, int[][] CCAry){  
 int x = currentP.row;  
 int y = currentP.col;  
 neighborCoord = loadNeighbors(x, y, CCAry);  
 int r = 0, c = 0, chainDir = 0;  
 for(int loop=0; loop<8; loop++){  
 //i = i >= 8 ? i%8 : i;  
 nextQ = nextQ%8;  
 r = neighborCoord[nextQ].row;  
 c = neighborCoord[nextQ].col;  
 if(CCAry[r][c] < 0){  
 CCAry[r][c] = -CCAry[r][c];  
 }  
 if(CCAry[r][c] >0 ){  
 // nextP = neighborCoord[i];  
 chainDir = nextQ;  
 break;  
 }else{  
 nextQ++;  
 }  
 }  
 //System.out.println("chainDir = " + chainDir);  
 return chainDir;  
 }  
  
  
 public void getChainCode(Image.CCproperty CC, int[][] CCAry, BufferedWriter outFile) throws IOException {  
 int label = CC.label;  
 int minRow = CC.minRow;  
 int minCol = CC.minCol;  
 int maxRow = CC.maxRow;  
 int maxCol = CC.maxCol;  
 int startRow = 0, startCol = 0;  
 int currRow = -1, currCol = -1;  
 boolean isFound = false;  
 for(int i=0; i<numRows+2; i++){  
 for(int j=0; j<numCols+2; j++){  
 if(CCAry[i][j]==label){  
 startRow = i;  
 startCol = j;  
 startP = new Point(i, j);  
 currentP = new Point(i, j);  
 lastQ = 4;  
 // System.out.println(label + " " + startRow + " " + startCol);  
 outFile.write(label + " " + startRow + " " + startCol + " ");  
 isFound = true;  
 break;  
 }  
 }  
 if(isFound){  
 break;  
 }  
 }  
 while(currentP != startP){  
 //System.out.println("x = " + currRow + ", y = " + currCol);  
 nextQ = Math.*floorMod*(lastQ+1, 8);  
 PchainDir = findNextP(currentP, nextQ, nextP, CCAry);  
 nextP = neighborCoord[PchainDir];  
 if(CCAry[currentP.row][currentP.col] > 0){  
 CCAry[currentP.row][currentP.col] = -CCAry[currentP.row][currentP.col];  
 }  
 //System.out.println( PchainDir + " ");  
 outFile.write(PchainDir + " ");  
 if(PchainDir==0){  
 PchainDir = 8;  
 }  
 lastQ = zeroTable[PchainDir-1];  
 // System.out.println("lastQ = " + lastQ);  
 currentP = nextP;  
 //new Point(nextP.row, nextP.col);  
 currRow = currentP.row;  
 currCol = currentP.col;  
 //System.out.println("x = " + currRow + ", y = " + currCol);  
 if(currRow==startRow && currCol==startCol){  
 break;  
 }  
 }  
 }  
  
 public void construcBoundary(int[][] boundAry, String chainCodeFileName, BufferedWriter outFile, int numCC) throws IOException {  
 Scanner chainCodeFile = new Scanner(new FileReader(chainCodeFileName));  
 int numRows = chainCodeFile.nextInt();  
 int numCols = chainCodeFile.nextInt();  
 int minVal = chainCodeFile.nextInt();  
 int maxVal = chainCodeFile.nextInt();  
 outFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");  
 String line = "";  
 while(chainCodeFile.hasNextLine()){  
 line = chainCodeFile.nextLine();  
 //System.out.println(line);  
 if(line.length()>0){  
 String[] strs = line.split("\\s+");  
 int[] numbers = new int[strs.length];  
 for(int i=0; i<strs.length; i++){  
 numbers[i] = Integer.*parseInt*(strs[i]);  
 //System.out.print(numbers[i] + " ");  
 }  
 int label = numbers[0];  
 int startRow = numbers[1];  
 int startCol = numbers[2];  
 //System.out.println();  
 int nextDir = 0;  
 boundAry[startRow][startCol] = label;  
 int x = startRow, y = startCol;  
 Point currentP = new Point(startRow, startCol);  
 Point nextP = null;  
 //System.out.println(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");  
 //System.out.println(label + " " + startRow + " " + startCol + " ");  
 for(int i=3; i<numbers.length; i++){  
 nextDir = numbers[i];  
 //System.out.println(nextDir + " ");  
 neighborCoord = loadNeighbors(x, y, boundAry);  
 nextP = neighborCoord[nextDir];  
 currentP = nextP;  
 x = currentP.row;  
 y = currentP.col;  
 boundAry[x][y] = label;  
 }  
 }  
 line = "";  
 }  
 outFile.write("\n");  
 for(int k=0; k<numRows+2; k++){  
 for(int j=0; j<numCols+2; j++){  
 // System.out.print(boundAry[k][j] + " ");  
 outFile.write(boundAry[k][j] + " ");  
 }  
 // System.out.println();  
 outFile.write("\n");  
 }  
 }  
 }  
  
 public static void main(String[] args) throws IOException {  
 Scanner labelFile = new Scanner(new FileReader(args[0]));  
 Scanner propFile = new Scanner(new FileReader(args[1]));  
 //read header from labelFile  
 int numRows = labelFile.nextInt();  
 int numCols = labelFile.nextInt();  
 int minVal = labelFile.nextInt();  
 int maxVal = labelFile.nextInt();  
 //read header from property File  
 numRows = propFile.nextInt();  
 numCols = propFile.nextInt();  
 minVal = propFile.nextInt();  
 maxVal = propFile.nextInt();  
 int numCC = propFile.nextInt();  
 //System.out.println(numRows + " " + numCols + " " + minVal + " " + maxVal + " " + numCC);  
  
 //Create output File on the run time according to input File  
 String imgName = args[0].substring(0, 4);  
 String chainCodeFileName = imgName + "\_chainCode.txt";  
 String boundaryFileName = imgName + "\_boundary.txt";  
 String printImageName = imgName + "\_output.txt";  
 BufferedWriter chainCodeFile = new BufferedWriter(new FileWriter(new File(chainCodeFileName)));  
 BufferedWriter boundaryFile = new BufferedWriter(new FileWriter(new File(boundaryFileName)));  
 BufferedWriter printImageFile = new BufferedWriter(new FileWriter(new File(printImageName)));  
 chainCodeFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");  
  
 //Load Original Image to imageAry  
 Image image = new Image(numRows, numCols, minVal, maxVal);  
 int[][] imageAry = image.imageAry;  
 int[][] boundAry = image.boundAry;  
  
 printImageFile.write("Print " + imgName + "CC " + "\n");  
 printImageFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");  
 image.loadImage(labelFile, imageAry, printImageFile);  
 int[][] CCAry = image.CCAry;  
  
 Image.CCproperty ccProp = image.new CCproperty();  
 Image.ChainCode chaincode = image.new ChainCode();  
 Image.CCproperty[] CCs = new CCproperty[numCC];  
 //Loop through every Connected Component to get ChainCode  
  
 printImageFile.write("\nPrint "+ imgName + " Property:" +"\n");  
 printImageFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");  
 printImageFile.write(numCC + "\n");  
 for(int i=0; i<numCC; i++){  
 chainCodeFile.write("\n");  
 CCs[i] = image.new CCproperty();  
 CCs[i].label = propFile.nextInt();  
 CCs[i].numPixels = propFile.nextInt();  
 CCs[i].minRow = propFile.nextInt();  
 CCs[i].minCol = propFile.nextInt();  
 CCs[i].maxRow = propFile.nextInt();  
 CCs[i].maxCol = propFile.nextInt();  
 /\*System.out.println(CCs[i].label + " " + CCs[i].numPixels + " " +  
 CCs[i].minRow + " " + CCs[i].minCol + " " +  
 CCs[i].maxRow + " " + CCs[i].maxCol);\*/  
 printImageFile.write(CCs[i].label + "\n");  
 printImageFile.write(CCs[i].numPixels + "\n");  
 printImageFile.write(CCs[i].minRow + " " + CCs[i].minCol + "\n");  
 printImageFile.write(CCs[i].maxRow + " " + CCs[i].maxCol + "\n");  
 ccProp.loadCCAry(CCs[i].label, CCAry, imageAry);  
 /\*for(int k=0; k<numRows+2; k++){  
 for(int j=0; j<numCols+2; j++){  
 System.out.print(CCAry[k][j] + " ");  
 }  
 System.out.println();  
 }\*/  
 chaincode.getChainCode(CCs[i], CCAry, chainCodeFile);  
 }  
 chainCodeFile.close();  
 //re-open chainCode file and construct the boundary image  
 chaincode.construcBoundary(boundAry, chainCodeFileName, boundaryFile, numCC);  
 printImageFile.close();  
 labelFile.close();  
 propFile.close();  
 boundaryFile.close();  
 }  
}  
  
/\*  
An instance of an inner class cannot be created without an instance of the outer class.  
\*/

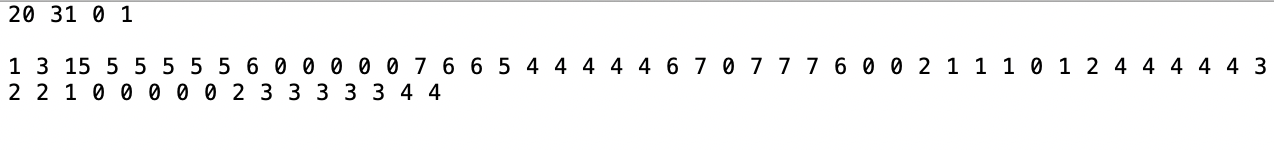
**- Print img1CC and img1Property**

**Background pattern

Description automatically generated** **A picture containing graphical user interface

Description automatically generated**

**- Print ChainCodeFile for Img1CC**

****

**- Print Boundary file for img1CC**

**Background pattern

Description automatically generated**

**- Print img2CC and img2Property**

Background pattern

Description automatically generated

Text

Description automatically generated

**- Print ChainCodeFile for Img2CC**

A picture containing shape

Description automatically generated

**- Print Boundary file for img2CC**

Background pattern

Description automatically generated