CV Project 7: Chain Code for Image Compression Java

**Student:** Pawan Bhatta

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

**Chain Code Algorithm Steps:**

Step 0: image ← given a Binary Image e.g. 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 // starting Pixel found

startRow ← i // row

startCol ← j // column

gray-scale ← startingP( i,j ) // pixel value

currentP( i,j ) ← startingP( i,j )

lastZero ← 4 //starts with 4 b/c we scan L→R

output ← startRow , startCol , gray-scale

Step 3: repeat steps 1 to 2 until startingP( i,j ) is found (if found break)

Step 4: direction ← lastZero + 1 // # from 0 to 7 (++lastZero % 8)

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

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

direction = ++direction % 8

Step 2: repeat steps 1 until nextP( i,j ) > 0.

Step 6: output ← direction // direction of nextP( i,j ) set by findNextPixel()

// direction is Chain-Code Link

currentP( i,j ) ← nextP( i,j )

lastZero ← ZeroTable[ direction - 1 ]

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

**Finding Next Point based on current point and direction value:**

Step 1: loadNeighborCoord (currentP)

Step 2: chainDir← scan currentP's 8 neighbors counter clockwise from nextQ direction (mod 8) until a none zero neighbor with the same label as currentCC is found. The row and col of each of the 8 neighbors are stored in neighborCoord [].

Step 3: returns chainDir

**Source Code:**

import java.io.\*;

import java.util.Scanner;

class Image {

int numRows, numCols, minVal, maxVal;

int[][] imageAry;

int[][] boundaryAry;

int[][] CCAry;

Image(Scanner imgFile) {

loadHeader(imgFile);

zeroFrameImageAry();

loadImage(imgFile);

}

void zeroFrameImageAry() {

imageAry = new int[numRows + 2][numCols + 2];

}

void loadHeader(Scanner imgFile) {

numRows = imgFile.nextInt();

numCols = imgFile.nextInt();

minVal = imgFile.nextInt();

maxVal = imgFile.nextInt();

}

void loadImage(Scanner imgFile) {

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

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

imageAry[i][j] = imgFile.nextInt();

}

}

}

void writeHeader(BufferedWriter outFile) throws IOException {

outFile.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");

}

void prettyPrint(BufferedWriter outFile) throws IOException {

writeHeader(outFile);

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

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

if (imageAry[i][j] == 0) {

outFile.write(". ");

} else {

outFile.write(Integer.toString(imageAry[i][j]) + " ");

}

}

outFile.write("\n");

}

}

void printBoundaryAry(BufferedWriter outFile) throws IOException {

writeHeader(outFile);

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

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

outFile.write(Integer.toString(boundaryAry[i][j]) + " ");

}

outFile.write("\n");

}

}

void prettyPrintBoundaryAry(BufferedWriter outFile) throws IOException {

writeHeader(outFile);

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

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

if (boundaryAry[i][j] == 0) {

outFile.write(". ");

} else {

outFile.write(Integer.toString(boundaryAry[i][j]) + " ");

}

}

outFile.write("\n");

}

}

// Give the chainCode file, create an image contains only the boundary of

// objects in the labelled file

void constructBoundary(Scanner chainCodeFile) {

numRows = chainCodeFile.nextInt();

numCols = chainCodeFile.nextInt();

minVal = chainCodeFile.nextInt();

maxVal = chainCodeFile.nextInt();

boundaryAry = new int[numRows][numCols];

// initializing whole array to zero

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

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

boundaryAry[i][j] = 0;

}

}

// reading the chain Code to put pixel values

while (chainCodeFile.hasNextInt()) {

int pixelVal = chainCodeFile.nextInt();

Point startP = new Point(chainCodeFile.nextInt(), chainCodeFile.nextInt());

boundaryAry[startP.row][startP.col] = pixelVal;

Point currentP = getNextP(startP, chainCodeFile.nextInt());

while (!currentP.equals(startP)) {

boundaryAry[currentP.row][currentP.col] = pixelVal;

currentP = getNextP(currentP, chainCodeFile.nextInt());

}

}

}

Point getNextP(Point currentP, int direction) {

Point returnVal = new Point(0, 0);

switch (direction) {

case 0:

returnVal.update(currentP.row, currentP.col + 1);

break;

case 1:

returnVal.update(currentP.row - 1, currentP.col + 1);

break;

case 2:

returnVal.update(currentP.row - 1, currentP.col);

break;

case 3:

returnVal.update(currentP.row - 1, currentP.col - 1);

break;

case 4:

returnVal.update(currentP.row, currentP.col - 1);

break;

case 5:

returnVal.update(currentP.row + 1, currentP.col - 1);

break;

case 6:

returnVal.update(currentP.row + 1, currentP.col);

break;

case 7:

returnVal.update(currentP.row + 1, currentP.col + 1);

break;

default:

break;

}

return returnVal;

}

}

class CCproperty {

int numCC, label, numPixels, minRow, minCol, maxRow, maxCol;

int[][] CCAry;

CCproperty(Scanner propImgFile) {

int numRows = propImgFile.nextInt();

int numCols = propImgFile.nextInt();

int minVal = propImgFile.nextInt();

int maxVal = propImgFile.nextInt();

numCC = propImgFile.nextInt();

}

void clearCCAry() {

for (int i = 0; i < maxRow - minRow; i++) {

for (int j = 0; j < maxCol - minCol; j++) {

CCAry[i][j] = 0;

}

}

}

void loadCCAry(Scanner propImg, int[][] imgAry) {

label = propImg.nextInt();

numPixels = propImg.nextInt();

minRow = propImg.nextInt();

minCol = propImg.nextInt();

maxRow = propImg.nextInt();

maxCol = propImg.nextInt();

// Initializing CCAry according to the current CC label

CCAry = new int[maxRow - minRow + 1 + 2][maxCol - minCol + 1 + 2];

clearCCAry();

// Copying all the pixel values of a given CC by using bounding box's values

for (int i = 1; i < maxRow - minRow + 1 + 1; i++) {

for (int j = 1; j < maxCol - minCol + 1 + 1; j++) {

CCAry[i][j] = imgAry[i + minRow][j + minCol];

}

}

}

void prettyPrint(BufferedWriter outFile) throws IOException {

outFile.write("\n");

for (int i = 1; i < maxRow - minRow + 1 + 1; i++) {

for (int j = 1; j < maxCol - minCol + 1 + 1; j++) {

if (CCAry[i][j] == 0) {

outFile.write(". ");

} else {

outFile.write(Integer.toString(CCAry[i][j]) + " ");

}

}

outFile.write("\n");

}

}

}

class Point {

int row, col;

Point(int i, int j) {

row = i;

col = j;

}

void update(int i, int j) {

row = i;

col = j;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if ((obj == null) || (obj.getClass() != this.getClass()))

return false;

// object must be Point at this point

Point p = (Point) obj;

return (row == p.row) && (col == p.col);

}

@Override

public String toString() {

return "(" + row + "," + col + ")";

}

}

class ChainCode {

Point[] neighborCoord;

int[] zeroTable;

Point startP;

Point currentP;// current none zero border pixel

Point nextP;// next none-zero border pixel

int lastQ; // Range from 0 to 7; it is the direction of the last zero scanned from currentP

int nextDir;// the next scanning direction of currentP's neighbors

int pChainDir; // chain code direction from currentP to nextP

CCproperty ccProp;

ChainCode() {

zeroTable = new int[] { 6, 0, 0, 2, 2, 4, 4, 6 };

}

void getChainCode(CCproperty cc, BufferedWriter chainCodeFile) throws IOException {

chainCodeFile.write("\n");

ccProp = cc;

startP = new Point(1, 1);

currentP = new Point(1, 1);

lastQ = 4;

outerloop: for (int i = 1; i < cc.maxRow - cc.minRow + 1 + 1; i++) {

for (int j = 1; j < cc.maxCol - cc.minCol + 1 + 1; j++) {

if (cc.CCAry[i][j] == cc.label) {

startP.row = i;

startP.col = j;

currentP.row = startP.row;

currentP.col = startP.col;

lastQ = 4;

chainCodeFile.write(cc.label + " " + (cc.minRow + i - 1) + " " + (cc.minCol + j - 1) + " ");

break outerloop;

}

}

}

// at this point we will get our startingPoint

int count = 0;

while (count == 0 || !currentP.equals(startP)) {

count++;

loadNeigborsCoord(currentP);

nextDir = ++lastQ % 8;

pChainDir = findNextP(nextDir, currentP);

nextP = new Point(neighborCoord[pChainDir].row, neighborCoord[pChainDir].col);

ccProp.CCAry[nextP.row][nextP.col] = (-1) \* ccProp.CCAry[nextP.row][nextP.col];

chainCodeFile.write(pChainDir + " ");

if (pChainDir == 0) {

lastQ = zeroTable[7];

} else {

lastQ = zeroTable[pChainDir - 1];

}

currentP.row = nextP.row;

currentP.col = nextP.col;

}

}

// Given currentP's row and col, the method determines and stores the row and

// col of each of currentP's

// 8 neighbors (0 to 7 w.r.t the chain-code direction) in neighborCoord[] array.

void loadNeigborsCoord(Point currentP) {

neighborCoord = new Point[8];

neighborCoord[0] = new Point(currentP.row, currentP.col + 1);

neighborCoord[1] = new Point(currentP.row - 1, currentP.col + 1);

neighborCoord[2] = new Point(currentP.row - 1, currentP.col);

neighborCoord[3] = new Point(currentP.row - 1, currentP.col - 1);

neighborCoord[4] = new Point(currentP.row, currentP.col - 1);

neighborCoord[5] = new Point(currentP.row + 1, currentP.col - 1);

neighborCoord[6] = new Point(currentP.row + 1, currentP.col);

neighborCoord[7] = new Point(currentP.row + 1, currentP.col + 1);

}

int findNextP(int direction, Point p) {

int i = p.row;

int j = p.col;

int loop = 0;

while (loop < 8) {

switch (direction) {

case 0:

if (ccProp.CCAry[i][j + 1] > 0 || ccProp.CCAry[i][j + 1] == -1)

return 0;

break;

case 1:

if (ccProp.CCAry[i - 1][j + 1] > 0 || ccProp.CCAry[i - 1][j + 1] == -1)

return 1;

break;

case 2:

if (ccProp.CCAry[i - 1][j] > 0 || ccProp.CCAry[i - 1][j] == -1)

return 2;

break;

case 3:

if (ccProp.CCAry[i - 1][j - 1] > 0 || ccProp.CCAry[i - 1][j - 1] == -1)

return 3;

break;

case 4:

if (ccProp.CCAry[i][j - 1] > 0 || ccProp.CCAry[i][j - 1] == -1)

return 4;

break;

case 5:

if (ccProp.CCAry[i + 1][j - 1] > 0 || ccProp.CCAry[i + 1][j - 1] == -1)

return 5;

break;

case 6:

if (ccProp.CCAry[i + 1][j] > 0 || ccProp.CCAry[i + 1][j] == -1)

return 6;

break;

case 7:

if (ccProp.CCAry[i + 1][j + 1] > 0 || ccProp.CCAry[i + 1][j + 1] == -1)

return 7;

break;

default:

break;

}

direction = ++direction % 8;

loop++;

}

return 0;

}

public static void main(String[] args) throws IOException {

String labelFileName = args[0] + ".txt";

FileReader labelFileReader = null;

BufferedReader labelFileBuffReader = null;

Scanner labelFile = null;

String propFileName = args[1] + ".txt";

FileReader propFileReader = null;

BufferedReader propFileBuffReader = null;

Scanner propFile = null;

String chainCodeFileName = args[0] + "\_chainCode.txt";

FileWriter chainCodeFileWriter = null;

BufferedWriter chainCodeFile = null;

String boundaryFileName = args[0] + "\_Boundary.txt";

FileWriter boundaryFileWriter = null;

BufferedWriter boundaryFile = null;

String chainCodeInputFileName = args[0] + "\_chainCode.txt";

FileReader chainCodeInputReader = null;

BufferedReader chainCodeInputBuffReader = null;

Scanner chainCodeInput = null;

try {

labelFileReader = new FileReader(labelFileName);

labelFileBuffReader = new BufferedReader(labelFileReader);

labelFile = new Scanner(labelFileBuffReader);

propFileReader = new FileReader(propFileName);

propFileBuffReader = new BufferedReader(propFileReader);

propFile = new Scanner(propFileBuffReader);

chainCodeFileWriter = new FileWriter(chainCodeFileName);

chainCodeFile = new BufferedWriter(chainCodeFileWriter);

boundaryFileWriter = new FileWriter(boundaryFileName);

boundaryFile = new BufferedWriter(boundaryFileWriter);

Image img = new Image(labelFile);

// img.prettyPrint(chainCodeFile);

CCproperty ccProp = new CCproperty(propFile);

img.writeHeader(chainCodeFile);

for (int i = 0; i < ccProp.numCC; i++) {

ccProp.loadCCAry(propFile, img.imageAry);

ChainCode chainCode = new ChainCode();

// ccProp.prettyPrint(chainCodeFile);

chainCode.getChainCode(ccProp, chainCodeFile);

}

// Closing chain Code file

if (chainCodeFile != null)

chainCodeFile.close();

// Reopening the Chain Code file

chainCodeInputReader = new FileReader(chainCodeInputFileName);

chainCodeInputBuffReader = new BufferedReader(chainCodeInputReader);

chainCodeInput = new Scanner(chainCodeInputBuffReader);

img.constructBoundary(chainCodeInput);

img.printBoundaryAry(boundaryFile);

img.prettyPrintBoundaryAry(boundaryFile);

if (chainCodeInput != null) {

chainCodeInput.close();

}

} finally {

if (labelFile != null)

labelFile.close();

if (propFile != null)

propFile.close();

if (boundaryFile != null)

boundaryFile.close();

}

}

}

**Outputs**

**Image\_1:**

Labeled Image

20 31 0 1

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 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 1 1 1 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 1 1 1 1 1 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 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 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 1 1 1 1 1 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 1 1 1 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 1 1 1 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 0

Property File

20 31 0 1

1

1

119

2 9

18 21

Result of: Chain Coding

20 31 0 1

1 2 14 5 5 5 5 5 6 0 0 0 0 0 7 6 6 5 4 4 4 4 4 6 7 0 7 7 7 6 0 0 2 1 1 1 0 1 2 4 4 4 4 4 3 2 2 1 0 0 0 0 0 2 3 3 3 3 3 4 4

Result of: Boundary Construction from the above Chain Code

20 31 0 1

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 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 1 1 1 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 1 0 0 0 1 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 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 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 1 0 0 0 1 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 1 0 1 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 1 1 1 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 0

20 31 0 1

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . 1 1 1 . . . . . . . . . . . . . .

. . . . . . . . . . . . . 1 . . . 1 . . . . . . . . . . . . .

. . . . . . . . . . . . 1 . . . . . 1 . . . . . . . . . . . .

. . . . . . . . . . . 1 . . . . . . . 1 . . . . . . . . . . .

. . . . . . . . . . 1 . . . . . . . . . 1 . . . . . . . . . .

. . . . . . . . . 1 . . . . . . . . . . . 1 . . . . . . . . .

. . . . . . . . . 1 1 1 1 1 1 . 1 1 1 1 1 1 . . . . . . . . .

. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . .

. . . . . . . . . 1 1 1 1 1 1 . 1 1 1 1 1 1 . . . . . . . . .

. . . . . . . . . 1 . . . . . . . . . . . 1 . . . . . . . . .

. . . . . . . . . . 1 1 . . . . . . . 1 1 . . . . . . . . . .

. . . . . . . . . . . . 1 . . . . . 1 . . . . . . . . . . . .

. . . . . . . . . . . . . 1 . . . 1 . . . . . . . . . . . . .

. . . . . . . . . . . . . . 1 . 1 . . . . . . . . . . . . . .

. . . . . . . . . . . . . . 1 1 1 . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Image\_2:**

Labeled Image

20 40 0 3

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 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0

0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 3 3 3 3 0 3 3 3 3 3 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 3 3 3 3 0 0 3 3 3 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Property File

20 40 0 3

3

1

172

2 4

19 20

2

73

2 25

10 35

3

68

12 23

19 37

Result of: Chain Coding

20 40 0 3

1 2 7 5 4 5 7 0 7 5 4 4 6 6 6 6 6 6 6 7 7 7 0 7 6 1 1 1 0 0 7 0 7 7 1 3 2 1 0 2 2 2 3 3 2 3 3 2 2 6 6 5 5 5 5 5 3 3 2 2 2 2 2 2 2 4 3

2 2 29 5 5 5 5 6 6 7 7 0 0 0 0 0 0 1 1 2 2 3 3 3 3 4 4

3 12 23 7 7 5 5 6 6 7 2 1 0 2 1 7 7 0 0 1 1 1 0 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4

Result of: Boundary Construction from the above Chain Code

20 40 0 3

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 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 2 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 2 0 0 0 0 0 0

0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 0

0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0

0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0

0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0

0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 0

0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 3 0 0 0 0 0 0 0 0 3 3 0 0 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 3 0 0 3 0 0 0 0 0 3 0 0 0 0 0 0

0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 3 0 0 3 0 3 0 0 0 3 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 3 0 3 3 0 0 3 3 3 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

20 40 0 3

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . 2 2 2 . . . . . . . .

. . . . . 1 1 . 1 1 . . . . . . . . . . . . . . . . . . 2 . . . 2 . . . . . . .

. . . . 1 . . . . 1 . . . . . . . . . . . . . . . . . 2 . . . . . 2 . . . . . .

. . . . . 1 1 . . 1 . . . . . . 1 . . . . . . . . . 2 . . . . . . . 2 . . . . .

. . . . . . . 1 . 1 . . . . . . 1 . . . . . . . . 2 . . . . . . . . . 2 . . . .

. . . . 1 1 1 . . 1 . . . . . . 1 . . . . . . . . 2 . . . . . . . . . 2 . . . .

. . . . 1 . . . . 1 . . . . . 1 . 1 . . . . . . . 2 . . . . . . . . . 2 . . . .

. . . . 1 . . . . 1 . . . . 1 . . . 1 . . . . . . . 2 . . . . . . . 2 . . . . .

. . . . 1 . . . . 1 . . . 1 . . . . 1 . . . . . . . . 2 2 2 2 2 2 2 . . . . . .

. . . . 1 . . . . . 1 . 1 . . . . . . 1 . . . . . . . . . . . . . . . . . . . .

. . . . 1 . . . . . . 1 . . . . . . . . 1 . . 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 . .

. . . . 1 . . . . . . . . . . . . . . . 1 . . . 3 . . . . . . . . . . . 3 . . .

. . . . 1 . . . . . . . . . . . . . . . 1 . . . . 3 . . . . . . . . 3 3 . . . .

. . . . . 1 . . . . . . . . . . . . . 1 1 . . . 3 . . 3 . . . . . 3 . . . . . .

. . . . . . 1 . . . . . 1 1 1 . . . 1 . . . . 3 . . 3 . 3 . . . 3 . . . . . . .

. . . . . . . 1 1 . . 1 . . . 1 1 . 1 . . . . 3 . 3 3 . . 3 3 3 . . . . . . . .

. . . . . . . . . 1 1 . . . . . . 1 . 1 . . . 3 3 . . . . . . . . . . . . . . .

. . . . . . . . . 1 . . . . . . . . 1 . . . . . 3 . . . . . . . . . . . . . . .

**Hand Tracing to check the validity of chain code produced for Data\_1 above:**

**A piece of paper with writing on it

Description automatically generated with medium confidence**