JAVA

CV

**Chain Code Algorithm:**

Step 0: Process each object 1 by one. Get the label of the current Object and have empty the dimensional Array for every object then place each object to the empty 2D array. Send it in chain code method

Chain code method:

Step 1:Have a outer for loop that goes over each rows and have inner for loop goes over each columns.

Step2: In inner for loop iterate through each pixel of the array. When current pixel is equal to the label you make the start P’s row equal to the current row and start P’s column equal to the current column. Then set current P equal to start P. Then set last Q = 4.

Step 3: after you found the pixel in step 2 you exit out of the double for loop

Step 4: let next Q = (lastQ + 1) mod 8

Step 5: let PchainDir = findNextP(Ary, currP, nextQ)

Step 6: nextP = neighborCoord[PchainDir]

Step7: write the chain code to the file by writing the value of PchainDir

Step 8: if PchainDir is equal to zero then set last Q = zeroTable[7]. Else set last Q = zeroTable[PchainDir – 1]

Step 9: set currP equal to nextP

Step 10: Have a loop that will repeat step 4 through 9. That ends when startP is equal to currP.

**findNextP Algorithm:**

Step1: loadNeighborCoord(currP) // this will load the current Pixels to a neighbor cooridinate array

Step2: create a integer variable for row and column. Create a temp constructor of class Point.

Note: class Point has two variables row and column.

Step 3 set row = temp.row and column = temp.col

Step 4: while Ary[row][col] is not greater than 0. Set temp = new point().

Step 5: nextQ = (++nextQ) mod 8

Step 6 : repeat step 3

Step 7: after the whole loop is done you return the value of nextQ

**Project 7: Chain Code**

Student: Trisha Espejo

Project Due Date: 4/30/2021

**Hand Drawn Chain Code trace of Img1CC.txt:**

**Background pattern

Description automatically generated**

**Graphical user interface, text

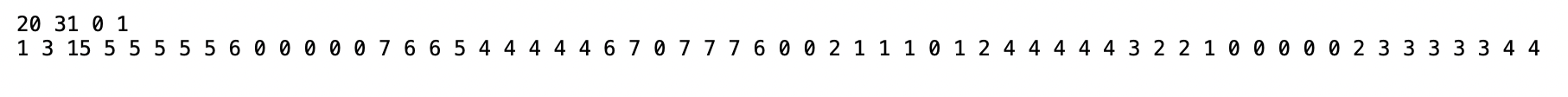
Description automatically generated with medium confidence**

**Hand Drawn Chain Code trace of Image 1 output:**

**Background pattern

Description automatically generated**

**Chain Code:**

****

**Source Code:**

**import** java.io.\*;

**import** java.util.\*;

**public** **class** main {

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

**try**(

Scanner imgFile = **new** Scanner(**new** BufferedReader( **new** FileReader( args[0])));

Scanner CCFile = **new** Scanner(**new** BufferedReader( **new** FileReader( args[1])));

BufferedWriter chainCodeFile = **new** BufferedWriter( **new** FileWriter("chainCode.txt"));

BufferedWriter boundaryFile = **new** BufferedWriter( **new** FileWriter("Boundary.txt"));

){

image img = **new** image();

CCproperty prop = **new** CCproperty();

chainCode chain = **new** chainCode();

img.loadImg(imgFile);

prop.readImageheader(CCFile);

**int** numCC = 0;

**if** (CCFile.hasNextInt()) numCC = CCFile.nextInt();

chain.numCC = numCC;

chain.constructor();

chain.CC = prop;

chain.printImgHeader(chainCodeFile);

**while**(prop.label <= chain.numCC){

prop.loadprop(CCFile);

prop.clearCC(img.CCAry);

prop.loadCCAry(img.imgAry, img.CCAry);

chain.CC = prop;

chain.getChainCode(img.CCAry, chainCodeFile);

**if** (prop.label == chain.numCC)

**break**;

}

System.***out***.println("finish getting chain code");

imgFile.close();

CCFile.close();

chainCodeFile.close();

Scanner chainCodeFile2 = **new** Scanner(**new** BufferedReader( **new** FileReader( "chainCode.txt")));

chain.constructBoundary (img.boundaryAry, chainCodeFile2);

System.***out***.println("finish creating boundary");

chain.reformatPrettyPrint(img.boundaryAry, boundaryFile);

System.***out***.println("finish printing boundary");

boundaryFile.close();

System.***out***.println("Compilation Complete");

}

}

}

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** image {

**public** **int** numRows = -1;

**public** **int** numCols = -1;

**public** **int** minValue = -1;

**public** **int** maxValue = -1;

**public** **int**[][] imgAry;

**public** **int**[][] boundaryAry;

**public** **int**[][] CCAry;

**public** **void** initAry() {

**this**.imgAry = **new** **int** [numRows + 2][numCols + 2];

**this**.boundaryAry = **new** **int** [numRows + 2][numCols + 2];

**this**.CCAry = **new** **int** [numRows + 2][numCols + 2];

}

**public** **void** setZero(**int**[][] Ary) {

**for** (**int** i = 0; i < numRows + 2; i++) {

**for**(**int** j = 0; j < numCols + 2; j++) {

Ary[i][j] = 0;

}

}

}

**public** **void** loadImg(Scanner file) **throws** IOException {

**if** (file.hasNextInt()) **this**.numRows = file.nextInt();

**if** (file.hasNextInt()) **this**.numCols = file.nextInt();

**if** (file.hasNextInt()) **this**.minValue = file.nextInt();

**if** (file.hasNextInt()) **this**.maxValue = file.nextInt();

initAry();

setZero(imgAry);

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

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

**if** (file.hasNextInt())imgAry[i][j] = file.nextInt();

}

}

}

}

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** chainCode {

**public** **int** lastQ = -1;

**public** **int** nextDir = -1;

**public** **int** PchainDir;

**public** **int** numCC = -1;

**public** **int**[] zeroTable;

**public** point[] neighborCoord;

**public** point startP;

**public** point currP;

**public** point nextP;

**public** CCproperty CC;

**public** **void** constructor() {

zeroTable = **new** **int**[8];

neighborCoord = **new** point[8];

startP = **new** point();

currP = **new** point();

nextP = **new** point();

initzeroTable();

}

**private** **void** initzeroTable() {

zeroTable[0] = 6;

zeroTable[1] = 0;

zeroTable[2] = 0;

zeroTable[3] = 2;

zeroTable[4] = 2;

zeroTable[5] = 4;

zeroTable[6] = 4;

zeroTable[7] = 6;

}

**public** **void** printImgHeader(BufferedWriter output) **throws** IOException {

output.write( (CC.numRows) + " " + Integer.*toString*(CC.numCols) + " " );

output.write( Integer.*toString*(CC.minValue) + " " + Integer.*toString*(CC.maxValue));

output.write("\n");

}

**public** **void** getChainCode(**int**[][] CCAry, BufferedWriter output) **throws** IOException {

**int** label = CC.label;

**int** nextQ = 0;

**boolean** isfound = **false**;

**for** (**int** i = 1; i < CC.numRows + 1 && !isfound; i++) {

**for** (**int** j = 1; j < CC.numCols + 1 && !isfound; j++) {

**if** (CCAry[i][j] == label) {

output.write(label + " " + i + " " + j + " ");

isfound = **true**;

startP.row = i;

startP.col = j;

currP.row = i;

currP.col = j;

lastQ = 4;

}

}

}

nextQ = (lastQ + 1) % 8;

PchainDir = findNextP(CCAry, currP, nextQ);

nextP = neighborCoord[PchainDir];

output.write(PchainDir + " ");

**if**( PchainDir == 0)

lastQ =zeroTable[7];

**else**

lastQ = zeroTable[PchainDir - 1];

currP = nextP;

**while** (**true**) {

nextQ = (lastQ + 1) % 8;

PchainDir = findNextP(CCAry, currP, nextQ);

nextP = neighborCoord[PchainDir];

output.write(PchainDir + " ");

**if**( PchainDir == 0)

lastQ =zeroTable[7];

**else**

lastQ = zeroTable[PchainDir - 1];

currP = nextP;

**if** (currP.row == startP.row && currP.col == startP.col) {

**break**;

}

}

output.write("\n");

}

**private** **int** findNextP(**int**[][] CCAry, point currP, **int** nextQ) {

**int** row = 0;

**int** col = 0;

point temp = **new** point();

loadNeighborCoord(currP);

temp = neighborCoord[nextQ];

row = temp.row;

col = temp.col;

**while**(CCAry[row][col] <= 0) {

temp = **new** point();

nextQ = (++nextQ) % 8;

temp = neighborCoord[nextQ];

row = temp.row;

col = temp.col;

}

**return** nextQ;

}

**private** **void** loadNeighborCoord( point currP) {

**int** row = currP.row;

**int** col = currP.col;

point temp = **new** point();

temp.row = row;

temp.col = col + 1;

neighborCoord[0] = temp;

temp = **new** point();

temp.row = row - 1;

temp.col = col + 1;

neighborCoord[1] = temp;

temp = **new** point();

temp.row = row - 1;

temp.col = col;

neighborCoord[2] = temp;

temp = **new** point();

temp.row = row - 1;

temp.col = col - 1;

neighborCoord[3] = temp;

temp = **new** point();

temp.row = row;

temp.col = col - 1;

neighborCoord[4] = temp;

temp = **new** point();

temp.row = row + 1;

temp.col = col - 1;

neighborCoord[5] = temp;

temp = **new** point();

temp.row = row + 1;

temp.col = col;

neighborCoord[6] = temp;

temp = **new** point();

temp.row = row + 1;

temp.col = col + 1;

neighborCoord[7] = temp;

}

**public** **void** constructBoundary(**int**[][] boundaryAry, Scanner file) {

**if** (file.hasNextInt()) CC.numRows = file.nextInt();

**if** (file.hasNextInt()) CC.numCols = file.nextInt();

**if** (file.hasNextInt()) CC.minValue = file.nextInt();

**if** (file.hasNextInt()) CC.maxValue = file.nextInt();

**int** direct = 0;

**int** r = 0;

**int** c = 0;

**if** (file.hasNextInt()) CC.label = file.nextInt();

**while**(CC.label <= numCC){

**if** (file.hasNextInt()) r = file.nextInt();

**if** (file.hasNextInt()) c = file.nextInt();

**if** (file.hasNextInt()) direct = file.nextInt();

startP.row = r;

startP.col = c;

currP = startP;

boundaryAry[currP.row][currP.col] = CC.label;

loadNeighborCoord(currP);

point temp = **new** point();

temp = neighborCoord[direct];

currP = temp;

**while** (**true**) {

**if** (file.hasNextInt()) direct = file.nextInt();

boundaryAry[currP.row][currP.col] = CC.label;

loadNeighborCoord(currP);

temp = **new** point();

temp = neighborCoord[direct];

currP = temp;

**if**(currP.row == startP.row && currP.col == startP.col)

**break**;

}

**if** (CC.label == numCC)

**break**;

**if** (file.hasNextInt()) CC.label = file.nextInt();

}

}

**public** **void** reformatPrettyPrint(**int**[][] Ary, BufferedWriter output) **throws** IOException {

output.write( (CC.numRows) + " " + Integer.*toString*(CC.numCols) + " " );

output.write( Integer.*toString*(CC.minValue) + " " + Integer.*toString*(CC.maxValue));

output.write("\n");

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

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

**if** (Ary[i][j] > 0) output.write( Integer.*toString*(Ary[i][j]) + " ");

**else** output.write(". ");

}

output.write("\n");

}

output.write("\n\n");

}

}

**import** java.io.BufferedWriter;

**import** java.util.Scanner;

**public** **class** CCproperty {

**public** **int** numRows = -1;

**public** **int** numCols = -1;

**public** **int** minValue = -1;

**public** **int** maxValue = -1;

**public** **int** label = 0;

**public** **int** numpixels = 0;

**public** **int** minRow = -1;

**public** **int** minCol = -1;

**public** **int** maxRow = -1;

**public** **int** maxCol = -1;

**public** **void** readImageheader(Scanner file){

**if** (file.hasNextInt()) **this**.numRows = file.nextInt();

**if** (file.hasNextInt()) **this**.numCols = file.nextInt();

**if** (file.hasNextInt()) **this**.minValue = file.nextInt();

**if** (file.hasNextInt()) **this**.maxValue = file.nextInt();

}

**public** **void** clearCC(**int**[][] Ary) {

**for** (**int** i = 0; i < numRows + 2; i++) {

**for**(**int** j = 0; j < numCols + 2; j++) {

Ary[i][j] = 0;

}

}

}

**public** **void** loadprop(Scanner file) {

**if** (file.hasNextInt()) **this**.label = file.nextInt();

**if** (file.hasNextInt()) **this**.numpixels = file.nextInt();

**if** (file.hasNextInt()) **this**.minRow = file.nextInt();

**if** (file.hasNextInt()) **this**.minCol = file.nextInt();

**if** (file.hasNextInt()) **this**.maxRow = file.nextInt();

**if** (file.hasNextInt()) **this**.maxCol = file.nextInt();

}

**public** **void** loadCCAry(**int**[][] imgAry, **int**[][] CCAry) {

**for**(**int** i = minRow + 1; i < maxRow + 2; i++) {

**for**(**int** j = minCol + 1; j < maxCol + 2; j++) {

CCAry[i][j] = imgAry[i][j];

}

}

}

}

**public** **class** point **extends** chainCode {

**public** **int** row = -1;

**public** **int** col = -1;

}

**Image 1:**

**Input :**

**Img1CC.txt:**

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

**Img1Property.txt:**

20 31 0 1

1

1

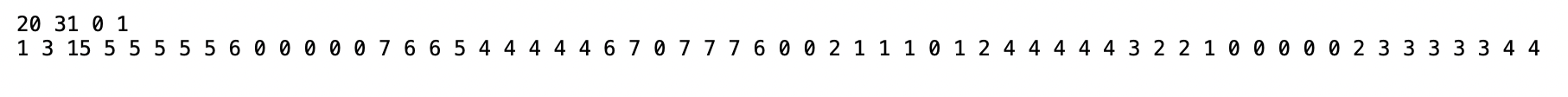
119

2 9

18 21

**Output:**

**Img1ChainCode.txt:**

****

**Img1Boundar.txt:**

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:**

**Input :**

**Img2CC.txt:**

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

**Img2Property.txt:**

20 40 0 3

3

1

172

2 4

19 20

2

73

2 25

10 35

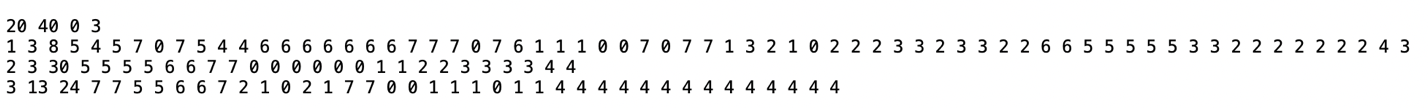
3

68

12 23

19 37

**Img2ChainCode.txt:**

****

**Img2Boundar.txt:**

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