Computer Vision

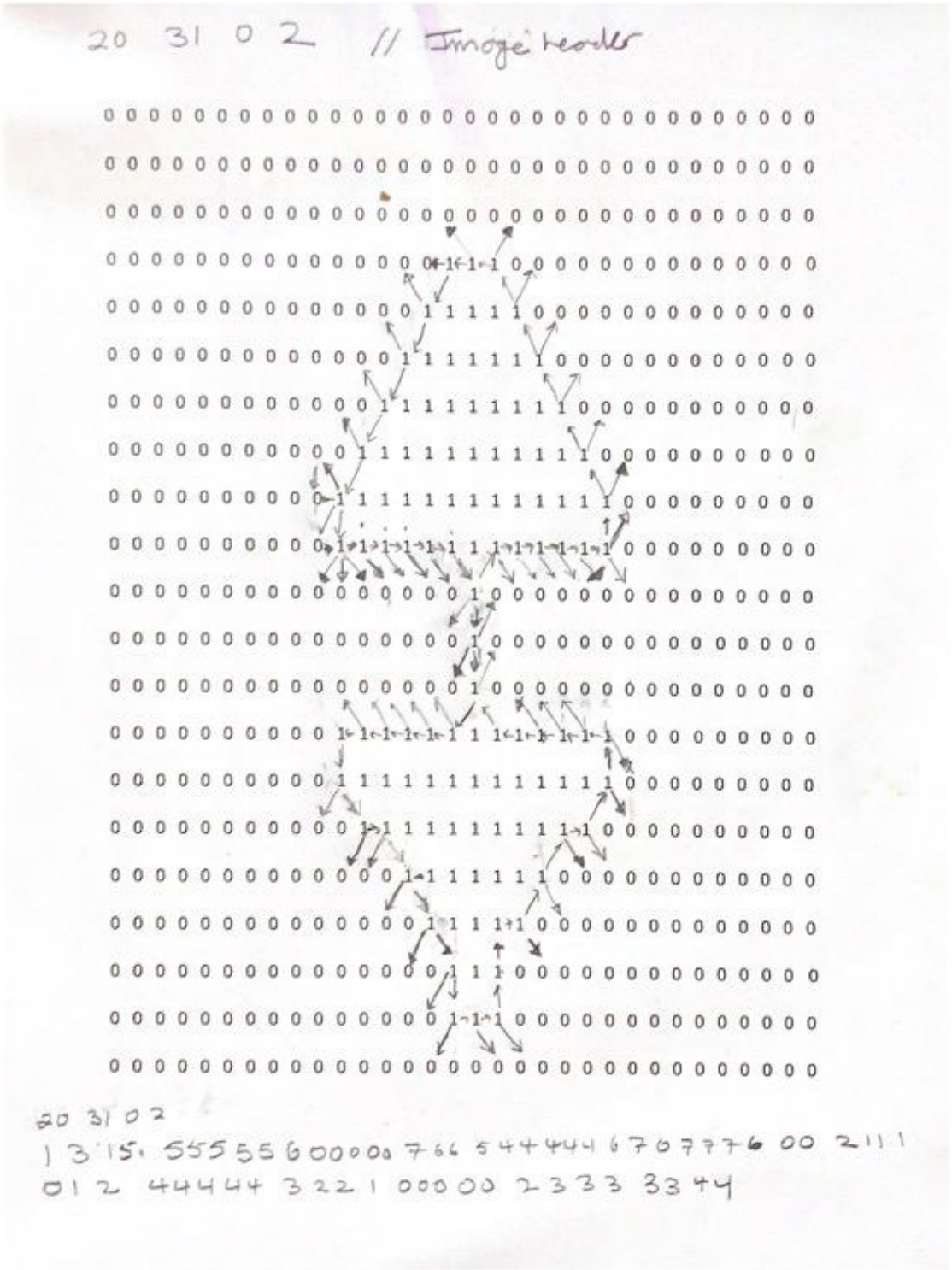
Project 7: Chain Code

Student: Andrew Alleyne

Algorithm Steps for Chain Code

1. Given binary image.
2. Scan image left to right and top bottom until you encounter a nonzero pixel
3. Store the current row and column along with the current pixels' grayscale.
4. Record the last encountered zero position.
5. Store the current direction by using the last encountered zero incremented by 1 and find the remainder of its results by 8. Doing this gives us the direction to potentially find the starting pixel
6. Find the next pixel by scanning counterclockwise until the current pixel is greater than zero.
7. If pixel is found return, it's direction and store it in the next pixels value.
8. Swap the current pixel with the returned next pixel value.
9. Last zero position is recorded by observing the position before the current direction.

Scanned hand traced chain code.



Source Code

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Main {

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

        //check arguments
        if (args.length < 2) {

            System.out.println("You need more arguments!");
        }

        int numRows = 0;
        int numCols = 0;
        int minVal = 0;
        int maxVal = 0;

        int nextBorder = 0;

        //To reconstruct image
        int startingRow = 0;
        int startingCol = 0;
        int pixelValue = 0;

        int[][] imageAry; // needs to dynamically allocate at run time (numRows+2 by numCols+2)
        int[][] CC_Ary; // needs to dynamically allocate at run time (numRows+2 by numCols+2)
        int[][] boundary_Ary; // needs to dynamically allocate at run time (numRows+2 by numCols+2)

        //Provided input files for processing
        String labelFile = args[0];
        String propFile = args[1];

        //Files generated at runtime
        String chainCode = args[0].replace(".txt", "") + "_ChainCodeFile.txt";
        String boundaryFile = args[0].replace(".txt", "") + "_BoundaryFile.txt";


        //Open stream to input files
        File inputFile = new File(labelFile);
        File inputFile_2 = new File(propFile);
```

```
Scanner input = new Scanner(inputFile);
Scanner input_2 = new Scanner(inputFile_2);


//Create new files to output to
File outputFile = new File(chainCode);
outputFile.createNewFile();

File outputFile_2 = new File(boundaryFile);
outputFile_2.createNewFile();

//Initialize r,c,mv,Mv
if (input.hasNextInt()) numRows = input.nextInt();
if (input.hasNextInt()) numCols = input.nextInt();
if (input.hasNextInt()) minVal = input.nextInt();
if (input.hasNextInt()) maxVal = input.nextInt();

//Dynamically allocate arrays

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

//Zero-frame arrays from imageAry
for (int i = 0; i < numRows + 2; i++) {
   for (int j = 0; j < numCols + 2; j++) {
      imageAry[i][j] = 0;
   }
}

//Load image from input image
for (int i = 1; i <= numRows; i++) {
   for (int j = 1; j <= numCols; j++) {
      if (input.hasNextInt()) imageAry[i][j] = input.nextInt();
   }
}

//Write Headers to files.
FileWriter fileWriter = new FileWriter(chainCode);

//Write header to Chain code file
fileWriter.write(numRows + " " + numCols + " " + minVal + " " + maxVal + "\n");

FileWriter fileWriter_2 = new FileWriter(boundaryFile);
```

```
//Write header to Boundary file
fileWriter_2.write(numRows + " " +
       numCols + " " +
       minVal + " " +
       maxVal + "\n");

//CC_property class
CC cc = new CC(input_2);

//Zero out CC_Ary
cc.clearCC_Ary(CC_Ary);

//Extract CC.label from the image array into CC_Ary
cc.loadCC(cc.label, CC_Ary, imageAry);

//Get chain code
cc.getChainCode(fileWriter, CC_Ary);

//Construct Boundary

//Open stream to labelFile_ChainCodeFile.txt
Scanner input_3 = new Scanner(outputFile);

//Read next items from stream
if (input_3.hasNextInt()) numRows = input_3.nextInt();
if (input_3.hasNextInt()) numCols = input_3.nextInt();


if (input_3.hasNextInt()) minVal = input_3.nextInt();
if (input_3.hasNextInt()) maxVal = input_3.nextInt();


nextBorder++;

while (input_3.hasNextLine()) {

   if (input_3.hasNextInt()) pixelValue = input_3.nextInt();


   //get starting row and starting column
   if (input_3.hasNextInt()) startingRow = input_3.nextInt();
   if (input_3.hasNextInt()) startingCol = input_3.nextInt();


   //We can use the header to recreate the original image
   boundary_Ary[startingRow][startingCol] = pixelValue;
```

```
int rOffset = startingRow;
int cOffset = startingCol;


String line = input_3.nextLine();


nextBorder++;

Scanner lineScan = new Scanner(line);

while (lineScan.hasNextInt()) {

   int direction = lineScan.nextInt();

   switch (direction) {

      case 0:
         cOffset++;
         boundary_Ary[rOffset][cOffset] = pixelValue;
         break;

      case 1:
         rOffset--;
         cOffset++;
         boundary_Ary[rOffset][cOffset] = pixelValue;
         break;

      case 2:
         rOffset--;
         boundary_Ary[rOffset][cOffset] = pixelValue;
         break;

      case 3:
         rOffset--;
         cOffset--;
         boundary_Ary[rOffset][cOffset] = pixelValue;
         break;

      case 4:
         cOffset--;
         boundary_Ary[rOffset][cOffset] = pixelValue;
         break;

      case 5:
```

```
                rOffset++;
                cOffset--;
                boundary_Ary[rOffset][cOffset] = pixelValue;
                break;

            case 6:
                rOffset++;
                boundary_Ary[rOffset][cOffset] = pixelValue;
                break;

            case 7:
                rOffset++;
                cOffset++;
                    boundary_Ary[rOffset][cOffset] = pixelValue;
                    break;

                default:
                    break;

            }

        }

    }


    for (int i = 0; i <= numRows; i++) {
        for (int j = 0; j <= numCols; j++) {
            fileWriter_2.write(boundary_Ary[i][j] + " ");
        }
        fileWriter_2.write("\n");
    }


    //Close all outputs
    input.close();
    input_2.close();
    input_3.close();
    fileWriter.close();
    fileWriter_2.close();
  }
}
```

CC.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;



public class CC {

    //Point class
    class Point {

        int row;
        int col;

        public Point() {

        }
    }

    Point startP = new Point();

    Scanner inputFile;

    public int label;
    public int numPixels;
    public int minRow;
    public int minCol;
    public int maxRow;
    public int maxCol;

    int numRows;
    int numCols;
    int minVal;
    int maxVal;
    Point currentP = new Point();
    Point nextP = new Point();
    Point[] neighborCoord = new Point[8];
    int zeroTable[] = {6, 0, 0, 2, 2, 4, 4, 6};
```

```
int PchainDir;
int nextDir;
int lastQ;
int nextQ;
int numConnectedComponents;

int nextChain = 0;


CC(Scanner inputFile) {
   this.inputFile = inputFile;

   //Reads the header information
   if (inputFile.hasNextInt()) numRows = inputFile.nextInt();
   if (inputFile.hasNextInt()) numCols = inputFile.nextInt();
   if (inputFile.hasNextInt()) minVal = inputFile.nextInt();
   if (inputFile.hasNextInt()) maxVal = inputFile.nextInt();

   //Reads the number of connected components
   if (inputFile.hasNextInt()) numConnectedComponents = inputFile.nextInt();

   //CC label
   if (inputFile.hasNextInt()) label = inputFile.nextInt();

   //numPixels
   if (inputFile.hasNextInt()) numPixels = inputFile.nextInt();


   //Reads the minRow, minCol of greyscale pixel values
   if (inputFile.hasNextInt()) minRow = inputFile.nextInt();
   if (inputFile.hasNextInt()) minCol = inputFile.nextInt();

   //Reads the maxRow, maxCol of greyscale pixel values
   if (inputFile.hasNextInt()) maxRow = inputFile.nextInt();
   if (inputFile.hasNextInt()) maxCol = inputFile.nextInt();

}

public void getChainCode(FileWriter fileWriter, int[][] cc_ary) throws IOException {

   nextChain++;

   while (nextChain <= numConnectedComponents) {
      boolean isThere = false;
```

```
for (int i = 0; i < numRows + 2; i++) {
   for (int j = 0; j < numCols + 2; j++) {

      //get the first pixel
      if (cc_ary[i][j] == label) {
         fileWriter.write(label + " " + i + " " + j + " ");
         startP.row = i;
         startP.col = j;
         currentP.row = i;
         currentP.col = j;
         lastQ = 4;

         while (currentP != startP) {
            nextQ = (lastQ + 1) % 8;

            PchainDir = findNextP(currentP, nextQ, cc_ary);
            nextP.row = neighborCoord[PchainDir].row;
            nextP.col = neighborCoord[PchainDir].col;

            fileWriter.write(PchainDir + " ");

            if (PchainDir == 0) {
               lastQ = zeroTable[7];
            } else {
               lastQ = zeroTable[PchainDir - 1];
            }

            //change currentP to nextP
            currentP.row = nextP.row;
            currentP.col = nextP.col;

            //check if currentP == nextP, if so break
            if (currentP.row == startP.row && currentP.col == startP.col) {
               isThere = true;
               break;
            }
         }
         fileWriter.flush();
      }

      if (isThere) {
         break;
      }
   }
```

```
            //check condition before moving to the next row
            if (isThere) {
                break;
            }
        }

        //Next chain
        nextChain++;


        //CC label
        if (inputFile.hasNextInt()) label = inputFile.nextInt();

        //numPixels
        if (inputFile.hasNextInt()) numPixels = inputFile.nextInt();


        //Reads the minRow, minCol of greyscale pixel values
        if (inputFile.hasNextInt()) minRow = inputFile.nextInt();
        if (inputFile.hasNextInt()) minCol = inputFile.nextInt();

        //Reads the maxRow, maxCol of greyscale pixel values
        if (inputFile.hasNextInt()) maxRow = inputFile.nextInt();
        if (inputFile.hasNextInt()) maxCol = inputFile.nextInt();

        fileWriter.write("\n");
    }
}

//Zeroes out the CC_Ary
public void clearCC_Ary(int[][] cc_ary) {
    for (int i = 0; i < numRows + 2; i++) {
        for (int j = 0; j < numCols + 2; j++) {
            cc_ary[i][j] = 0;
        }
    }
}

//Extracts image where label is > 0
public void loadCC(int label, int[][] cc_ary, int[][] imageAry) {
    for (int i = 1; i <= numRows; i++) {
        for (int j = 1; j <= numCols; j++) {
            if (imageAry[i][j] > 0) {
                cc_ary[i][j] = imageAry[i][j];
                //System.out.print(cc_ary[i][j] + " ");
            } else {
```

```
                //  System.out.print("0 ");
               }
            }
            //System.out.println();
         }
      }

      //We must know the current P and then obtain the next

   public int findNextP(Point currentP, int nextQ, int[][] cc_ary) {
         int chainDir;
         int direction = nextQ;


         //we must first load the neighbor coordinates.
         loadNeighborCoord(currentP);

         while(true){
            if(cc_ary[neighborCoord[direction].row][neighborCoord[direction].col] == label){
               chainDir = direction;
               break;
            }
            direction = (direction+1)%8;
         }

         //next chain direction
         return chainDir;
      }


      private void loadNeighborCoord(Point currentP) {

         //array of objects to store x-y
         for (int i = 0; i < 8; i++) {
            neighborCoord[i] = new Point();
         }

         neighborCoord[0].row = currentP.row;
         neighborCoord[0].col = currentP.col + 1;
         neighborCoord[1].row = currentP.row - 1;
         neighborCoord[1].col = currentP.col + 1;
         neighborCoord[2].row = currentP.row - 1;
         neighborCoord[2].col = currentP.col;
         neighborCoord[3].row = currentP.row - 1;
         neighborCoord[3].col = currentP.col - 1;
         neighborCoord[4].row = currentP.row;
```

```
        neighborCoord[4].col = currentP.col - 1;
        neighborCoord[5].row = currentP.row + 1;
        neighborCoord[5].col = currentP.col - 1;
        neighborCoord[6].row = currentP.row + 1;
        neighborCoord[6].col = currentP.col;
        neighborCoord[7].row = currentP.row + 1;
        neighborCoord[7].col = currentP.col + 1;

    }



}
```

Program Output – img1CC

Original Image - img1CC

```
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 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 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 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 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 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 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 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 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 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 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 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 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 0 0 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 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 0 0 0 0
```

Connect Component Property File

```
20 31 0 1
1
1
119
2 9
18 21
```

Chain code – img1CC

```
20 31 0 1
1 3 15 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
```

Boundary File- img1CC

```
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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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 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 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 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 0 0 1 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 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 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 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 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 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 0 1 1 0 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 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 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 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 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 0 0 0 0
```

Program Output 2 – img2CC

Original Image – img2CC

```
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 1 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 2 2 2 2 2 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 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 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 0 0 2 2 2 2 2 2 2 2 2 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 0 0 2 2 2 2 2 2 2 2 2 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 0 0
0 0 0 0 1 1 1 1 1 1 0 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 1 1 1 1 1 1 1 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 1 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 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 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 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 3 3 3 3 0 3 3 3 3 3 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 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 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
```

Connect Component Property File – img2C

```
20 40 0 3
3
1
172
2 4
19 20
2
73
2 25
10 35
3
68
12 23
19 37
```

Chain Code – img2CC

```
20 40 0 3
1 3 8 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 3 30 5 5 5 5 6 6 7 7 0 0 0 0 0 0 0 1 1 2 2 3 3 3 3 4 4
3 13 24 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 4
```

Boundary File – img2CC

```
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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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 0 2 0 0 0 2 0 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 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 0 2 0 0 0 0 0 0 0 2 0 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 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 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 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 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 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 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 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 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 1 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 3 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 1 0 0 0 3 0 0 0 0 0 0 0 0 0 3 3 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 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 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 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 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 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 0
```