CV **Image Compression Via Distance Transform** Java

Student: Shuhua Song

Project Due Date: 03/30/2021

(1) Algorithm Steps for Distance Transform Pass1 (inArray, outAry):
step 1: image ← given binary image
step 2: scan image left->right & top->bottom
step 3: p(i, j) ← next pixel
if p(i, j) > 0
look at neighbors: a, b, c, d
8-Connected Distance-> p(i,j) ← min(a, b, c, d) + 1
step 4: repeat step 2 to step 3 until all pixels are processed

(2) Algorithm Steps for Distance Transform Pass2 (inArray, outAry):
step 1: image given the result of Pass-1
step 2: scan image right-> left & bottom-> top
step 3: p(i, j) ← next pixel
if p(i, j) > 0
look at neighbors: a, b, c, d
8-Connected Distance-> p(i,j) ← min(e+1, f+1, g+1, h+1, p(I, j))
step 4: repeat step 2 to step 3 until all pixels are processed

(3) Algorithm steps for Local Maxima Operation(inArray, outArray)
step 1: image ← given the result of Pass-2
skeleton ← array of same size of image
step 2: Scan image left->right & top->bottom
p(i, j) ← next pixel
step 3: check if p(i, j) is a local Maxima
iff p(i, j) >= a, b, c, d, e, f, g, h
skeleton(i, j) ← p(i,j)
else
skeleton(i,j) ← 0
step 4: repeat steps 2 to 3 until all pixels are processed

(4) Algorithm steps for Skeleton Image Compression (inArray, outArray)
Step 1: image ← given the result of local Maxima Operation
Output ← file for result of compression
Step 2: Scan image left->right & top ->bottom
p(i, j) ← next pixel
Step 3: if p(i, j) > 0
output ← i, j, p(i, j)
Step 4: repeat steps 2 to 3 until all pixels are processed

(5) Algorithm steps for Expansion Pass 1(inArray, outArray)
Step 1: image ← load the file rendered by skeleton compression
Step 2: scan image left -> right & top -> bottom
p(i, j) ← next pixel
Step 3: if p(i, j) == 0
Look at neighbors of p(i, j) : a, b, c, d, e, f, g, h
8-Connected Distance -> max ← max(a-1, b-1, c-1, d-1, e-1, f-1, g-1, h-1)
Step 4: repeat steps 2 to 3 until all pixels are processed

(6) Algorithm Steps for Expansion Pass2 (inArray, outAry):
step 1: image ← given the result of Expansion Pass-1
step 2: scan image right-> left & bottom-> top
p(i, j) ← next pixel
step 3: Look at neighbors of p(i, j) : a, b, c, d, e, f, g, h
8-Connected Distance -> max ← max(a, b, c, d, e, f, g, h)
If p(i, j) < max
p(i,j) ← max - 1
Step 4: repeat steps 2 to 3 until all pixels are processed

Source Code:

```
import java.io.*;
import java.util.PriorityQueue;
import java.util.Scanner;

public class ImageProcessing {
    int numRows, numCols;
    int minVal, maxVal;
    int newMinVal, newMaxVal;

    public  int[][] zeroFramedAry;
    public  int[][] skeletonAry;

    public ImageProcessing(){}
    public ImageProcessing(int numRows, int numCols, int minVal, int maxVal, int
newMinVal, int newMaxVal){
        this.numRows = numRows;
        this.numCols = numCols;
        this.minVal = minVal;
        this.maxVal = maxVal;
        this.newMinVal = newMinVal;
        this.newMaxVal = newMaxVal;

        zeroFramedAry = new int[numRows+2][numCols+2];
        skeletonAry = new int[numRows+2][numCols+2];

    }
    public void setZero(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[][] zeroFrameAry){
        for(int i=1; i<numRows+1; i++){
            for(int j=1; j<numCols+1; j++){
                zeroFrameAry[i][j] = inFile.nextInt();
            }
        }
    }

    public void firstPass_8Distance(int[][] inArray, BufferedWriter outFile) throws
IOException {
        outFile.write("\nFirst Pass for 8-Connected Distance \n\n");
         for(int i=1; i<inArray.length-1; i++){
             for(int j=1; j<inArray[0].length-1; j++){
                 if(inArray[i][j] > 0){
                     int val = getCellVal_pass1(inArray, i, j);
                     inArray[i][j] = val + 1;
                     newMinVal = Math.min(newMinVal, inArray[i][j]);
                     newMaxVal = Math.max(newMaxVal, inArray[i][j]);
                 }
             }
         }
        System.out.println("firstPass_8Distance: " + newMinVal + " " + newMaxVal);
    }

    public int getCellVal_pass1(int[][] inArray, int i, int j){
        int a = inArray[i-1][j-1];
        int b = inArray[i-1][j];
        int c = inArray[i-1][j+1];
        int d = inArray[i][j-1];
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(a); pq.add(b);
        pq.add(c); pq.add(d);
        return pq.peek();
    }

    public void secondPass_8Distance(int[][] inArray, BufferedWriter outFile) throws
IOException {
        newMaxVal = newMinVal;
        outFile.write("\n\nSeconde Pass for 8-Connected Distance\n");
         for(int i=inArray.length-2; i>=1; i--){
             for(int j=inArray[0].length-2; j>=1; j--){
                 int val = getCellVal_pass2(inArray, i, j);
                 inArray[i][j] = val;
                 newMinVal = Math.min(newMinVal, inArray[i][j]);
                 newMaxVal = Math.max(newMaxVal, inArray[i][j]);
             }
         }
        System.out.println("secondPass_8Distance: " + newMinVal + " " + newMaxVal);
    }

    public int getCellVal_pass2(int[][] inArray, int i, int j){
        int e = inArray[i][j+1]+1;
        int f = inArray[i+1][j-1]+1;
        int g = inArray[i+1][j]+1;
        int h = inArray[i+1][j+1]+1;
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(e); pq.add(f);
```

```
        pq.add(g); pq.add(h);
        pq.add(inArray[i][j]);
        return pq.peek();
    }

    public void compute8Distance(int[][] zeroFramedAry, BufferedWriter outFile) throws
IOException {
        firstPass_8Distance(zeroFramedAry, outFile);
        reformatPrettyPrint(zeroFramedAry, outFile);
        secondPass_8Distance(zeroFramedAry, outFile);
        reformatPrettyPrint(zeroFramedAry, outFile);
    }


   public void skeletonExtraction(int[][] zeroFramedAry, int[][] skeletonAry,
BufferedWriter skeletonFile, BufferedWriter outFile) throws IOException {
        outFile.write("\n Skeleton Extraction\n");
        computeLocalMaxima(zeroFramedAry, skeletonAry);
        reformatPrettyPrint(skeletonAry, outFile);
        extractLocalMaxima(skeletonAry, skeletonFile);
        skeletonFile.close();
   }

   public void computeLocalMaxima(int[][] zeroFramedAry, int[][] skeletonAry){
       newMaxVal = newMinVal;
          int rows = zeroFramedAry.length;
          int cols = zeroFramedAry[0].length;
          for(int i=1; i<rows-1; i++){
              for(int j=1; j<cols-1; j++){
                  int localMax = getLocalMax(zeroFramedAry, i, j);
                  //System.out.println();
                  if(zeroFramedAry[i][j] >= localMax){
                      skeletonAry[i][j] = zeroFramedAry[i][j];
                  }else{
                      skeletonAry[i][j] = 0;
                  }
                  newMinVal = Math.min(newMinVal, skeletonAry[i][j]);
                  newMaxVal = Math.max(newMaxVal, skeletonAry[i][j]);
              }
          }
       System.out.println("compute Local Maxima: " + newMinVal + " " + newMaxVal);
   }
   public int getLocalMax(int[][] zeroFramedAry, int i, int j){
        PriorityQueue<Integer> pq = new PriorityQueue<>((a, b)->b-a);
        pq.add(zeroFramedAry[i-1][j-1]);
        pq.add(zeroFramedAry[i-1][j]);
        pq.add(zeroFramedAry[i-1][j+1]);
        pq.add(zeroFramedAry[i][j-1]);
        pq.add(zeroFramedAry[i][j+1]);
        pq.add(zeroFramedAry[i+1][j-1]);
        pq.add(zeroFramedAry[i+1][j]);
        pq.add(zeroFramedAry[i+1][j+1]);
        return pq.peek();
   }

   public void extractLocalMaxima(int[][] skeletonAry, BufferedWriter skeletonFile)
throws IOException {
         for(int i=0; i<skeletonAry.length; i++){
             for(int j=0; j<skeletonAry[0].length; j++){
                 if(skeletonAry[i][j] > 0){
                     skeletonFile.write(i + "  " + j + "  " + skeletonAry[i][j] +
```

```
"\n");
                }
            }
        }
    }
    public void skeletonExpansion(int[][] zeroFramedAry,  BufferedWriter outFile,
String skeletonFileName) throws IOException {
        Scanner skeletonFile = new Scanner(new FileReader(skeletonFileName));
        setZero(zeroFramedAry);
        loadSkeleton(skeletonFile, zeroFramedAry);
        firstPass_Expension(zeroFramedAry, outFile);
        reformatPrettyPrint(zeroFramedAry, outFile);
        secondPass_Expension(zeroFramedAry, outFile);
        reformatPrettyPrint(zeroFramedAry, outFile);
    }

    public void loadSkeleton(Scanner skeletonFile, int[][] zeroFramedAry){
         int i, j, value;
         while(skeletonFile.hasNextInt()){
             i = skeletonFile.nextInt();
             j = skeletonFile.nextInt();
             value = skeletonFile.nextInt();
             //System.out.println(i + " " + j + " " + value);
             zeroFramedAry[i][j] = value;
         }
    }

    public void firstPass_Expension(int[][] zeroFramedAry,  BufferedWriter outFile)
throws IOException {
          outFile.write("\n" + "First Pass Expansion for 8-Connectness"  + "\n");
          int maxValue = 0;
          for(int i=1; i<zeroFramedAry.length-1; i++){
              for(int j=1; j<zeroFramedAry[0].length-1; j++){
                  if(zeroFramedAry[i][j]==0){
                      maxValue  = getLocalMax_firstExpansion(zeroFramedAry, i, j);
                      if(zeroFramedAry[i][j] < maxValue){
                          zeroFramedAry[i][j] = maxValue;
                          newMinVal = Math.min(newMinVal, zeroFramedAry[i][j]);
                          newMaxVal = Math.max(newMaxVal, zeroFramedAry[i][j]);
                      }
                  }
              }
          }
        System.out.println("firstPass_Expension: " + newMinVal + " " + newMaxVal);
    }

    public int getLocalMax_firstExpansion(int[][] zeroFramedAry, int i, int j){
        PriorityQueue<Integer> pq = new PriorityQueue<>((a, b)->b-a);
        pq.add(zeroFramedAry[i-1][j-1]-1);
        pq.add(zeroFramedAry[i-1][j]-1);
        pq.add(zeroFramedAry[i-1][j+1]-1);
        pq.add(zeroFramedAry[i][j-1]-1);
        pq.add(zeroFramedAry[i][j+1]-1);
        pq.add(zeroFramedAry[i+1][j-1]-1);
        pq.add(zeroFramedAry[i+1][j]-1);
        pq.add(zeroFramedAry[i+1][j+1]-1);
        return pq.peek();
    }

    public void secondPass_Expension(int[][] zeroFramedAry, BufferedWriter outFile)
throws IOException {
```

```
        newMaxVal = newMinVal;
         outFile.write("\n" + "Second Pass Expansion for 8-Connectness"  + "\n");
          int maxValue = 0;
         for(int i=zeroFramedAry.length-2; i>=1; i--){
              for(int j=zeroFramedAry[0].length-2; j>=1; j--){
                  maxValue = getLocalMax_secondExpansion(zeroFramedAry, i, j);
                  if(zeroFramedAry[i][j] < maxValue){
                      zeroFramedAry[i][j] = maxValue - 1;
                      newMinVal = Math.min(newMinVal, zeroFramedAry[i][j]);
                      newMaxVal = Math.max(newMaxVal, zeroFramedAry[i][j]);
                  }
              }
          }
        System.out.println("secondPass_Expension: " + newMinVal + " " + newMaxVal);
    }

    public int getLocalMax_secondExpansion(int[][] zeroFramedAry, int i, int j){
        PriorityQueue<Integer> pq = new PriorityQueue<>((a, b)->b-a);
        pq.add(zeroFramedAry[i-1][j-1]);
        pq.add(zeroFramedAry[i-1][j]);
        pq.add(zeroFramedAry[i-1][j+1]);
        pq.add(zeroFramedAry[i][j-1]);
        pq.add(zeroFramedAry[i][j+1]);
        pq.add(zeroFramedAry[i+1][j-1]);
        pq.add(zeroFramedAry[i+1][j]);
        pq.add(zeroFramedAry[i+1][j+1]);
        return pq.peek();
    }

    public void decompressFile(int[][] zeroFramedAry, BufferedWriter outFile) throws
IOException {
         outFile.write("\n Decompressed File \n");
         outFile.write(numRows + "   " + numCols + "   " + minVal + "   " + maxVal +
"\n");
         for(int i=1; i<zeroFramedAry.length; i++){
             for(int j=1; j<zeroFramedAry[0].length; j++){
                 if(zeroFramedAry[i][j] >= 1){
                     outFile.write(1 + " ");
                 }else{
                     outFile.write(0 + " ");
                 }
             }
             outFile.write("\n");
         }
    }
   //Pretty Print
    public void reformatPrettyPrint(int[][] inArray, BufferedWriter outFile) throws
IOException {
         //System.out.println("PrettyPrint Test");
        outFile.write(numRows + " " + numCols + " " + newMinVal + " " + newMaxVal +
"\n");
         for(int i=1; i<inArray.length-1; i++){
             for(int j=1; j<inArray[0].length-1; j++){
                 if(inArray[i][j]==0) {
                     outFile.write("." + " ");
                 }else{
                     outFile.write(inArray[i][j] + " ");
                 }
             }
             outFile.write("\n");
         }
```

```
    }

    public static void main(String[] args) throws IOException {
        Scanner inFile = new Scanner(new FileReader(args[0]));
        String infileName = args[0];
        BufferedWriter outFile1 = new BufferedWriter(new FileWriter(new
File(args[1])));
        BufferedWriter outFile2 = new BufferedWriter(new FileWriter(new
File(args[2])));
        int numRows, numCols, minVal, maxVal, newMinVal, newMaxVal;
        numRows = inFile.nextInt();
        numCols = inFile.nextInt();
        minVal  = inFile.nextInt();
        maxVal  = inFile.nextInt();
        newMinVal = minVal;
        newMaxVal = maxVal;
        //Define Custom File Name
        System.out.println(numRows + " " + numCols + " " + minVal + " " + maxVal);
        String skeletonFileName = infileName.substring(0,infileName.length()-4) +
"_skeleton.txt";
        String decompressFileName = infileName.substring(0,infileName.length()-4) +
"_decompressed.txt";
        BufferedWriter skeletonFile = new BufferedWriter(new FileWriter(new
File(skeletonFileName)));
        BufferedWriter decompressFile = new BufferedWriter(new FileWriter(new
File(decompressFileName)));

        ImageProcessing imageProcess = new ImageProcessing(numRows, numCols, minVal,
maxVal, newMinVal, newMaxVal);
        int[][] zeroFramedAry = imageProcess.zeroFramedAry;
        int[][] skeletonAry = imageProcess.skeletonAry;
        imageProcess.setZero(zeroFramedAry);
        imageProcess.setZero(skeletonAry);
        //Load Original Image
        imageProcess.loadImage(inFile, zeroFramedAry);
        outFile1.write("\n" + "Print of input File \n");
        imageProcess.reformatPrettyPrint(zeroFramedAry, outFile1);
        //Compute 8 Distance Transfer
        outFile1.write(infileName + "  Printing \n\n");
        imageProcess.compute8Distance(zeroFramedAry, outFile1);
        //Extract Skeleton
        imageProcess.skeletonExtraction(zeroFramedAry, skeletonAry, skeletonFile,
outFile1);
        //Expand Skeleton
        imageProcess.skeletonExpansion(zeroFramedAry,  outFile2, skeletonFileName);
        //Decompress File
        imageProcess.decompressFile(zeroFramedAry, decompressFile);
        inFile.close();
        outFile1.close();
        outFile2.close();
        //skeletonFile.close();
        decompressFile.close();
    }
}
```

- Print the input file

```
Print of input File
30 40 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 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 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 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 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 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 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 . . . . . . . .
. . . 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 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 . . . . . .
. . . 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 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 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 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 . . . . . . . . . . . . . . . . . . . . . . . . .
```

- Print outFile1

```
First Pass for 8-Connected Distance

30 40 0 10
. . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . 1 2 2 2 2 2 2 2 2 1 . . . . . . . .
. . . . . . . . . 1 1 1 . . . . . . . . . . 1 2 3 3 3 3 3 3 2 1 1 . . . . . . .
. . . . . . . . 1 1 2 1 1 . . . . . . . . . 1 2 3 4 4 4 4 3 2 2 1 1 . . . . . .
. . . . . . . 1 1 2 2 2 1 1 . . . . . . . . 1 2 3 4 5 5 4 3 3 2 2 1 1 . . . . .
. . . . . . 1 1 2 2 3 2 2 1 1 . . . . . . . 1 2 3 4 5 5 4 4 3 3 2 2 1 1 . . . .
. . . . . . 1 2 2 3 3 3 2 2 1 . . . . . . . 1 2 3 4 5 5 5 4 4 3 3 2 2 1 1 . . .
. . . . . . 1 2 3 3 4 3 3 2 1 . . . . . . . 1 2 3 4 5 6 5 5 4 4 3 3 2 2 1 1 . .
. . . . . . 1 2 3 4 4 4 3 2 1 . . . . . . . 1 2 3 4 5 6 6 5 5 4 4 3 3 2 2 1 1 .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . 1 1 2 3 4 5 6 6 6 5 5 4 4 3 3 2 2 1 1
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 6 7 6 6 5 5 4 4 3 3 2 2 .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 6 7 7 6 6 5 5 4 4 3 3 . .
. . . 1 1 1 1 2 3 4 5 4 3 2 1 1 1 1 1 . . . 1 2 3 4 5 6 7 7 7 6 6 5 5 4 4 . . .
. . . 1 2 2 2 2 3 4 5 4 3 2 2 2 2 2 1 . . . 1 2 3 4 5 6 7 8 7 7 6 6 5 5 . . . .
. . . 1 2 3 3 3 3 4 5 4 3 3 3 3 3 2 1 . . . 1 2 3 4 5 6 7 8 8 7 7 6 . . . . . .
. . . 1 2 3 4 4 4 4 5 4 4 4 4 4 3 2 1 . . . 1 2 3 4 5 6 7 8 8 8 7 . . . . . . .
. . . 1 2 3 4 5 5 5 5 5 5 5 5 4 3 2 1 . . . 1 2 3 4 5 6 7 8 9 8 . . . . . . . .
. . . 1 2 3 4 5 6 6 6 6 6 6 5 4 3 2 1 . . . 1 2 3 4 5 6 7 8 9 . . . . . . . . .
. . . 1 2 3 4 5 6 7 7 7 7 6 5 4 3 2 1 . . . 1 2 3 4 5 6 7 8 1 1 . . . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1 1 1 1 1 2 3 4 5 6 7 2 2 1 1 . . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 2 2 2 2 2 2 3 4 5 6 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 5 4 3 3 3 3 3 3 3 3 3 4 5 4 4 3 3 2 2 1 1 . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 5 4 4 4 4 4 4 4 4 4 4 4 5 5 4 4 3 3 2 2 . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 3 3 . . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5 4 4 . . . . . . . .
. . . 1 2 3 4 5 6 7 8 8 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6 6 5 5 . . . . . . . . .
. . . . 1 2 3 4 5 6 7 8 8 8 8 8 8 8 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 2 3 4 5 6 7 8 9 9 9 9 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . 1 2 3 4 5 6 7 8 9 10 . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . 1 2 3 4 5 6 7 8 . . . . . . . . . . . . . . . . . . . . . . . . .
```

Seconde Pass for 8-Connected Distance

```
30 40 0 7
. . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . 1 2 2 2 2 2 2 2 1 . . . . . . . .
. . . . . . . . . . 1 1 1 . . . . . . . . . 1 2 3 3 3 3 3 3 2 1 1 . . . . . .
. . . . . . . . . 1 1 2 1 1 . . . . . . . . 1 2 3 4 4 4 4 3 2 2 1 1 . . . . . .
. . . . . . . . 1 1 2 2 2 1 1 . . . . . . . 1 2 3 4 5 5 4 3 3 2 2 1 1 . . . . .
. . . . . . . 1 1 2 2 3 2 2 1 1 . . . . . . 1 2 3 4 5 5 4 4 3 3 2 2 1 1 . . . .
. . . . . . . 1 2 2 3 3 3 2 2 1 . . . . . . 1 2 3 4 5 5 5 4 4 3 3 2 2 1 1 . . .
. . . . . . . 1 2 3 3 4 3 3 2 1 . . . . . . 1 2 3 4 5 6 5 5 4 4 3 3 2 2 1 1 . .
. . . . . . . 1 2 3 4 4 4 3 2 1 . . . . . . 1 2 3 4 5 6 6 5 5 4 4 3 3 2 2 1 1 .
. . . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . 1 1 2 3 4 5 6 6 5 5 5 4 4 3 3 2 2 1 1
. . . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . 1 2 3 4 5 6 5 5 4 4 4 3 3 2 2 1 1 .
. . . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . 1 2 3 4 5 5 5 4 4 3 3 3 2 2 1 1 . .
. . . 1 1 1 1 2 3 4 5 4 3 2 1 1 1 1 1 . . . 1 2 3 4 5 5 4 4 3 3 2 2 2 1 1 . . .
. . . 1 2 2 2 2 3 4 5 4 3 2 2 2 2 2 1 . . . 1 2 3 4 5 4 4 3 3 2 2 1 1 1 . . . .
. . . 1 2 3 3 3 3 4 5 4 3 3 3 3 3 2 1 . . . 1 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 4 4 4 5 4 4 4 4 4 3 2 1 . . . 1 2 3 4 5 4 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 5 5 5 5 5 5 5 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 1 . . . . . . .
. . . 1 2 3 4 5 6 6 6 6 6 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 . . . . . . . .
. . . 1 2 3 4 5 6 7 7 7 7 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 1 . . . . . . .
. . . 1 2 3 4 5 6 7 7 7 7 6 5 4 3 2 1 1 1 1 1 2 3 4 5 4 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 5 6 6 7 7 6 6 5 4 3 2 2 2 2 2 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 5 5 6 6 6 5 5 4 3 3 3 3 3 3 3 3 4 5 4 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 4 5 5 6 6 6 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 3 4 4 5 5 5 5 5 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 1 . . . . . .
. . . 1 2 2 3 3 4 4 5 5 4 4 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 . . . . . . .
. . . 1 1 2 2 3 3 4 4 4 4 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . 1 1 2 2 3 3 4 4 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 2 2 3 3 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 2 2 2 2 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . .
```

Skeleton Extraction

```
30 40 0 7
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 5 5 . . . . . . . . . . . . . . . .
. . . . . 3 . . . . . . . . . . . . . . . . . . . . 5 5 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 4 . . . . . . . . . . . . . . . . . . . . 6 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 6 6 . 5 . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . 1 . . . . . 6 6 . 5 5 . 4 . 3 . 2 . 1
. . . . . 5 . . . . . . . . . . . . . . . . . . . . 6 . . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . . . . . . . . . . 5 5 . . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . 5 . . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . 7 7 7 7 . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . 7 7 7 7 . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . 7 7 . . . . . . . . . . . . . . . . . . . 5 . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 5 . 4 . 3 . 2 . 1 . . . . .
. . . . . 6 6 . . . . 4 4 4 4 4 4 4 4 4 . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 5 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 4 4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

- Print outFile2

```
First Pass Expansion for 8-cocnnectness
30 40 0 7
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . 1 2 1 . . . . . . . . . . . . . 4 4 4 4 3 2 1 . . . . . . . .
. . . . . . . . . 2 2 2 1 . . . . . . . . . . . 3 4 5 5 4 3 2 1 . . . . . . . .
. . . . . . . . 1 2 3 2 1 . . . . . . . . . . 2 3 4 5 5 4 3 2 1 . . . . . . . .
. . . . . . . . 1 3 3 3 2 1 . . . . . . . . 1 2 3 4 5 5 5 4 3 2 1 . . . . . . .
. . . . . . . . 2 3 4 3 2 1 . . . . . . . . 1 2 3 4 5 6 5 5 4 4 3 2 1 . . . . .
. . . . . . . 1 2 4 4 4 3 2 1 . . . . . . . 1 2 3 4 5 6 6 5 5 4 4 3 3 2 2 1 1 .
. . . . . . . 1 3 4 5 4 3 2 1 . . . . . . 1 1 2 3 4 5 6 6 5 5 5 4 4 3 3 2 2 1 1
. . . . . . . 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 6 5 5 4 4 4 3 3 2 2 1 1 .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 5 5 4 4 3 3 3 2 2 1 1 . .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 5 4 4 3 3 2 2 2 1 1 . . .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 4 4 3 3 2 2 1 1 1 . . . .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 4 3 2 2 1 1 . . . . . . .
. . . . . . 1 2 3 4 4 4 3 2 1 . . . . . . . 1 2 3 4 5 4 3 2 1 1 . . . . . . . .
. . . . . . 1 2 6 6 6 6 6 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 . . . . . . . . .
. . . . . . 1 5 6 7 7 7 7 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 . . . . . . . . .
. . . . . . 4 5 6 7 7 7 7 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 . . . . . . . . .
. . . . . 3 4 5 6 6 7 7 6 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . . 2 3 4 5 5 6 6 6 6 5 5 4 3 3 3 3 3 3 3 3 3 4 5 4 4 3 3 2 2 1 1 . . . . .
. . . 1 2 3 4 4 5 5 6 6 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 3 4 4 5 5 5 5 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 1 . . . . . . .
. . . 1 2 2 3 3 4 4 5 5 4 4 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 . . . . . . . .
. . . 1 1 2 2 3 3 4 4 4 4 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . .
. . . . 1 1 2 2 3 3 4 4 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 2 2 3 3 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . 1 1 2 2 2 2 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . .
```

```
Second Pass Expansion for 8-cocnnectness
30 40 0 6
. . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . 1 2 2 2 2 2 2 2 2 1 . . . . . . . .
. . . . . . . . . 1 1 1 . . . . . . . . . . 1 2 3 3 3 3 3 3 2 1 1 . . . . . . .
. . . . . . . . 1 1 2 1 1 . . . . . . . . . 1 2 3 4 4 4 4 3 2 2 1 1 . . . . . .
. . . . . . . 1 1 2 2 2 1 1 . . . . . . . . 1 2 3 4 5 5 4 3 3 2 2 1 1 . . . . .
. . . . . . 1 1 2 2 3 2 2 1 1 . . . . . . . 1 2 3 4 5 5 4 4 3 3 2 2 1 1 . . . .
. . . . . . 1 2 2 3 3 3 2 2 1 . . . . . . . 1 2 3 4 5 5 5 4 4 3 3 2 2 1 1 . . .
. . . . . . 1 2 3 3 4 3 3 2 1 . . . . . . . 1 2 3 4 5 6 5 5 4 4 3 3 2 2 1 1 . .
. . . . . . 1 2 3 4 4 4 3 2 1 . . . . . . . 1 2 3 4 5 6 6 5 5 4 4 3 3 2 2 1 1 .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . 1 1 2 3 4 5 6 6 5 5 5 4 4 3 3 2 2 1 1
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 6 5 5 4 4 4 3 3 2 2 1 1 .
. . . . . . 1 2 3 4 5 4 3 2 1 . . . . . . . 1 2 3 4 5 5 5 4 4 3 3 3 2 2 1 1 . .
. . . 1 1 1 1 2 3 4 5 4 3 2 1 1 1 1 1 . . . 1 2 3 4 5 5 4 4 3 3 2 2 2 1 1 . . .
. . . 1 2 2 2 2 3 4 5 4 3 2 2 2 2 2 1 . . . 1 2 3 4 5 4 4 3 3 2 2 1 1 1 . . . .
. . . 1 2 3 3 3 3 4 5 4 3 3 3 3 3 2 1 . . . 1 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 4 4 4 5 4 4 4 4 4 3 2 1 . . . 1 2 3 4 5 4 3 2 2 1 1 . . . . . . .
. . . 1 2 3 4 5 5 5 5 5 5 5 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 1 . . . . . . . .
. . . 1 2 3 4 5 6 6 6 6 6 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 . . . . . . . . .
. . . 1 2 3 4 5 6 7 7 7 7 6 5 4 3 2 1 . . . 1 2 3 4 5 4 3 2 1 1 . . . . . . . .
. . . 1 2 3 4 5 6 7 7 7 7 6 5 4 3 2 1 1 1 1 1 2 3 4 5 4 3 2 2 1 1 . . . . . . .
. . . 1 2 3 4 5 6 6 7 7 6 6 5 4 3 2 2 2 2 2 2 2 3 4 5 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 4 5 5 6 6 6 6 5 5 4 3 3 3 3 3 3 3 3 3 4 5 4 4 3 3 2 2 1 1 . . . . .
. . . 1 2 3 4 4 5 5 6 6 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 2 2 1 1 . . . . . .
. . . 1 2 3 3 4 4 5 5 5 5 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 1 . . . . . . .
. . . 1 2 2 3 3 4 4 5 5 4 4 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 . . . . . . . .
. . . 1 1 2 2 3 3 4 4 4 4 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . .
. . . . 1 1 2 2 3 3 4 4 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 2 2 3 3 3 3 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . 1 1 2 2 2 2 2 2 1 1 . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . .
```

- Print skeleton file

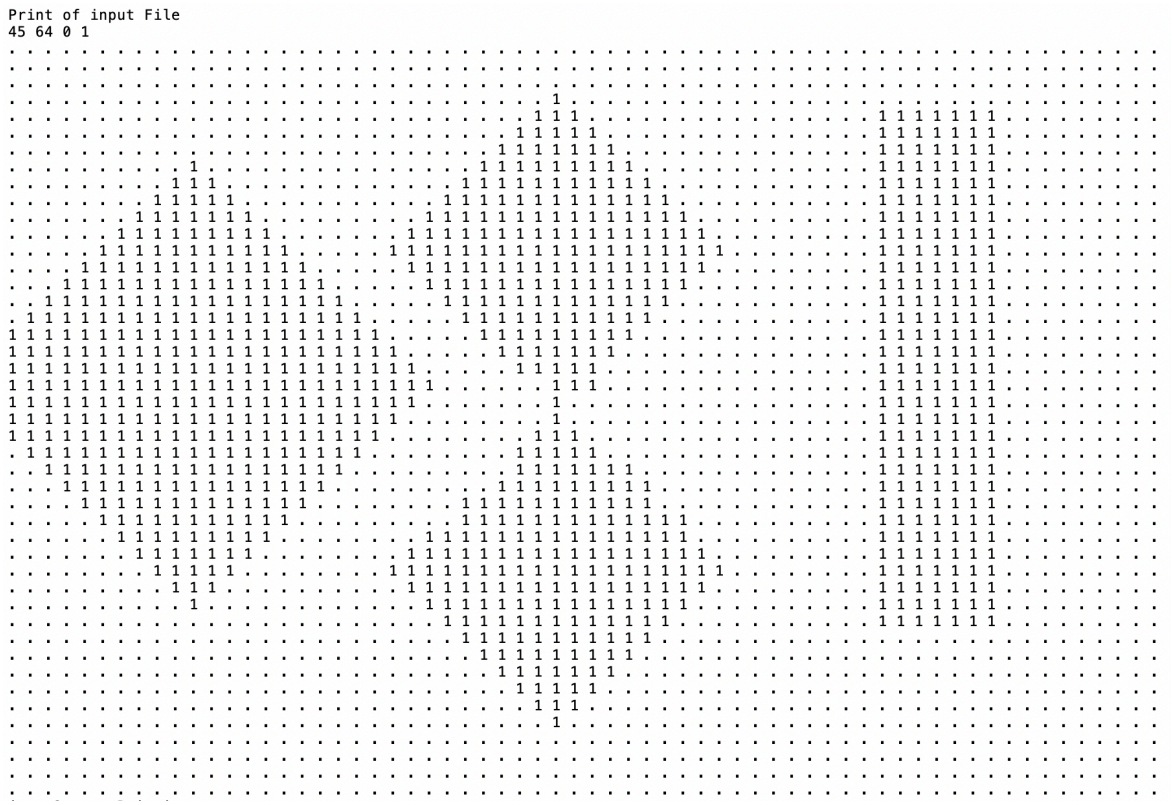
2 11 1
4 11 2
5 27 5
5 28 5
6 11 3
6 27 5
6 28 5
8 11 4
8 28 6
9 28 6
9 29 6
9 31 5
10 11 5
10 22 1
10 28 6
10 29 6
10 31 5
10 32 5
10 34 4
10 36 3
10 38 2
10 40 1
11 11 5
11 28 6
12 11 5
13 11 5
13 27 5
13 28 5
14 11 5
14 27 5
15 11 5
15 27 5
16 11 5
16 27 5
17 27 5
18 27 5
19 10 7
19 11 7
19 12 7
19 13 7
19 27 5
20 10 7

20 11 7
20 12 7
20 13 7
20 27 5
21 11 7
21 12 7
21 27 5
22 27 5
22 29 4
22 31 3
22 33 2
22 35 1
23 11 6
23 12 6
23 17 4
23 18 4
23 19 4
23 20 4
23 21 4
23 22 4
23 23 4
23 24 4
23 25 4
25 11 5
25 12 5
27 11 4
27 12 4

- Print decompressed file

```
Decompressed File
30  40  0  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 1 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 1 1 1 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 1 1 1 1 1 0 0 0 0 0 0 0 0 0 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 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 1 1 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 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 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 0 0 0 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 0 0 0 0 0 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 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- Print the input file for Image 2



- Print outFile1

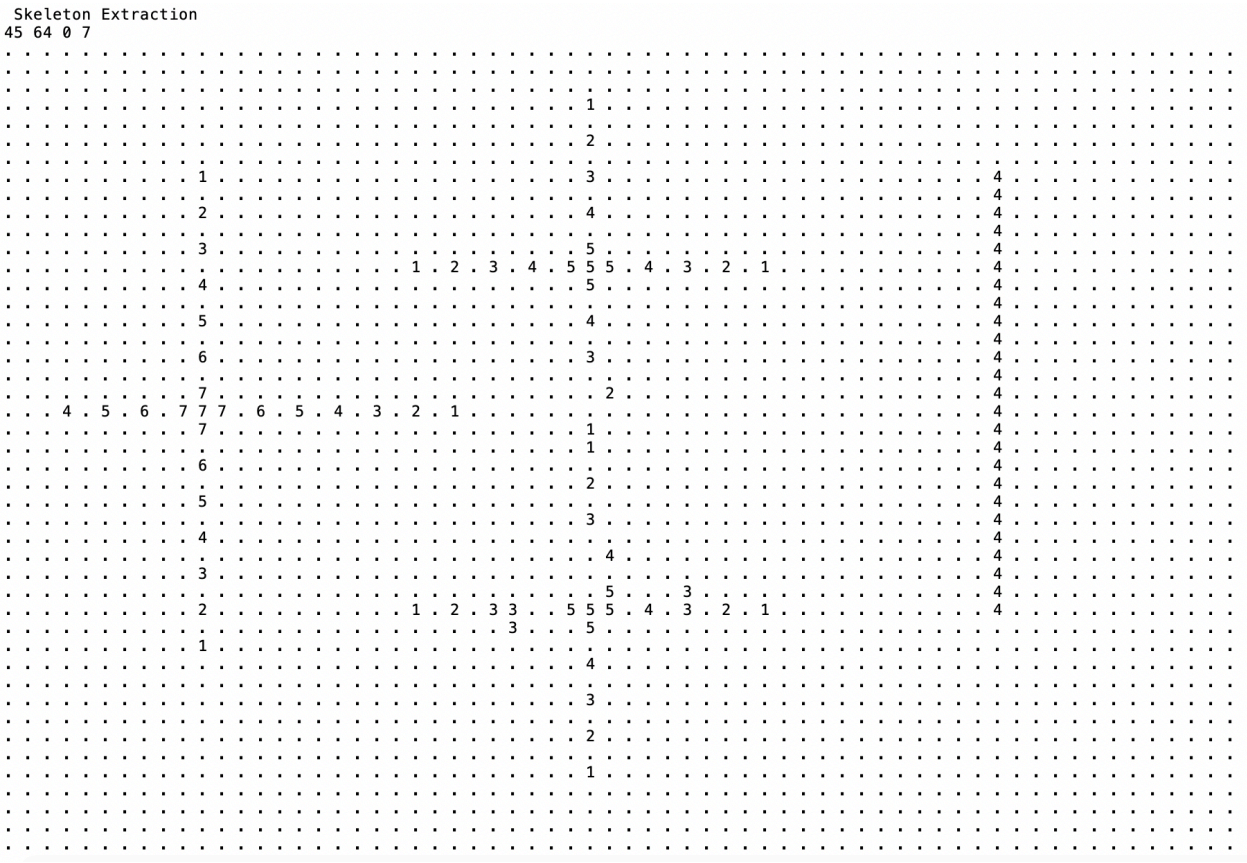
First Pass for 8-Connected Distance

45 64 0 9

Seconde Pass for 8-Connected Distance

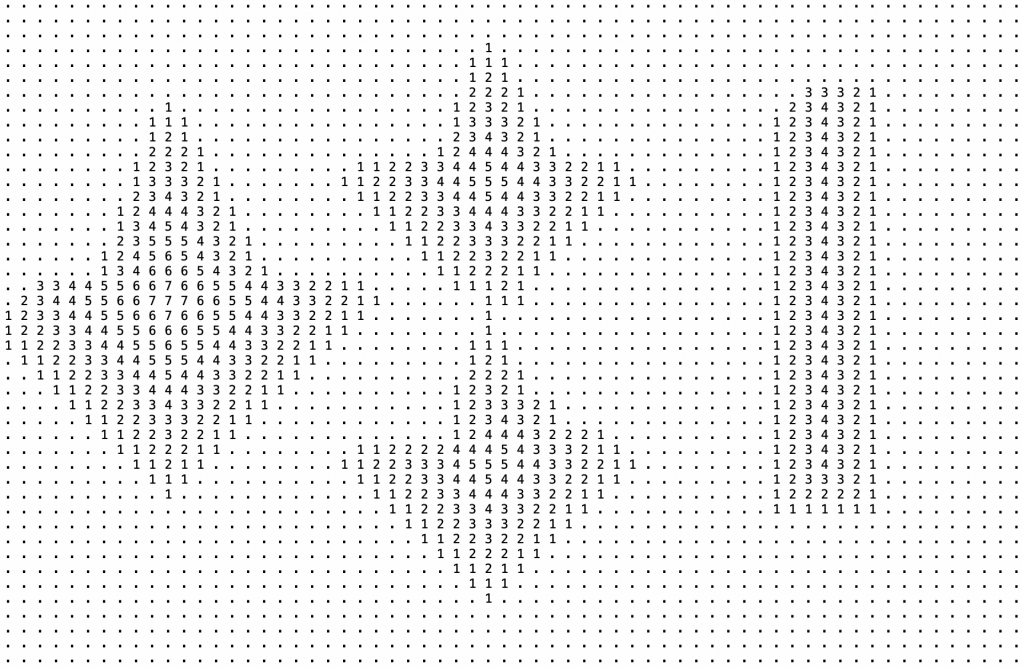
45 64 0 7

Skeleton Extraction
45 64 0 7



- Print outFile2

First Pass Expansion for 8-Connectness
45 64 0 7



- Print skeleton file

```
4  31  1
6  31  2
8  11  1
8  31  3
8  52  4
9  52  4
10  11  2
10  31  4
10  52  4
11  52  4
12  11  3
12  31  5
12  52  4
13  22  1
13  24  2
13  26  3
13  28  4
13  30  5
13  31  5
```

13 32 5
13 34 4
13 36 3
13 38 2
13 40 1
13 52 4
14 11 4
14 31 5
14 52 4
15 52 4
16 11 5
16 31 4
16 52 4
17 52 4
18 11 6
18 31 3
18 52 4
19 52 4
20 11 7
20 32 2
20 52 4
21 4 4
21 6 5
21 8 6
21 10 7
21 11 7
21 12 7
21 14 6
21 16 5
21 18 4
21 20 3
21 22 2
21 24 1
21 52 4
22 11 7
22 31 1
22 52 4
23 31 1
23 52 4
24 11 6
24 52 4
25 31 2
25 52 4
26 11 5
26 52 4
27 31 3

27 52 4
28 11 4
28 52 4
29 32 4
29 52 4
30 11 3
30 52 4
31 32 5
31 36 3
31 52 4
32 11 2
32 22 1
32 24 2
32 26 3
32 27 3
32 30 5
32 31 5
32 32 5
32 34 4
32 36 3
32 38 2
32 40 1
32 52 4
33 27 3
33 31 5
34 11 1
35 31 4
37 31 3
39 31 2
41 31 1

- Print decompressed file

Decompressed File

```
45  64  0  1
```