Project 7 (in Java): Given a binary image, the task is to produce a loss-less compression of the input image via the skeleton of 8-connectness distance transform. (Read the class note sent by your TA.)

Summary of what your program will do:

1) Opens the input file and load to a 2D array with extra 2 rows and extra 2 cols.

2) Performs the 1st-pass of the 8-connectness distance transform on input begins at (1,1) and ends at (numRows, numCols).

3) PrettyPrint the result of 1st-pass to outFile1 with captions.

4) Performs the 2nd-pass of the 8-connectness distance transform on the result of 1st pass, begins at (numRows, numCols).

5) PrettyPrint the result of 2nd pass to outFile1 with captions.

6) Performs local maxima operation on the result of 2nd-pass.

p(i, j) is a local maxima

if p(i, j) >= all its 3x3 8-neighbors

skeleton (i,j) 🡨 p(i, j) // retaining the distance

else

skeleton (i,j) 🡨 0

7) PrettyPrint the local maxima to outFile1 with captions.

8) Produce skeleton: for each skeleton (i, j) > 0 (i.e., local maxima),

write a triplet i j skeleton (i,j) to \*skeleton\* file,

one triplet per text-line

// skeleton file is the compressed (skeleton) file.

9) The name of the compressed file is to be created during the run time of your program, using the original file name with an extension “\_skeleton.” For example, if the name of the input file is “image1”, then the name of the compressed file should be “image1\_skeleton”. (This can be done simply using string concatenation.)

10) close the compressed file (image1\_skeleton)

// To make sure your program works correctly; you are going to do a de-compression on the compressed file as follows.

11) re-open the compressed file (image1\_skeleton).

12) re-set ZeroFramedAry to zero

13) Load triplets from compressed file to ZeroFramedAry, i.e., for

each triplet (i, j, dist), ZeroFramedAry(i, j) 🡨 dist

14) Perform 1st-pass expansion on the ZeroFramedAry

// algorithm given below

15) PrettyPrint the result of 1st-pass expansion to outFile2 with captions.

16) Perform 2nd pass expansion on the result of 1st expansion

// algorithm given below

17) PrettyPrint the result of 2nd-pass expansion to outFile2 with caption.

// If your program work correctly, the result of 2nd-pass expansion should be

// identical to the result of the 2nd pass of distance transform.

18) Produce decompressed file:

a) Write the original image header to the decompressed file

b) Threshold ZeroFramedAry with threshold value == 1 begins at (1,1)

and ends at (?,?)

i.e., if ZeroFramedAry (i, j) >= 1

output 1 and a blank space to de-compressed file.

else

output 0 and a blank space to de-compressed file.

19) The name of the decompressed file is to be created during the run time of your program, using the name of the input file with an extension “\_decompressed.” For example, if the name of the input file is “image1”, then the name of the compressed file should be “image1\_decompressed”. (This can be done simply using string concatenation.)

20) Closed the de-compressed file.

// after this step your directory should have these three files: image1, image1\_skeleton, and image1\_decompressed.

21) If your program works correctly, image1\_decompressed should be identical to image1.

22) run your program twice: with image1 and image2

Include in your hard copies:

- cover page

- source code

- Run on image1

- Print the input file

- Print outFile1

- Print outFile2

- Print skeleton file

- Print decompressed file

- Run on image2

- Print the input file

- Print outFile1

- Print outFile2

- Print skeleton file

- Print decompressed file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: Java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Points: 12 pts

Due Date: Soft copy and pdf hard copies: 3/29/2020 Sunday before midnight

1 day late: -1 pt 3/30/2020 Monday before midnight

2 days late: -3 pts 3/31/2020 Tuesday before midnight

-12 pts: after 3/31/2020 Tuesday after midnight

\*\*\* Name your pdf file using the same format as your soft copy except

instead of \_CPP use \_HardCopy; for example if your name is Joe Golden

then pdf name would be GoldenJ\_Project6\_HardCopy.pdf

\*\*\* All on-line submission MUST include Soft copy and pdf hard copy in the same email with proper file names.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. Input (args[0]): a binary image

II. Outputs: \*\* Please make sure all pixels are line-up nicely for good visualization (use font currier New)

- OutFile1 (args[1]): for

- prettyPrint the results of 1st pass 8-connectness

distance transform

- prettyPrint the results of 2nd pass 8-connectness

distance transform

- prettyPrint the local maxima

- OutFile2 (args[2]): for

- prettyPrint the results of 1st pass expansion

- prettyPrint the results of 2nd pass expansion

- skeleton file (generated at run-time) for store the compressed file

using the following format:

Example:

20 20 0 7 // the header of the distance transform image.

4 7 2 // the skeleton pixel at (4, 7) with distance of 2

6 7 3 // the skeleton pixel at (6, 7) with distance of 3

:

:

- DeCompressed file (generated at run-time)- is an image file where

the first text-line is the image header, follows by rows and cols of pixel values.

// The algorithm steps for all methods given in the specs

// may contain bugs, you are responsible to debug it.

// However, when you find a bug, let me know.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- An ImageProcessing class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- newMinVal (int)

- newMinVal (int)

- zeroFramedAry (int \*\*) a 2D array, need to dynamically allocate

of size numRows + 2 by numCols + 2.

- skeletonAry (int \*\*) a 2D array, need to dynamically allocate

of size numRows + 2 by numCols + 2.

- methods:

- setZero (Ary) // set 2D Ary to zero. You should know how to do this.

- loadImage (...)

// Read from the given File onto zeroFramedAry

// the first pixel of File is loaded

// at zeroFramedAry [1][1]

// You should know how to do this.

- Compute8Distance (...) // See algorithm below

- fistPass8Distance (Ary) // On your own

// Scan from L -> R & T -> B begin at pixel (1,1)

// look at the upper 3 neighbor and the left neighbor

// a b c

// d x

// Ary[i, j] 🡨 min(a, b, c, d) + 1

// You should know how to do this

- secondPass\_8Distance (zeroFramedAry) // on your own

// Scan from R -> L & B -> T

// begin at pixel (numRows, numCols)

// look at itself, x, the right neighbor and the lower 3 neighbor

// x e

// f g h

// Ary[i, j] 🡨 min(x, 1+ min(e, f, g, h))

// Note\*\* In second pass, you need

// to keep track the newMinVal and newMaxVal

// You should know how to do this

- skeletonExtraction (...) // See algorithm below

- isLocalMaxima (zeroFramedAry, i, j)

// if zeroFramedAry (i, j)>= all its \*8-connected\* neighbors

return 1

else return 0

- computeLocalMaxima (zeroFramedAry, skeletonAry)

// if zeroFramedAry (i,j) > 0

// and isLocalMaxima (zeroFramedAry, i, j)

skeletonAry(i,j) = zeroFramedAry(i,j)

else

skeletonAry (i, j) = 0

- extractLocalMaxima(...)

// extract and write each local maxima as triplet to skeletonFile

- skeletonExpansion(...)// See algorithm below.

- firstPassExpension (...)// See algorithm below.

- secondPassExpension (...)// See algorithm below.

- ary2File(...)

// do a threshold on zeroFramedAry

// with the threshold value at 1, begins at (1,1)

// and ends at (?,?)

i.e., if zeroFramedAry (i, j) >= 1

output 1 and a blank space to decompressed file.

else

output 0 and a blank space to decompressed file.

- prettyPrint (Ary, outFile)

// if Ary(i,j) == 0 print 2 blank space

else print Ary(i,j) use 2 digit space

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. main (…)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open input file

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate zeroFramedAry with extra 2 rows and 2 cols

dynamically allocate skeletonAry with extra 2 rows and 2 cols

open outFile\_1, outFile\_2

Step 1: skeletonFileName 🡨 argv[1] + “\_skeleton”

Step 2: skeletonFile 🡨 open ( skeletonFileName )

Step 3: decompressedFileName 🡨 argv[1] + “\_decompressed”

Step 4: decompressFile 🡨 open (decompressedFileName)

step 5: setZero (zeroFramedAry)

setZero (skeletonAry)

Step 6: loadImage (inFile, zeroFramedAry) // begins at zeroFramedAry (1,1)

Step 7: compute8Distance (zeroFramedAry, outFile1) // Perform distance transform

Step 8: skeletonExtraction (zeroFramedAry, skeletonAry, skeletonFile, outFile1)

// perform lossless compression

Step 9: skeletonExpansion (zeroFramedAry, skeletonFile, outFile2)

// perform decompression

step 10: Output numRows, numCols, newMinVal, newMaxVal to decompressFile

Step 11: ary2File (zeroFramedAry, decompressFile)

Step 12: close all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. Compute8Distance (zeroFramedAry, outFile1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 1: fistPass\_8Distance (zeroFramedAry) // begins at ZeroFramedAry(1,1)

step 2: prettyPrint (zeroFramedAry, outFile1)

// with proper caption i.e., 1st pass distance transform

step 3: secondPass8Distance (zeroFramedAry) // begins at zeroFramedAry(?,?)

Step 4: prettyPrint (zeroFramedAry, outFile1)

// with proper caption i.e., 2nd pass distance transform

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V. skeletonExtraction (zeroFramedAry, skeletonAry, skeletonFile, outFile1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 1: computeLocalMaxima (zeroFramedAry, skeletonAry)

Step 2: prettyPrint (skeletonAry, outFile1)

// with proper caption i.e., Local maxima

step 3: extractLocalMaxima (skeletonAry, skeletonFile)

// for each skeletonAry[i,j] > 0 write the triplet to

// skeletonFile. For easy programming, i and j do not need to

// subtract by 1 when output the triplets to skeletonFile.

Step 4: close skeletonFile

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VI. skeletonExpansion (zeroFramedAry, skeletonFile, outFile2)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step 1: re-open skeletonFile

Step 2: setZero (zeroFramedAry)

step 3: load (skeletonFile, zeroFramedAry)

step 4: firstPassExpension (zeroFramedAry) // See algorithm below

step 5: prettyPrint (zeroFramedAry, outFile2)

// with proper caption i.e., 1st pass Expansion

step 6: secondPassExpension (zeroFramedAry) // begins at ZeroFramedAry(?,?)

// During the 2nd pass, you need to track the newMinVal and newMaxVal

Step 7: prettyPrint (zeroFramedAry, outFile2)

// with proper caption i.e., 2nd pass Expansion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VII. firstPassExpension (Ary)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step 1: Scan Ary L to R & T to B begins at (1,1)

// use index i for row and index j for column

step 2: Ary(i,j) 🡨 next pixel

Step 3: if Ary (i, j) == 0

Ary (i, j) 🡨 max (0, max (its 3x3 8-neighbors) - 1)

step 4: repeat step 1 to step 3 until all pixels are processed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VII. secondPassExpension (Ary)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step 1: Scan Ary R to L & B to T begins at (? ,?)

// use index i for row and index j for column

step 2: Ary(i,j) 🡨 next pixel

Step 3: Ary (i, j) 🡨 max (Ary(i, j), max (its 3x3 8-neighbors) - 1)

step 4: repeat step 1 to step 3 until all pixels are processed