Project 4A: You are to implement the distance transform (using Euclidian distance)

then, from the resulting distance transform to extract the skeletons of object

using pixel’s 8-neighbors. (Use the algorithm taught in class.)

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

Language: Java

Due date: Soft copy: 10/3/2017 Wednesday before Midnight

hard copy: 10/4/2017 Thursday in class

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

I. Input: a binary image

// Use argv[1]

II. Outputs:

- OutFile\_1 Use argv[2] for:

The result of the Euclidian distance transform image with

\*newRowVal and newColVal\* image header for future processing.

- OutFile\_2 Use argv[3] to do:

Output the result of the skeleton image with

\*newRowVal and newColVal\* image header for future processing.

- OutFile\_3 Use argv[4] to do: (For visualization!)

a) Pretty print the result of the Pass-1 of distance transform

with proper caption.

// use blank space for background 0’s.

b) Pretty print the result of the Pass-2 of distance transform

with proper caption.

// use blank space for background 0’s.

c) Pretty print the result of Skeleton array

with proper caption.

// use blank space for background 0’s.

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

III. Data structure:

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

- An ImageProcessing class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- newMinVal (int)

- newMaxVal (int)

- ZeroFramedAry (int \*\*) a 2D array, need to dynamically allocate at run time

of size numRows + 2 by numCols + 2.

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

of size numRows + 2 by numCols + 2.

- methods:

- constructor(s)

- need to dynamically allocate ZeroFrameAry and skeletonAry

- assign values to numRows,..., etc.

- zeroFramed (ZeroFramedAry)

- loadImage (ZeroFramedAry)

// Read from the input file onto ZeroFrameAry

// the first pixel of input image is loaded

// at ZeroFrameAry[1][1]

- fistPass\_EuclidianDistance (ZeroFramedAry)

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

// Use the algorithm taught in class

- secondPass\_EuclidianDistance (ZeroFramedAry)

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

// begin at pixel (?,?)

// Use the algorithm taught in class

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

// to keep track the newMinVal and newMaxVal

- is\_maxima (ZeroFramedAry, i,j)

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

return 1

else return 0

- compute\_localMaxima(ZeroFramedAry, skeletonAry)

// if ZeroFramedAry (i,j) > 0

// and is\_maxima (ZeroFramedAry, i,j)

skeletonAry(i,j) = 1

else

skeletonAry(i,j) = 0

- printImage(Ary, outFile) // You are to use this method

// to output distance image and the skeleton image.

- prettyPrintDistance (ZeroFramedAry)// to outFile3

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

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

- prettyPrintSkeleton (skeletonAry) // to outFile3

// if p(i,j) == 0 print ‘**.**’ // a dot

else print ‘9’

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

III. Algorithms

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

step 0: inFile 🡨 open input file

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate zeroFramedAry and skeletonAry with extra 2 rows and 2 cols

step 1: zeroFramed (ZeroFramedAry)

Step 2: loadImage (ZeroFramedAry)

step 3: fistPass\_EuclidianDistance (ZeroFramedAry)

step 4: prettyPrintDistance (ZeroFramedAry)to outFile\_3

// with proper caption i.e., Pass-1 result

step 5: secondPass\_EuclidianDistance (ZeroFramedAry)

Step 6: output newMinVal and newMaxVal to outFile\_1

Step 7: printImage(ZeroFramedAry, outFile\_1)

// output the result of Pass-2 to outFile\_1

(\*without\* the 2 extra rows and columns)

Step 8: prettyPrintDistance (ZeroFramedAry) // to outFile\_3

// with proper caption i.e., Pass-2 result

step 9: compute\_localMaxima(ZeroFramedAry, skeletonAry)

// see this method given in the above

Step 10: output newMinVal and newMaxVal to outFile\_2

Step 11: printImage(skeletonAry, outFile\_2)

// output the result of skeleton to outFile\_2

(\*without\* the 2 extra rows and columns)

Step 12: prettyPrintSkeleton (skeletonAry)// to outFile\_3

Step 13: close all files