Project 4: Compute the distance transform (using 8-distance) of the input image

Using the algorithm taught in class.

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

Language: C++

Due date: C++ soft copy: 2/23/2018 Friday before midnight

+1 pt for early submission, deadline: 2/21/2018 Wednesday before midnight

Due date: C++ hard copy: 2/27/2017 Tuesday in class before exam

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

I. Input: a binary image

// Use argv[0]

II. Outputs (argv[]):

1) outFile1 – Create a distance transform image from the result of Pass-2

with updated image header (numRows numCols newRowVal newColVal) for future processing.

2) outFile2 for visualization

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

with proper caption

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

with proper caption

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

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.

- NeighborAry[5](int)

- methods:

- constructor(s)

- need to dynamically allocate ZeroFrameAry

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

- zeroFramed

- loadImage

// Read from the input file onto ZeroFrameAry

// the first pixel of input image is loaded

// at ZeroFrameAry[1][1]

- loadNeighbors // load the respective neighbors of given pixel(i,j)

- fistPassDistance

// Using the algorithm steps given in class.

- secondPassDistance

// Using the algorithm steps given in class.

- prettyPrint

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

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

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

III. Algorithms

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

step 0: read the image header

dynamically allocate zerpFramedAry with extra 2 rows and 2 cols

step 1: zeroFrame the zerpFramedAry.

Step 2: loadImage

step 3: firstPassDistance (…) // 8-distance algorithm taught in class

step 4: prettyPrintof the result of Pass-1 to outFile2

with caption indicating the result of pass-1,

step 5: secondPassDistance (…) // 8-distance algorithm taught in class

// In second pass, you need to keep track the newMinVal and newMaxVal

Step 6: prettyPrintDistace of the result of Pass-2 to outFile2 w/ caption

Step 7: Write the result of Pass-2 (without the 2 extra rows and columns) to outFile1 with updated image header.

Step 8: close all files.