Project 5 (in C++): Given a binary image, the task is to perform a loss-less compression via the distance-transform (4-distance, or 8-distance or Euclidean distance). Your program will let the user choose which distance to run the program, via argv [2]. To insure the correctness of your compression, you will perform a decompression on the compressed file.

There are two parts of this project: a) binary image compression; b) decompression.

*** Compression:
1) Dynamically allocate a 2D ZFAry with 2 extra rows and 2 extra columns (zero-framed), and load input data onto inside frame of ZFAry.
2) Performs 1st pass and 2nd pass of distance-transform for all pixels inside the frame of ZFAry.
3) Performs local maxima operation on the result of 2nd pass distance-transform which produce compressed file.
*** Decompression:

4) Close and re-open the compressed file and load the compressed data into ZFAry.
5) Perform 1st pass and 2nd pass of the expansion operations on ZFAry.
// If your program works correctly, the result of 2nd pass expansion should be identical to the result of
// the 2nd pass of distance-transform. You should check for the correctness of your program before submission!
6) Perform the binary threshold operation on the result of the 2nd pass expansion, using threshold value 1.
// If your program works correctly, your decompressed file should be identical to input image. You should check for
//the correctness of your program before submission!

*** What do you need to do:
a) Implement your program based on the specs given below and debug your program until it passes compilation.
b) You will be given 2 data files: img1 and img2. Run and debug your program with img1 using 4-distanceuntil your program produces the decompress file is identical to img1.
c) When the result is correct, run your program with img1 using 8-distance.
d) Run your program with img2 **using 8-distance only.**
==========

Include in your hard copies:
- cover page (include only the main () algorithm steps, -1 otherwise)
- source code
- prettyPrintFile for img1 using 4-distance
- skeletonFile for img1 using 4-distance
- deCompressedFile for img1 using 4-distance
- logFile for img1 using 4-distance // limited to 3 pages if more.

- prettyPrintFile for img1 using 8-distance
- skeletonFile for img1 using 8-distance
- deCompressedFile for img1 using 8-distance
- logFile for img1 using 8-distance // limited to 3 pages if more.

- prettyPrintFile for img2 using 8-distance
- skeletonFile for img2 using 8-distance
- deCompressedFile for img2 using 8-distance
- logFile for img2 using 8-distance // limited to 3 pages if more.

****

Language: C++
****

Project Name: Image Compression via Distance Transform
Project points: 12pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
+1 (13/12 pts): early submission, 10/17/2024, Thursday before midnight
(12/12 pts): on time, 10/20/2024 Sunday before midnight
(-12/12 pts): non-submission, 10/20/2024 Sunday after midnight
*** Name your soft copy and hard copy files the naming convention given in Project Submission Requirements. (-2 if not.)
*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in the same email attachments with correct email subject as below; otherwise, your submission will be rejected.

Email subject: (CV) first name last name <Project 5: Image Compression vis Distance Transform (C++)>

*** Inside the email body include your answer to the 4 questions. Optional screen recording if you wish.

****

I. Inputs:

a) inFile (argv [1]): a txt file representing a binary image with header.

b) distance choice (argv [2]): 4 for 4-distance (City-block); 8 for 8-distance, 10 for Euclidean distance

// You are not implement Euclidean distance in this project.

****

II. Outputs:

a) prettyPrintFile (argv [3]): as specs dictates.

b) skeletonFile (argv [4]): The compressed file using the following format:

Example:

20 30 0 25 // 20 rows, 30 columns, minVal is 0 and maxVal is 25.

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

6 7 5 // the skeleton pixel at (6, 7) with distance of 5

:

:

c) deCompressedFile (argv [5]): This is a binary image, the result of the decompressed file.

d) logFile (argv [6]): as specs dictates.

****

III. Data structure:

****

- a distanceSkeleton class
  - (int) numRows
  - (int) numCols
  - (int) minVal
  - (int) maxVal
  - (int) newMinVal
  - (int) newMinVal
  - (int **) ZFAry //a 2D integer array, need to dynamically allocate of size numRows + 2 by numCols + 2.
  - (int **) skeletonAry //a 2D array, need to dynamically allocate of size numRows + 2 by numCols + 2.
  - (int) distanceChoice // 4 or 8.
- methods:
  - setZero (Ary) // set 2D Ary to zero.
  - loadImage (…) // Load input onto inside frame of ZFAry.
  - loadSkeleton (…) // Load the skeleton file onto inside frame of ZFAry.
  - DistanceTransform (...) // // Perform distance transform. **See algorithm below**.
  - DistancePass1 (…) // algorithm is given in class. If P[i, j] > 0

    P[i, j] ← for 4 distance is min (a+2, b+1, c+2, d+1)

    For 8-distance is min (a+1, b+1, c+1, d+1)

    For Euclidean ?

    // Note** In Pass1, you need to keep track the newMinVal and newMaxVal.
  - DistancePass2 (…) // algorithm is given in class.

    P[i, j] ← for 4 distance is min (e+1, f+2, g+1, h+2, p[i, j])

    For 8-distance is min (e+1, f+1, g+1, h+1, p[i, j])

    For Euclidean ?

    // Note** In Pass2, you need to keep track the newMinVal and newMaxVal.
  - compression (...) // // Perform compression. **See algorithm below.**
  - isLocalMaxima (…)// algorithm is given in class. If P[i, j] > 0

    // P [i, j] is local maxima

    For 4 distance, if P [i, j] >= b, d, e, g

    For 8- distance, if P [i, j] >= a, b, c, d, e, f, g, h

    For Euclidean ?
  - computeLocalMaxima (ZFAry, skeletonAry,…)

    // if isLocalMaxima (ZFAry [i, j])

    skeletonAry [i, j] ← ZFAry [i, j]

    else

    skeletonAry [i, j] ← 0

- extractSkeleton (...) // if skeletonAry [i, j] > 0 write the triplet: i, j, skeletonAry[i,j] to skeletonFile.
- deCompression (...) // Perform decompression**; see algorithm below**.
- expensionPass1 (...)// algorithm is given in class.

If P[i, j] == 0

P[i, j] ← for 4 distance is max (a-2, b-1, c-2, d-1, e-1, f-2, g-1, h-2, P[i, j])
For 8-distance is min (a-1, b-1, c-1, d-1, e-1, f-1, g-1, h-1, P[i, j])
For Euclidean ?

- expensionPass2 (...)// algorithm is given in class.

For all P[i, j]

P[i, j] ← for 4 distance is max (a-2, b-1, c-2, d-1, e-1, f-2, g-1, h-2, P[i, j]),
For 8-distance is min (a-1, b-1, c-1, d-1, e-1, f-1, g-1, h-1, P[i, j])
For Euclidean ?

- binaryThreshold (...) // do a binary threshold on all pixels inside of ZFAry with the threshold value at 1;
i.e., if ZFAry (i, j) >= 1
output 1 and a blank space to deCompressed file.
else
output 0 and a blank space to deCompressed file.
- prettyPrint (…) // Re-use code from your previous Project;
// use "Courier New" font, smaller font size (but not too small) to display image within one page.

****

VI. main (…)

****

step 0: inFile, prettyPrintFile, skeletonFile, deCompressedFile, logFile ← open via argv []
numRows, numCols, minVal, maxVal ← read from inFile
dynamically allocate ZFAry and skeletonAry with extra 2 rows and 2 cols
distanceChoice ← get from argv [2]

Step 1: setZero (ZFAry)
setZero (skeletonAry)
Step 2: loadImage (inFile, ZFAry)
prettyPrint (ZFAry, prettyPrintFile) // with caption "** Below is input image**"
Step 3: distanceTransform (ZFAry, distanceChoice, prettyPrintFile, logFile)
Step 4: compression (ZFAry, distanceChoice, skeletonAry, skeletonFile, prettyPrintFile, logFile)
Step 5: close skeletonFile

Step 6: reopen skeletonFile
Step 7: setZero (ZFAry)
Step 8: loadSkeleton (skeletonFile, ZFAry, logFile)
prettyPrint (ZFAry, prettyPrintFile) // with caption "** Below is the loaded skeleton with choice = **"
Step 9: deCompression (ZFAry, distanceChoice, prettyPrintFile, logFile) // Perform decompression
Step 10: deCompressedFile ← output numRows, numCols, minVal, maxVal

Step 11: binThreshold (ZFAry, deCompressedFile)
Step 12: close all files

****

V. distanceTransform (ZFAry, distanceChoice, prettyPrintFile, logFile)

****

Step 0: logFile ← "Entering DistanceTransform () method."
Step 1: distancePass1 (ZFAry, distanceChoice, logFile)
Step 2: prettyPrint (ZFAry, prettyPrintFile) // with proper caption i.e., 1st pass distance transform with choice =
Step 3: distancePass2 (ZFAry, distanceChoice, logFile)
Step 4: prettyPrint (ZFAry, prettyPrintFile) // with proper caption i.e., 2nd pass distance transform with choice =
Step 5: logFile ← "Leaving DistanceTransform () method."

****
VI. compression (ZFAry, distanceChoice, skeletonAry, skeletonFile, prettyPrintFile, logFile)
****
Step 0: logFile ← "Entering compression () method,"
Step 1: computeLocalMaxima (ZFAry, skeletonAry, distanceChoice, logFile)
Step 2: prettyPrint (skeletonAry, prettyPrintFile)// with proper caption i.e., Local maxima, skeletonAry with choice =
Step 3: extractSkeleton (skeletonAry, skeletonFile, logFile)
prettyPrint (skeletonAry, logFile) // with caption: "In compression() Below is skeleton Array with choice = "
Step 4: logFile ← "Leaving compression () method."

****
VII. deCompression (ZFAry, distanceChoice, prettyPrintFile, logFile)
****
Step 0: logFile ← "Entering deCompression () method."
Step 1: expensionPass1 (ZFAry, distanceChoice, logFile)
Step 2: prettyPrint (ZFAry, prettyPrintFile) // with proper caption i.e., 1st pass Expansion with choice =
Step 3: expensionPass2 (ZFAry, distanceChoice, logFile)
Step 4: prettyPrint (ZFAry, prettyPrintFile) // with proper caption i.e., 2nd pass Expansion with choice =
Step 5: logFile ← "Leaving deCompression () method."