**CSC 381Image Processing (CPP)**

**Project: Thinning**

**Student: Jeffrey Pulinat**

**Language: C++**

**Due date: soft copy: 3/19/2018 Monday before midnight**

**Early submission deadline: 3/17/2018 Saturday before midnight**

**Due date: hard copy: 3/20/2018 Tuesday in class**

Main

step 1: inFile <-- open the input file, argv[1]

- open outFile1 and outFile2

- read image header from inFile

- output the image header to outFile1

- output image header to outFile2 // per text line

step 2: imgAry <-- dynamically allocated

step 3: loadImage()

step 4: zeroFramed ()

step 5: getChainCode()

step 6: close all files

getChainCode algorithm steps

step 1: scan imgAry from L to R & T to B

step 2: if imgAry(iRow, jCol) > 0

output iRow, jCol, imgAry(iRow, jCol) to outFile1 output iRow, jCol, imgAry(iRow, jCol) to outFile2

startP <-- (iRow, jCol)

currentP <-- (iRow, jCol)

lastQ <-- 4

step 3: nextQ <-- mod(lastQ, 8)

step 4: PchainDir <-- findNextP(currentP, nextQ, nextP)

// nextP will be determined inside the findNextP method.

step 5: output PchainDir to outFile1 // see format given in the above

output PchainDir to outFile2 // see format given in the above

step 6: lastQ <-- nextDirTable[PchainDir]

currentP <-- nextP // nextP was determined inside the findNextP method.

step 7: repeat step 3 to step 6 until currentP == startP

findNextP(currentP, nextQ, nextP)

step 1: loadNeighborsCoord (currentP)

step 2: chainDir <-- getChainDir(currentP, nextQ)

step 3: nextP <-- neighborCoord [chainDir]

step 4: returns chainDir

#include <iostream>

#include <fstream>

#include <string>

#include <stdlib.h>

#include "Point.h"

**using** **namespace** std**;**

static int numRows**;**

static int numCols**;**

static int minVal**;**

static int maxVal**;**

static int**\*\*** imgAry**;**

static int neighborCoord **[**8**];**

void zeroFramed**(**int **\*\***ary**,** int nRows**,** int nCols**){**

**for(**int col**=**0**;** col**<=**nCols**+**1**;** col**++){**

ary**[**0**][**col**]** **=** 0**;** //top

ary**[**nRows**+**1**][**col**]** **=** 0**;** //bottom

**}**

**for(**int row**=**0**;** row**<=**nRows**+**1**;** row**++){**

ary**[**row**][**0**]** **=** 0**;** //left side

ary**[**row**][**nCols**+**1**]** **=** 0**;** //right side

**}**

**}**

void loadImage**(**ifstream**&** input**){**

int param**;**

input **>>** param**;**

numRows **=** param**;**

input **>>** param**;**

numCols **=** param**;**

input **>>** param**;**

minVal **=** param**;**

input **>>** param**;**

maxVal **=** param**;**

imgAry **=** **new** int **\*[**numRows**+**2**];**

**for(**int i**=**0**;** i**<**numRows**+**2**;** **++**i**){**

imgAry**[**i**]** **=** **new** int**[**numCols**+**2**];**

**}**

//fill image

**for(**int row **=** 1**;** row**<=** numRows**;** row**++){**

**for(**int col **=** 1**;** col**<=** numCols**;** col**++){**

int value**;**

input **>>** value**;**

imgAry**[**row**][**col**]** **=** value**;**

**}**

**}**

**}**

void loadNeighborsCoord**(**int**\*\*** ary**,** int row**,** int col**)** **{** //load 3x3 neighbors

neighborCoord**[**0**]** **=** ary**[**row**][**col**+**1**];**

neighborCoord**[**1**]** **=** ary**[**row**-**1**][**col**+**1**];**

neighborCoord**[**2**]** **=** ary**[**row**-**1**][**col**];**

neighborCoord**[**3**]** **=** ary**[**row**-**1**][**col**-**1**];**

neighborCoord**[**4**]** **=** ary**[**row**][**col**-**1**];**

neighborCoord**[**5**]** **=** ary**[**row**+**1**][**col**-**1**];**

neighborCoord**[**6**]** **=** ary**[**row**+**1**][**col**];**

neighborCoord**[**7**]** **=** ary**[**row**+**1**][**col**+**1**];**

**}**

int getChainDir**(**Point currentP**,** int nextQ**){**

**for(**int i **=**nextQ**;** i**<=**7**;** i**++){**

**if(**neighborCoord**[**i**]>**0**){**

**return** i**;**

**}**

**}**

**for(**int i **=**0**;** i**<**nextQ**;** i**++){**

**if(**neighborCoord**[**i**]>**0**){**

**return** i**;**

**}**

**}**

**}**

int findNextP**(** Point currentP**,** int nextQ**,** Point **\***nextP**){**

int row **=** currentP**.**getX**();**

int col **=** currentP**.**getY**();**

loadNeighborsCoord **(**imgAry**,** row**,** col **);** //current non zero border pixel

int chainDir **=** getChainDir**(**currentP**,** nextQ**);**

//getting row from chainDir

**if(**chainDir**==**1**||**chainDir**==**2**||**chainDir**==**3**){**

row**=**row**-**1**;**

**}**

**if(**chainDir**==**5**||**chainDir**==**6**||**chainDir**==**7**){**

row**=**row**+**1**;**

**}**

//getting col from chainDir

**if(**chainDir**==**0**||**chainDir**==**1**||**chainDir**==**7**){**

col**=**col**+**1**;**

**}**

**if(**chainDir**==**3**||**chainDir**==**4**||**chainDir**==**5**){**

col**=**col**-**1**;**

**}**

nextP**->**setXY**(**row**,**col**);**//pass by refrence

**return** chainDir**;**

**}**

void getChainCode**(**ofstream**&** output1**,** ofstream**&** output2**){**

bool labelfound**=false;**

**for(**int row **=** 1**;** row**<=** numRows**;** row**++){**

**for(**int col **=** 1**;** col**<=** numCols**;** col**++){**

**if(**imgAry**[**row**][**col**]>**0**){**

labelfound**=true;**

output1**<<**row**<<**" "**<<**col**<<**" "**<<**imgAry**[**row**][**col**]<<**" "**;**

output2**<<**row**<<**" "**<<**col**<<**" "**<<**imgAry**[**row**][**col**]<<**endl**;**

Point startP **(**row**,** col**);**

Point currentP **(**row**,** col**);**

Point nextP**(**0**,**0**);**

int lastQ **=** 4**;**

int count **=**0**;**

**do{**

int nextQ **=** **(**lastQ**)%**8**;**

int PchainDir **=** findNextP**(**currentP**,** nextQ**,** **&**nextP**);**

// nextP will be determined inside the findNextP method.

output1**<<**PchainDir**<<**" "**;**

output2**<<**PchainDir**<<**" "**;**

count**++;**

**if(**count**>=**15**){**

output2**<<**endl**;**

count**=**0**;**

**}**

lastQ **=** PchainDir**;**

currentP**.**setXY**(** nextP**.**getX**(),** nextP**.**getY**()** **);** // nextP was determined inside the findNextP method.

**}while((**currentP**.**equal**(**startP**))==false);** //repeat until currentP == startP

**}**

**if(**labelfound**)break;**

**}**

**if(**labelfound**)break;**

**}**

**}**

void prettyPrint**(**int **\*\***ary**,** ofstream**&** output**,** bool zborder**,** bool iheader**){**

**if(**iheader**==true)**output**<<**endl**<<**numRows**<<**" "**<<**numCols**<<**" "**<<**minVal**<<**" "**<<**maxVal**;** //prints image params

output**<<**endl**;**

**if(**zborder**==false){**

**for(**int row **=** 1**;** row**<=**numRows**;** row**++){** //print out from [1][1] w/o border

**for(**int col **=** 1**;** col**<=**numCols**;** col**++){**

int pixel\_val **=** ary**[**row**][**col**];**

**if(**pixel\_val **>** 0**)** output **<<** pixel\_val **<<**" "**;** //if not 0. output 1 space,

**else** output **<<** " "**;** // if 0. output 2 space

**}**

output**<<**endl**;**

**}**

**}**

**if(**zborder**==true){**

**for(**int row **=** 0**;** row**<=**numRows**+**1**;** row**++){** //print out from [0][0] with border

**for(**int col **=** 0**;** col**<=**numCols**+**1**;** col**++){**

int pixel\_val**=** ary**[**row**][**col**];**

**if(**pixel\_val **>** 0**)** output **<<** pixel\_val **<<**" "**;** //if not 0. output 1 space,

**else** output **<<** " "**;** // if 0. output 2 space

**}**

output**<<**endl**;**

**}**

**}**

**}**

int main**(**int argc**,** char **\***argv**[])** **{**

ifstream input**(**argv**[**1**]);**

ofstream output1**(**argv**[**2**]);**

ofstream output2**(**argv**[**3**]);**

loadImage**(**input**);** //loads parameters andf fills array

zeroFramed**(**imgAry**,** numRows**,** numCols**);**

output1**<<**numRows**<<**" "**<<**numCols**<<**" "**<<**minVal**<<**" "**<<**maxVal**<<**endl**;**

output2**<<**numRows**<<**" "**<<**numCols**<<**" "**<<**minVal**<<**" "**<<**maxVal**<<**endl**;**

getChainCode**(**output1**,** output2**);**

input**.**close**();**

output1**.**close**();**

output2**.**close**();**

**for** **(**int i **=** 0**;** i **<** numRows**+**2**;** **++**i **)delete** **[]** imgAry**[**i**];**

**delete** **[]** imgAry**;**

//delete []neighborCoord;

**}**

**OUTPUT FILES**

**Data1output1.txt**

15 19 0 1

1 1 1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4

**Data1output2.txt**

15 19 0 1

1 1 1

6 6 6 6 6 6 6 6 6 6 6 6 6 6 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 2 2 2 2 2 2 2 2 2 2 2 2 2

2 4 4 4 4 4 4 4 4 4 4 4 4 4 4

4 4 4 4

**Data2output1.txt**

20 19 0 1

1 10 1 5 5 5 5 5 5 5 5 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3 3

**Data2output2.txt**

20 19 0 1

1 10 1

5 5 5 5 5 5 5 5 5 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 3 3 3

3 3 3 3 3 3

**Data3output1.txt**

17 17 0 1

1 9 1 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3

**Data3output2.txt**

17 17 0 1

1 9 1

5 5 5 5 5 5 5 5 7 7 7 7 7 7 7

7 1 1 1 1 1 1 1 1 3 3 3 3 3 3

3 3

**Data4output1.txt**

17 17 0 1

1 9 1 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3

**Data4output2.txt**

17 17 0 1

1 9 1

5 5 5 5 5 6 6 6 6 6 6 7 7 7 7

7 1 1 1 1 1 2 2 2 2 2 2 3 3 3

3 3