CSC 381 Computer Vision ( C++ )

Project 7: ChainCode & Recibstruct

Han Wen Loh

Soft copy: 04/25/2019

Hard copy: 04/30/2019

Algorithm Steps:

step 0: inFile 🡨 open input file

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate imgAry size of numRows by numCols

dynamically allocate ZeroFramedAry size of numRows+2 by numCo+2

Step 1: chainCodeFileName 🡨 argv[1] + “\_chainCode.txt”

Step 2: chainCodeFile 🡨 open (chainCodeFileName)

Step 3: loadImage (inFile, ZeroFramedAry) // begins at ZeroFramedAry (1,1)

Step 4: getChainCode (ZeroFramedAry, chainCodeFile)

Step 5: close (chainCodeFile)

Step 6: re-open (chainCodeFile)

Step 7: deCompressFileName 🡨 argv[1] + “\_chainCodeDecompressed.txt”

Step 8: deCompressFile 🡨 open (deCompressFileName)

Step 9: reconstructObject (chainCodeFile, deCompressFile, imgAry)

Step 10: close all files

#include <iostream>

#include <fstream>

using namespace std;

class Image {

public:

int numRows, numCols, minVal, maxVal, label, numBoundaryPts;

int\*\* zeroFramedAry;

int\*\* imgAry;

Image(ifstream& imageFile) {

imageFile >> numRows >> numCols >> minVal >> maxVal;

label = numBoundaryPts = 0;

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

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

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

zeroFramedAry[i] = new int[numCols + 2]();

imgAry[i] = new int[numCols + 2]();

}

for (int i = 1; i <= numRows; ++i)

for (int j = 1; j <= numCols; ++j)

imageFile >> zeroFramedAry[i][j];

}

void setZero() {

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

for (int j = 0; j < numCols + 2; ++j)

zeroFramedAry[i][j] = 0;

}

void printBoundary(ofstream& boundaryFile, int\*\* ary) {

for (int i = 1; i <= numRows; ++i) {

for (int j = 1; j <= numCols; ++j)

boundaryFile << ary[i][j] << " ";

boundaryFile << endl;

}

}

};

class ChainCode {

public:

class Point {

public:

int row, col;

Point() {

row = col = -1;

}

Point(int r, int c) {

row = r;

col = c;

}

Point& operator= (const Point& p) {

this->row = p.row;

this->col = p.col;

return \*this;

}

Point& operator+= (const Point& p) {

row += p.row;

col += p.col;

return \*this;

}

bool operator== (const Point& p) {

return row == p.row && col == p.col;

}

};

Point startP;

Point currentP;

Point nextP;

Point neightBorCoord[8] = {Point(0,1), Point(-1,1), Point(-1,0),

Point(-1, -1), Point(0, -1), Point(1, -1),

Point(1, 0), Point(+1, 1)};

int lastQ;

int dirTable[8] = { 7, 1, 1, 3, 3, 5, 5, 7 };

int chainDir;

ChainCode() {

lastQ = -1;

chainDir = -1;

}

void getChainCode(Image\* img, ofstream& outF) {

outF << img->numRows << " " << img->numCols << " "

<< img->minVal << " " << img->maxVal << endl;

int i = 1;

int j = 1;

while (startP.row < 0 && i < img->numRows) {

++i;

while (startP.col < 0 && j < img->numCols) {

++j;

if (img->zeroFramedAry[i][j] > 0) {

startP.row = i;

startP.col = j;

currentP.row = i;

currentP.col = j;

lastQ = 5;

img->label = img->zeroFramedAry[i][j];

outF << i-1 << " " << j-1 << " " << img->label << " ";

}

}

}

if (i == img->numRows || j == img->numCols) {

cout << "error: image file.\n";

exit(1);

}

do{

chainDir = findNextP(currentP, lastQ, img);

++img->numBoundaryPts;

outF << chainDir << " ";

currentP = nextP; //nextP is determined inside findNextP

//(chainDir -1 +8) mod 8, c++ cannot handle negative modular

lastQ = dirTable[(chainDir + 7) % 8];

} while (!(currentP == startP));

cout << "The total boundary points is " << img->numBoundaryPts << endl;

}

int findNextP(Point& currentP, int lastQ, Image\* img) {

do{

nextP = currentP;

nextP += neightBorCoord[lastQ];

lastQ = (lastQ + 1) % 8;

} while (img->zeroFramedAry[nextP.row][nextP.col] != img->label);

return (lastQ + 7) % 8;

}

void reconstructObject(ifstream& chainCodeFile, ofstream& deCompFile, ofstream& boundaryFile, Image\* img) {

int numRows = 0, numCols = 0, minVal = 0, maxVal = 0, label = 0, boundaryLabel;

chainCodeFile >> numRows >> numCols >> minVal >> maxVal >> startP.row >> startP.col >> label;

deCompFile << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;

boundaryLabel = label + 1;

constructBoundary(chainCodeFile, boundaryFile, img, boundaryLabel); //distinguish boundary

fillInterior(deCompFile, img, label, boundaryLabel);

}

void constructBoundary(ifstream& chainCodeFile, ofstream& boundaryFile, Image\* img, int boundaryLabel) {

currentP = startP;

currentP.row++;

currentP.col++;

img->setZero();

while(!chainCodeFile.eof()) {

img->imgAry[currentP.row][currentP.col] = boundaryLabel;

chainCodeFile >> chainDir;

currentP += neightBorCoord[chainDir];

}

img->printBoundary(boundaryFile, img->imgAry);

}

void fillInterior(ofstream& deCompFile, Image\* img, int label, int boundaryLabel) {

int countBoundary, startC = -1;

for (int r = 1; r <= img->numRows; ++r) {

countBoundary = -1;

for (int c = 1; c <= img->numCols; ++c) {

if (img->imgAry[r][c] >= boundaryLabel) {

if (img->imgAry[r][c - 1] < boundaryLabel)

startC = c;

//if it's end of boundary and not sharp or flat corner

if (img->imgAry[r][c + 1] < boundaryLabel && !isSharpCorner(r, startC, c, img, boundaryLabel)) {

++countBoundary;

++c;

}

}

//only fill interior when left boudary found (countBoundary is even)

if (!(countBoundary & 1) && img->imgAry[r][c] < boundaryLabel)

++img->imgAry[r][c];

}

}

for (int r = 1; r <= img->numRows; ++r)

for (int c = 1; c <= img->numCols; ++c) {

if (img->imgAry[r][c] >= label)

img->imgAry[r][c] = label;

else

img->imgAry[r][c] = 0;

}

img->printBoundary(deCompFile, img->imgAry);

}

/\*

\* check if the boundary is sharp or flat corner

\* \* \*\*...\*

\* \* \* \* \*

\* or

\* \* \* \* \*

\* \* \*\*...\*

\*/

bool isSharpCorner(int r,int startC, int c, Image\* img, int boundaryLabel) {

if ((img->imgAry[r - 1][startC - 1] >= boundaryLabel && img->imgAry[r - 1][startC] < boundaryLabel && img->imgAry[r - 1][c] < boundaryLabel && img->imgAry[r - 1][c + 1] >= boundaryLabel) ||

(img->imgAry[r + 1][startC - 1] >= boundaryLabel && img->imgAry[r + 1][startC] < boundaryLabel && img->imgAry[r + 1][c] < boundaryLabel && img->imgAry[r + 1][c + 1] >= boundaryLabel))

return true;

return false;

}

};

int main(int args, char\*\* argv) {

ifstream imageFile, iChainCodeFile;

ofstream oChainCodeFile, deCompFile, boundaryFile;

string imageFileName = string(argv[1]).substr(0, string(argv[1]).size() - 4);

string chainCodeFileName = imageFileName + "\_chainCode.txt";

string deCompFileName = imageFileName + "\_chainCodeDecompressed.txt";

string boundaryFileName = imageFileName + "\_Boundary.txt";

oChainCodeFile.open(chainCodeFileName);

deCompFile.open(deCompFileName);

imageFile.open(argv[1]);

if (imageFile.fail()) {

cout << "ERROR: cannot find \"" << argv[1] << "\"\n";

exit(1);

}

Image\* image = new Image(imageFile);

imageFile.close();

ChainCode\* chainCode= new ChainCode();

chainCode->getChainCode(image, oChainCodeFile);

oChainCodeFile.close();

iChainCodeFile.open(chainCodeFileName);

boundaryFile.open(boundaryFileName);

chainCode->reconstructObject(iChainCodeFile, deCompFile, boundaryFile, image);

iChainCodeFile.close();

deCompFile.close();

boundaryFile.close();

}

ChainCode\_image.txt

20 40 0 1

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

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0

0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ChainCode\_image\_chainCode.txt

20 40 0 1

1 7 1 5 5 5 7 0 7 5 4 4 6 6 6 6 6 6 6 7 7 7 7 7 1 1 1 0 0 7 0 7 7 1 1 0 0 7 7 1 1 2 1 7 7 7 1 1 1 1 0 1 1 4 4 4 4 4 3 1 0 1 1 0 0 3 3 3 3 3 3 3 5 5 5 5 5 5 5 0 0 7 7 0 7 5 4 4 4 4 4 7 7 5 5 4 4 3 2 2 2 3 3 2 3 3 5 5 5 5 5 3 3 2 2 2 2 2 2 2 3 3

ChainCode\_image\_Boundary.txt

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ChainCode\_image\_chainCodeDecompressed.txt

20 40 0 1

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

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0

0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0