CSC 381-34: Proj6 (C++)

Swrajit Paul

Due date: Nov. 1, 2018

```
III. In Main
*******************************
 step 1: labelFile <-- open label file, argv[1]
        propFile <-- open property file, argv[2]
               output image header to outFile1
               output image header to outFile2 // per text line
               imageAry <-- dynamically allocated
               lordImage(imageAry)
               zeroFramed (imageAry)
               CCAry <-- dynamically allocated

 step 2: CC <-- get the next connected component from the property file
        // CC a connectCC class object, therefore, all its properties are stored in the object CC
        // by the class's constructor.

 step 3: CClable <-- get the label of CC

 Step 4: clearCC () // zero out the old cclable for next cc

               loadCC (CClable, CCAry) // Extract the pixels with CClabel from imageAry to CCAry.

 step 5: getChainCode(CC, CCAry)

 step 6: repeat step 2 to step 5 until all connected components are processed.

 step 7: close all files

*******************************
IV. getChainCode(CC, CCAry)
*******************************

 step 1: minRow, minCol, maxRow, maxCol <-- get from CC's property

 step 2: scan the CCAry from L to R &  T to B within the bounding box

 step 3: if CCAry(iRow, jCol) > 0
                                    output iRow, jCol, CCAry(iRow, jCol) to outFile1 // see format specs above
                                    output iRow, jCol, CCAry(iRow, jCol) to outFile2 // see format specs above
                                    startP <-- (iRow, jCol)
                                    currentP <-- (iRow, jCol)
                                    lastQ <-- 4

step 4: nextQ <-- mod(lastQ+1, 8) // chain code for the outter boundry.
step 5: PchainDir <-- findNextP(currentP, nextQ, nextP) // nextP is assigned in the findNextP method.
            currentP <-- flip the label of currentP from positive to negitive

step 6: output PchainDir to outFile1 // no spaces.
            output PchainDir to outFile2 // with the readable format as given above

step 7: lastQ <-- nextDirTable[PchianDir]
            currentP <-- nextP

step 8: repeat step 4 to step 7 until currentP == startP
```

****

III. findNextP(currentP, nextQ, nextP)

****

```
step 1: loadNeighborCoord(currentP)

step 2: chainDir <-- scan currentP's 8 neighbors within nighborCoord[] array from nextQ direction (mod 8)
          until a none zero neighbor with the same label as currentCC is found, chainDir is the index of
          neighborCoord[] which with the same label as currentP

step 3: nextP <-- nighborCoord[chainDir]

step 4: returns chainDir
```

SOURCE CODE

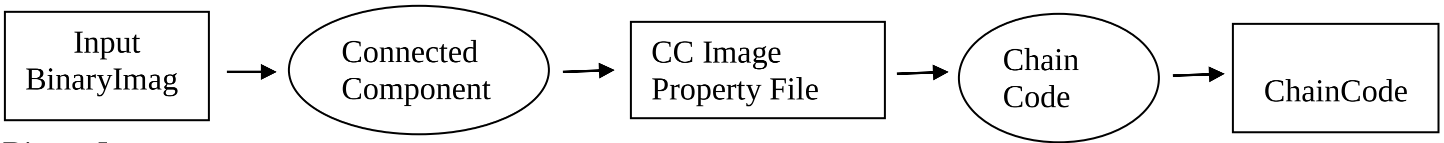
```
// Author: Swrajit Paul

#include <iostream>
#include <fstream>

using namespace std;

ifstream inFile;
ifstream inFiletwo;
ofstream outFile;
ofstream outFiletwo;

class image {

    public:

        int numRows;
        int numCols;
        int minVal;
        int maxVal;

        int** imageAry;
        int** CCAry;

     image() {

            inFile >> numRows;
            inFile >> numCols;
            inFile >> minVal;
            inFile >> maxVal;

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

            outFiletwo << numRows;
            outFiletwo << " ";
            outFiletwo << numCols;
            outFiletwo << " ";
            outFiletwo << minVal;
            outFiletwo << " ";
            outFiletwo << maxVal;
            outFiletwo << endl;
            imageAry = new int*[numRows+2];
            for(int i = 0; i < numRows+2; i++){
                imageAry[i] = new int[numCols+2]; }// set up the array with proper rows and cols
            for(int i = 0; i < numRows+2; i++) {
                for(int j = 0; j < numCols+2; j++) {
                    imageAry[i][j] = 0; } }// initialize the array

            CCAry = new int*[numRows+2];
            for(int i = 0; i < numRows+2; i++){
                CCAry[i] = new int[numCols+2];
            }// set up the array with proper rows and cols
            for(int i = 0; i < numRows+2; i++) {
                for(int j = 0; j < numCols+2; j++) {
                    CCAry[i][j] = 0; } }// initialize the array
     }

     void loadImage(int** FramedAry) {
            // reads line by line from the input into zeroFramedAry
            for(int i = 1; i < numRows+1; i++) {
                for(int j = 1; j < numCols+1; j++) {
                    inFile >> FramedAry[i][j]; } }

        }

        void zeroFramed(int** FramedAry) {
            for(int j = 0; j < numCols+2; j++) {
                FramedAry[0][j] = 0;
                FramedAry[numRows+1][j] = 0; }

            for(int j = 0; j < numRows+2; j++) {
                FramedAry[j][0] = 0;
                FramedAry[j][numCols+1] = 0; }
        }

};


class connectCC {

    public:

        int label;
        int numPixels;
        int minRow;
        int minCol;
        int maxRow;
        int maxCol;

     connectCC(int lab, int np, int minR, int minC, int maxR, int maxC, image im) {
         label = lab;
            numPixels = np;
            minRow = minR;
            minCol = minC;
            maxRow = maxR;
            maxCol = maxC;
            clearCC(im.CCAry, im.numRows+2, im.numCols+2);
            loadCC(label, im.CCAry, im);
      }

      void clearCC(int** CCAry, int row, int col){
        for(int i = 0; i < row; i++) {
                for(int j = 0; j < col; j++) {
                    CCAry[i][j] = 0;
                }
            }
        }

        void loadCC(int ccLable, int** CCAry, image im){
            for(int i = 0; i < im.numRows+2; i++) {
                for(int j = 0; j < im.numCols+2; j++) {
                    if(im.imageAry[i][j] == ccLable)
                        CCAry[i][j] = im.imageAry[i][j];
                }
            }
        }

};

class chainCode{

    struct Point {
        int row;
        int col;
    };

    public:
        Point neighborCoord[8];
        Point startP;
        Point currentP;
        Point nextP;
        int lastQ;
        int nextQ;
        int nextDirTable[8] = {6, 0, 0, 2, 2, 4, 4, 6};;
        int nextDir;
        int PchainDir;
        int** imgAry;

        chainCode(connectCC CC, image im){
            imgAry = im.CCAry;

            getChainCode(im.CCAry, CC.minRow, CC.minCol, CC.maxRow, CC.maxCol);
        }

        void getChainCode(int** ary, int minR, int minC, int maxR, int maxC){
```

```
bool flag = false;

for(int i= minR; i <= maxR; i++) {

    for(int j = minC; j <= maxC; j++){

        if(ary[i][j] > 0){

            flag = true;

            // set starting point
        startP.row = i;
        startP.col = j;

        // set current point
        currentP.row = i;
        currentP.col = j;

        lastQ = 4;
        if(ary[i][j]!=1)
            outFiletwo << "###" <<endl;
        outFile << ary[i][j] << " " << i-1 << " " << j-1 << " ";
        outFiletwo << ary[i][j] << " " << i-1 << " " << j-1 << " ";
        int count = 0;
            while (true){

                nextQ = (lastQ+1) % 8;

                PchainDir = findNextP(currentP, nextQ, nextP);
                count++;
            outFile << PchainDir << " ";
            if(count == 20){
                outFiletwo << endl;
                count = 0;
                }
                outFiletwo << PchainDir << " ";
                nextQ = PchainDir -1;

                if(nextQ < 0){
                nextQ += 8;
                }

                lastQ = nextDirTable[nextQ];
                currentP.row = nextP.row;
            currentP.col = nextP.col;

            if((currentP.row == startP.row) && (currentP.col == startP.col)){
                    break;
                }
            }
            if(flag == true){
                break;
                }
            }

    }

    if(flag == true){
            break;
        }
    }
    outFile << endl;
    outFiletwo << endl;
}
void loadNeighborsCoord(Point current) {
    neighborCoord[0].row = current.row;
    neighborCoord[0].col = current.col+1;

    neighborCoord[1].row = current.row-1;
    neighborCoord[1].col = current.col+1;

    neighborCoord[2].row = current.row-1;
    neighborCoord[2].col = current.col;

    neighborCoord[3].row = current.row-1;
    neighborCoord[3].col = current.col-1;

    neighborCoord[4].row = current.row;
    neighborCoord[4].col = current.col-1;

    neighborCoord[5].row = current.row+1;
    neighborCoord[5].col = current.col-1;

    neighborCoord[6].row = current.row+1;
    neighborCoord[6].col = current.col;

    neighborCoord[7].row = current.row+1;
    neighborCoord[7].col = current.col+1;
}

int findNextP(Point currentP, int nextQ, Point p){
    int chainDir;
    loadNeighborsCoord(currentP);
    chainDir = getChainDir(currentP, nextQ);
    nextP.row = neighborCoord[chainDir].row;
    nextP.col = neighborCoord[chainDir].col;
    return chainDir;
}

int getChainDir(Point current, int nextQ){
    int chainDir;
    int counter = nextQ;
    while(true){
        int i = neighborCoord[counter].row;
```

```
                int j = neighborCoord[counter].col;

                if( imgAry[i][j]> 0){
                    chainDir = counter;
                    break;
                }
                counter++;
                counter = counter % 8;
            }
            return chainDir;
        }
        void prettyPrint() {

        }
};


int main(int argc, char *argv[]){
    inFile.open(argv[1]);

    inFiletwo.open(argv[2]);

    outFile.open(argv[3]);

    outFiletwo.open(argv[4]);

    image img;
    img.loadImage(img.imageAry);
    img.zeroFramed(img.imageAry);

    int temp;
    inFiletwo >> temp;
    inFiletwo >> temp;
    inFiletwo >> temp;
    inFiletwo >> temp;
    // skipping lines

    inFiletwo >> temp;

    for(int i=0; i < temp; i++){

        int label, numPixels, minRow, minCol, maxRow, maxCol;
        inFiletwo >> label;
        inFiletwo >> numPixels;
        inFiletwo >> minRow;
        inFiletwo >> minCol;
        inFiletwo >> maxRow;
        inFiletwo >> maxCol;

        connectCC ConCC(label, numPixels, minRow, minCol, maxRow, maxCol, img);

        chainCode chainC(ConCC, img);
```

```
 }

        inFile.close();
        inFiletwo.close();
        outFile.close();
        outFiletwo.close();
        return 0;
}
```

INPUT



Binary Image

```
26 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 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 1 1 1 1 1 1 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 1 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 1 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 1 1 1 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 0 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 1 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 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 1 1 1 1 1 1 0 0 0 0 1 1 1 1 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 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 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 0 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 0 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 0 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 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 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 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 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 0 0 0 0 0 0 0 0 0 0 0 0
```

Connected Components

```
26 40 0 3
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  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  2  2  2  2  2  2  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  2  2  2  2  2  2  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  2  2  2  2  2  2  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  2  2  2  2  2  2  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  2  2  2  2  2  2  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  2  2  2  2  2  2  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  1  1  0  0  0
0  0  0  0  2  2  2  2  2  2  0  0  0  0  3  3  3  3  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  2  2  2  2  2  2  0  0  0  0  3  3  3  3  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  2  2  2  2  2  2  0  0  0  0  3  3  3  3  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  2  2  2  2  2  2  0  3  3  3  3  3  3  3  3  3  3  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0
0  0  0  0  2  2  2  2  2  2  0  3  3  3  3  3  3  3  3  3  3  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  0  0  2  2  2  2  2  2  0  3  3  3  3  3  3  3  3  3  3  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0
0  0  0  0  2  2  2  2  2  2  0  3  3  3  3  3  3  3  3  3  3  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  3  3  3  3  3  3  3  3  3  3  0  0  0  0  0  0  1  1  1  1  1  1  1  0  0  0  0  0  0
0  0  0  0  0  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  0  0  0  1  1  1  1  1  0  0  0  0  0  0
0  0  0  0  0  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  0  0  0  0  1  1  1  0  0  0  0  0  0  0
0  0  0  0  0  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  0  0  0  0  0  1  0  0  0  0  0  0  0  0
0  0  0  0  0  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  0  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  3  3  3  3  3  3  3  3  3  3  3  3  0  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  3  3  3  3  3  3  3  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  3  3  3  3  3  3  3  3  3  3  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  3  3  3  3  3  3  3  3  3  3  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  3  3  3  3  3  3  3  3  3  3  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  0  0  0  0  0  0  0  0  0  0  0  0
```

Property File

```
26 40 0 3
3
1
128
1 23
18 37
2
78
2 4
14 9
3
208
8 5
24 24
```

OUTPUT

Outfile One

```
26 40 0 3
1 1 30 5 5 5 5 5 5 5 0 0 0 0 0 7 6 6 5 4 4 4 4 4 7 7 0 7 7 7 7 1 1 1 1 0 1 1 4 4 4 4 4 3 2 2 1 0 0 0 0 0 3 3 3 3 3 3 3
2 2 4 6 6 6 6 6 6 6 6 6 6 6 6 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4
3 8 14 6 6 5 4 4 6 6 6 6 5 4 4 4 4 4 6 6 6 6 6 0 0 0 0 0 7 6 6 0 0 0 0 0 0 0 0 0 2 2 2 1 0 0 0 2 2 2 2 4 4 4 3 2 2 2 2
4 4 3 2 2 4 4 4
```

Outfile Two

```
26 40 0 3
1 1 30 5 5 5 5 5 5 5 0 0 0 0 0 7 6 6 5 4 4 4
4 4 7 7 0 7 7 7 7 1 1 1 1 0 1 1 4 4 4 4
4 3 2 2 1 0 0 0 0 0 3 3 3 3 3 3 3
###
2 2 4 6 6 6 6 6 6 6 6 6 6 6 6 0 0 0 0 0 2 2
2 2 2 2 2 2 2 2 2 2 4 4 4 4 4
###
3 8 14 6 6 5 4 4 6 6 6 6 5 4 4 4 4 4 6 6 6 6
6 0 0 0 0 0 7 6 6 0 0 0 0 0 0 0 0 0 2 2
2 1 0 0 0 2 2 2 2 4 4 4 3 2 2 2 2 4 4 3
2 2 4 4 4
```