Project 5.1: 8 Connected Components (C++)

CSC 381 Rafael Carmilema

Due Date: Oct / 20 /2016

III. Algorithms  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
step 0: read the image header  
           dynamically allocate zeroFramedAry  
  
step 1: load the input image onto zeroFramedAry  
  
  
step 2: 4ConnectCC\_Pass1 // similar to the 8-connected algorithm taught in class, except fewer neighbors  
           - prettyprint the result of pass1// with proper caption  
           - print EQAry        // with index up to newLable with proper caption  
  
step 3: 4ConnectCC\_Pass2  
           - prettyprint the result of pass2// with caption  
           - print EQAry // with index up to newLable with caption  
  
step 4: manageEQAry // the method was briefly taught in class  
           - print EQAry // with index up to newLable with caption  
  
step 5: 4ConnectCC\_Pass3  // algorithm taught in class  
                                        // In the pass3, you will use the EQAry to relable the components  
                                        // as well as compute the property of each c.c.  
  
           - prettyprint the result of pass3 of the connected c.c.  // with caption  
         - print the propertis of the connected c.c. // with proper caption

#include <iostream>

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

struct property{

//int label;

// int numbpixels;

//int boundingBox;

};

class ConnectedCC{

private:

int row;

int col;

int min;

int max;

int newMin;

int newMax;

int newLabel = 0;

int nonzero;

int minlabel;

int\*\* zeroFramedAry;

int neighborAry[4];

int\* EQAry;

int minVal=999;

int maxVal=0;

public:

ConnectedCC(int r, int c, int mi, int ma){

row =r;

col= c;

min= mi;

max= ma;

// allocate zeroframed 2d array

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

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

zeroFramedAry[i]= new int [c+2];

}

//ini

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

for(int j=0;j <c+2;j++){

zeroFramedAry[i][j]=0;

}

}

EQAry = new int[(row\*col)/4];

for(int i=0; i<(row\*col)/4; i++){

EQAry[i]=i;

}

}

void zeroFramed(){

}

void loadImage(int value, int r, int c){

zeroFramedAry[r+1][c+1]=value;

}

void loadNeighborsPass1(int a, int b){

//up left

neighborAry[0]=zeroFramedAry[a-1][b-1];

//up middle

neighborAry[1]=zeroFramedAry[a-1][b];

//up right

neighborAry[2]=zeroFramedAry[a-1][b+1];

//middle left

neighborAry[3]=zeroFramedAry[a][b-1];

}

void updateEQAry(int m){

for(int i=0; i<4;i++){

if(neighborAry[i]!=0){

EQAry[neighborAry[i]]=m;

}

}

}

bool Case1(){

for(int i = 0; i < 4; i++){

if(neighborAry[i]!=0){

return false;

}

}

newLabel++;

return true;

}

bool Case2(){

int maximum=findMaximum();

int minimum=findMinimum(maximum);

if(maximum==minimum){

nonzero=minimum;

return true;

}

return false;

}

bool Case3(){

int maximum=findMaximum();

int minimum=findMinimum(maximum);

if(minimum!=maximum){

minlabel=minimum;

return true;

}

return false;

}

// pass 1

void ConnectCC\_Pass1(){

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

for(int j=1; j<=col;j++){

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

//load neighbors of P(i,j)

loadNeighborsPass1(i,j);

//case1

if(Case1()){

zeroFramedAry[i][j]= newLabel;

//cout<<newLabel;

//break;

}

else if(Case2()){

zeroFramedAry[i][j]=nonzero;

zeroFramedAry[i][j];

//cout<<newLabel;

// break;

}

else if(Case3()){

zeroFramedAry[i][j]=minlabel;

updateEQAry(minlabel);

//break;

}

}

}// end of inner for lopp

}// end of outer for loop

}

void loadNeighborsPass2(int a, int b){

//right neighbor

neighborAry[0]=zeroFramedAry[a][b+1];

//bottom left

neighborAry[1]=zeroFramedAry[a+1][b-1];

//bottom middle

neighborAry[2]=zeroFramedAry[a+1][b];

//bottom right

neighborAry[3]=zeroFramedAry[a+1][b+1];

}

int findMinimum(int m){

int mini= m;

for(int i=0; i<4;i++){

if(mini>neighborAry[i]&&neighborAry[i]!=0){

//cout<<neighborAry[i]" ";

mini=neighborAry[i];

}

}

return mini;

}

int findMaximum(){

int maxi= 0;

for(int i=0;i<4;i++){

if(maxi<neighborAry[i]){

maxi=neighborAry[i];

}

}

return maxi;

}

void ConnectCC\_Pass2(){

for(int i = row ; i >= 1 ; i--){

for(int j = col; j >= 1; j--){

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

//cout<<"test 2";

loadNeighborsPass2(i,j);

int maximum=findMaximum();

int minimum=findMinimum(maximum);

if(zeroFramedAry[i][j]<minimum){

minimum=zeroFramedAry[i][j];

}

//case 3

if(maximum!=minimum){

//cout<<"min "<<minimum;

//cout<<"maximum "<<maximum;

zeroFramedAry[i][j]=minimum;

updateEQAry(minimum);

}

}

}

}

}

void manageEQAry(){

for(int i = newLabel; i >0;i--){

int swap = EQAry[i];

int index = i;

while(swap!=index){

index = swap;

swap = EQAry[index];

}

EQAry[i]=swap;

}

int count=1;

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

if(EQAry[i]!=count){

if(EQAry[i]>count){

EQAry[i]=count;

for(int j=i; j<=newLabel;j++){

if(EQAry[j]==i){

EQAry[j]=count;

}

}

count++;

}

}

else{

count++;

}

}

}

void ConnectCC\_Pass3(){

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

for(int j = 1; j <=col; j++){

zeroFramedAry[i][j] = EQAry[zeroFramedAry[i][j]];

}

}

}

void printEQ(ofstream& outfile1){

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

outfile1<<i<<" ";

cout<<i<<" ";

}

cout<<endl;

outfile1<<endl;

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

outfile1<<EQAry[i]<<" ";

cout<<EQAry[i]<<" ";

}

outfile1<<endl;

cout<<endl;

}

int findMinValAndMaxVal(){

maxVal=0;

minVal=999;

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

for(int j=1; j<=col;j++){

if(zeroFramedAry[i][j]>maxVal){

maxVal=zeroFramedAry[i][j];

}

if(zeroFramedAry[i][j]<minVal){

minVal=zeroFramedAry[i][j];

}

}

}

}

void prettyPrint(ofstream& outfile1){

findMinValAndMaxVal();

outfile1<<row<<" "<<col<<" "<<minVal<<" "<<maxVal<<endl;

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

for(int j=1; j<=col;j++){

cout<<zeroFramedAry[i][j]<<" ";

outfile1<<zeroFramedAry[i][j]<<" ";

}

outfile1<<endl;

cout<<endl;

}

}

void prettyPrints(ofstream& outfile1){

findMinValAndMaxVal();

outfile1<<row<<" "<<col<<" "<<minVal<<" "<<maxVal<<endl;

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

for(int j=1; j<=col;j++){

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

outfile1<<zeroFramedAry[i][j];

}

else{

outfile1<<" ";

}

}

outfile1<<endl;

cout<<endl;

}

}

void PrintZero(ofstream& outfile1){

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

for(int j=0; j<col+2;j++){

//if(zeroFramedAry[i][j]>0){

outfile1<<zeroFramedAry[i][j];

cout<<zeroFramedAry[i][j];

}

outfile1<<endl;

cout<<endl;

//outfile1<<endl;

}

}

void propertyFile(ofstream& outfile2){

int counter [newLabel+1]={};

int minrow=999;

int mincol=999;

int maxrow=0;

int maxcol=0;

int labelcounter=0;

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

counter[EQAry[i]]++;

}

for(int k=1;k<=newLabel;k++){

if(counter[k]!=0){

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

for(int j=1; j<=col;j++){

if(zeroFramedAry[i][j]==k){

labelcounter++;

if(minrow>i){

minrow=i;

}

if(mincol>j){

mincol=j;

}

if(maxrow<i){

maxrow=i;

}

if(maxcol<j){

maxcol=j;

}

}

}

}

outfile2<<k<<" "<<minrow-1<<" "<<mincol-1<<" "<<maxrow-1<<" "<<maxcol-1<<" "<<labelcounter<<endl;

cout<<"Label "<<k<<endl;

cout<<"minrow "<<minrow-1<<endl;

cout<<"mincol "<<mincol-1<<endl;

cout<<"maxrow "<<maxrow-1<<endl;

cout<<"maxcol "<<maxcol-1<<endl;

cout<<"num labels "<< labelcounter<<endl;

cout<<endl;

minrow=999;

mincol=999;

maxrow=0;

maxcol=0;

labelcounter=0;

}

}

}

};

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

if ( argc <4 ) // argc should be 2 for correct execution

// We print argv[0] assuming it is the program name

cout<<"usage: "<< argv[0] <<" <filename>\n";

else {

// We assume argv[1] is a filename to open

ifstream the\_file ( argv[1] );

// Always check to see if file opening succeeded

if ( !the\_file.is\_open() )

cout<<"Could not open file\n";

else {

int count =0;

int string;

int row;

int col;

int min;

int max;

int data;

int\*\* inputArray;

ofstream outfile1;

ofstream outfile2;

ofstream outfile3;

outfile1.open(argv[2]);

//ofstream bin;

outfile2.open(argv[3]);

outfile3.open(argv[4]);

the\_file>>data;

row=data;

the\_file>>data;

col=data;

the\_file>>data;

min=data;

the\_file>>data;

max=data;

ConnectedCC test(row,col,min,max);

inputArray= new int\*[row];

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

inputArray[i]= new int[col];

}

for(int r = 0; r<row;r++){

for(int c= 0; c<col; c++){

inputArray[r][c]=0;

}

}

for(int r= 0; r<row;r++){

for(int c= 0; c<col;c++){

while(the\_file>>string){

inputArray[r][c]=string;

test.loadImage(string,r ,c);

//cout<<r<<" "<< c<<" "<<string<<endl;

break;

}

}

// cout<<endl;

}

outfile3<<"Pass 1"<<endl;

test.ConnectCC\_Pass1();

test.prettyPrints(outfile3);

outfile3<<"EQ Array"<<endl;

test.printEQ(outfile3);

outfile3<<"Pass 2"<<endl;

test.ConnectCC\_Pass2();

test.prettyPrints(outfile3);

outfile3<<"Before manageEQArray"<<endl;

test.printEQ(outfile3);

outfile3<<"After manageEQArray"<<endl;

test.manageEQAry();

test.printEQ(outfile3);

//test.manageEQAry();

outfile3<<"Pass 3"<<endl;

test.ConnectCC\_Pass3();

test.prettyPrints(outfile3);

outfile3<<"EQ Array"<<endl;

test.printEQ(outfile3);

//test.ConnectCC\_Pass3();

test.prettyPrint(outfile1);

test.propertyFile(outfile2);

the\_file.close();

outfile1.close();

outfile2.close();

outfile3.close();

}//else

}

return 0;

}

Out1:

Pass 1

38 31 0 8

1

111

11111

1111111

22 111111111

22 11

2222 11

22222 11 3

22222 11 3

222222 333

22222 444333

2222 4443333

22222 4433333

222 333

222 3

22222 3

222 3

222

22222222

55

555 666

55 6666

77555 66666

7555555555555

55555

8888888

8888888

8888888

88888

888

888

8

EQ Array

0 1 2 3 4 5 6 7 8

0 1 2 3 3 5 5 5 8

Pass 2

38 31 0 8

1

111

11111

1111111

22 111111111

22 11

2222 11

22222 11 3

22222 11 3

222222 333

22222 434333

2222 3333333

22222 4433333

222 333

222 3

22222 3

222 3

222

22222222

55

555 656

55 5555

55555 65656

7555555555555

55555

8888888

8888888

8888888

88888

888

888

8

Before manageEQArray

0 1 2 3 4 5 6 7 8

0 1 2 3 3 5 5 5 8

After manageEQArray

0 1 2 3 4 5 6 7 8

0 1 2 3 3 4 4 4 5

Pass 3

38 31 0 5

1

111

11111

1111111

22 111111111

22 11

2222 11

22222 11 3

22222 11 3

222222 333

22222 333333

2222 3333333

22222 3333333

222 333

222 3

22222 3

222 3

222

22222222

44

444 444

44 4444

44444 44444

4444444444444

44444

5555555

5555555

5555555

55555

555

555

5

EQ Array

0 1 2 3 4 5 6 7 8

0 1 2 3 3 4 4 4 5

Out2:

1 1 11 9 19 33

2 5 7 19 21 63

3 8 15 17 22 31

4 21 9 26 21 42

5 30 12 36 18 33

Out3:

38 31 0 5

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

0 0 0 0 0 0 0 2 2 0 0 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 2 2 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 1 1 0 0 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 1 1 0 0 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 0 0 0 0 0 3 3 3 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 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 2 2 2 2 2 0 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 2 2 2 0 0 0 0 0 3 3 3 0 0 0 0 0 0 0 0 0 0

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

0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 4 4 4 0 0 0 4 4 4 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0 4 4 4 4 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 4 4 4 4 4 0 0 0 4 4 4 4 4 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0

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

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

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

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

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

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

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

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