Project 3 Connected Compound (C++)

CSC 381 Rafael Carmilema

Due Date: Sep / 13 /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; // edit in future

};

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[3];

int\* EQAry;

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

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

//left

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

}

void updateEQAry(int m){

for(int i=0; i<3;i++){

if(neighborAry[i]!=0){

EQAry[neighborAry[i]]=m;

}

}

}

bool Case1(){

if(neighborAry[0]==0&&neighborAry[1]==0){

newLabel++;

return true;

}

return false;

}

bool Case2(){

if((neighborAry[0]!=0&&neighborAry[1]!=0)&&(neighborAry[0]==neighborAry[1])){

nonzero=neighborAry[1];

return true;

}

else if(neighborAry[0]==0&&neighborAry[1]!=0){

nonzero=neighborAry[1];

return true;

}

else if(neighborAry[0]!=0&&neighborAry[1]==0){

nonzero=neighborAry[0];

return true;

}

return false;

}

bool Case3(){

if(neighborAry[0]!=0&&neighborAry[1]!=0 &&neighborAry[0]!=neighborAry[1]){

minlabel=neighborAry[0];

if(minlabel>neighborAry[1]){

minlabel=neighborAry[1];

return true;

}

else

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];

//down neighbor

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

// itself

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

}

int findMinimum(int m){

int mini= m;

for(int i=0; i<3;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<3;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);

//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;

}

void prettyPrint(ofstream& outfile1){

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

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

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

cout<<zeroFramedAry[i][j];

outfile1<<zeroFramedAry[i][j];

}

else{

cout<<" ";

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];

// }

//else{

// outfile1<<" ";

// }

// outfile1<<tempAry[i][j]<<" ";

}

outfile1<<endl;

cout<<endl;

//outfile1<<endl;

}

}

};

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\*\* inputArray;

ofstream outfile1;

//ofstream outfile2;

outfile1.open(argv[2]);

//ofstream bin;

//outfile2.open(argv[3]);

//ofstream outfile2;

//outfile2.open(argv[3]);

//ofstream outfile3;

//outfile3.open(argv[4]);

while(the\_file>>string){

count++;

if(count==1){

row=string;

cout<<row<<endl;

}

else if(count==2){

col=string;

cout<<col<<endl;

}

else if(count == 3){

min=string;

cout<<min<<endl;

}

else if(count ==4){

max=string;

cout<<max<<endl;

break;

}

else{

break;

}

}

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;

}

//test.PrintZero(outfile1);

outfile1<<"Pass 1"<<endl;

test.ConnectCC\_Pass1();

//test.PrintZero(outfile1);

test.prettyPrint(outfile1);

outfile1<<"EQ Array"<<endl;

test.printEQ(outfile1);

outfile1<<"Pass 2"<<endl;

test.ConnectCC\_Pass2();

//test.PrintZero(outfile1);

test.prettyPrint(outfile1);

outfile1<<"Before manageEQArray"<<endl;

test.printEQ(outfile1);

outfile1<<"After manageEQArray"<<endl;

test.manageEQAry();

test.printEQ(outfile1);

//test.manageEQAry();

outfile1<<"Pass 3"<<endl;

test.ConnectCC\_Pass3();

cout<<endl;

//test.PrintZero(outfile1);

test.prettyPrint(outfile1);

cout<<endl;

outfile1<<"EQ Array"<<endl;

test.printEQ(outfile1);

the\_file.close();

outfile1.close();

//outfile2.close();

// outfile3.close();

}//else

}

return 0;

}

Pass 1

11

2111

33 111

3 11

433 5111

664333331111

76643333311111

766433333111111

8766433333111111

9876643333311111111

109876643333311111 111

1110987664333331111111111

121110987664333331111 111111

13121110987664333331111 1111

141312111098766433333111 111111

1515141312111098766433333111 161111111

1515141312111098766433333111 171611111111

15141312111098766433333111 17161111111

1413121110987664333331111 16111111

131211109876 333331111 11111

12111098766 3333311111 111

1211109876663333311111 1

11109876663333311111

109876663333311111

98766633333111111

876663333311111

666

6

666 1818

6666 19181818

6666666

66666

666

6

EQ Array

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

0 6 1 1 3 3 6 6 7 8 9 10 11 12 13 14 1 16 6 6

Pass 2

11

1111

13 111

1 11

111 1111

111111111111

11111111111111

111111111111111

1111111111111111

1111111111111111111

11111111111111111 111

11111111111111111111111

111111111111111111 111111

1111111111111111111 1111

1111111111111111111 111111

111111111111111111111 11111111

111111111111111111111 1111111111

11111111111111111111 111111111

11111111111111111111 1111111

11111111 111111111 11111

11111111 1111111111 111

1111111111111111111 1

111111111111111111

11111111111111111

11111111111111111

111111111111111

666

6

666 66

6666 66618

6666666

66666

666

6

Before manageEQArray

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 6

After manageEQArray

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

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

Pass 3

11

1111

11 111

1 11

111 1111

111111111111

11111111111111

111111111111111

1111111111111111

1111111111111111111

11111111111111111 111

11111111111111111111111

111111111111111111 111111

1111111111111111111 1111

1111111111111111111 111111

111111111111111111111 11111111

111111111111111111111 1111111111

11111111111111111111 111111111

11111111111111111111 1111111

11111111 111111111 11111

11111111 1111111111 111

1111111111111111111 1

111111111111111111

11111111111111111

11111111111111111

111111111111111

111

1

111 11

1111 1111

1111111

11111

111

1

EQ Array

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1.0