CV Project 4 Connected Components Java

Jonathan Mathew

Project Due 03/19/23

Algorithm Steps

step 0: inFile-->open the input file from argv [1]

Connectness-->argv [2] option-->argv [3] RFprettyPrintFile, labelFile, propertyFile, deBugFile-->open from argv []

numRows, numCols, minVal, maxVal-->read from inFile zeroFramedAry-->dynamically allocate. newLabel --> 0

step 1: zero2D (zeroFramedAry) step 2: loadImage (inFile, zeroFramedAry)

step 3: if option == ‘y’ or ‘Y’

conversion (zeroFramedAry)

step 4: if connectness == 4

connected4 (zeroFramedAry, newLabel, EQAry, RFprettyPrintFile, deBugFile)

step 5: if connectness == 8

connected4 (zeroFramedAry, newLabel, EQAry, RFprettyPrintFile, deBugFile)

step 6: labelFile-->output numRows, numCols, newMin, newMax to labelFile

step 7: printImg (zeroFramedAry, labelFile) // Output the result of pass3 inside of zeroFramedAry

step 8: printCCproperty (propertyFile) // print cc properties to propertyFile step 9: drawBoxes (zeroFramedAry, CCproperty, trueNumCC) // draw on zeroFramed image. step 10: imgReformat (zeroFramedAry, RFprettyPrintFile)

step 11: print trueNumCC to RFprettyPrintFile with proper caption

step 12: close all files

CONN 8 Data 1.txt

PrettyPrint.txt

Calendar

Description automatically generated

Text

Description automatically generated

A picture containing text

Description automatically generated

Label.txt

A picture containing text

Description automatically generated

Property.txt

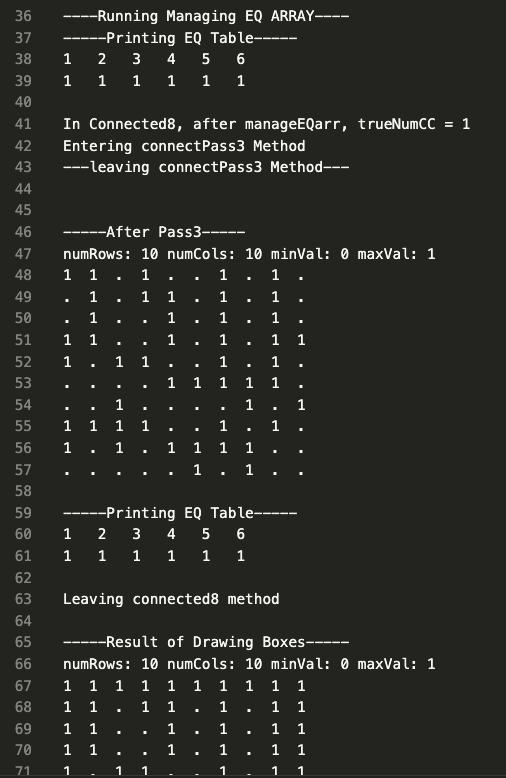
Text

Description automatically generated

Debug.txt

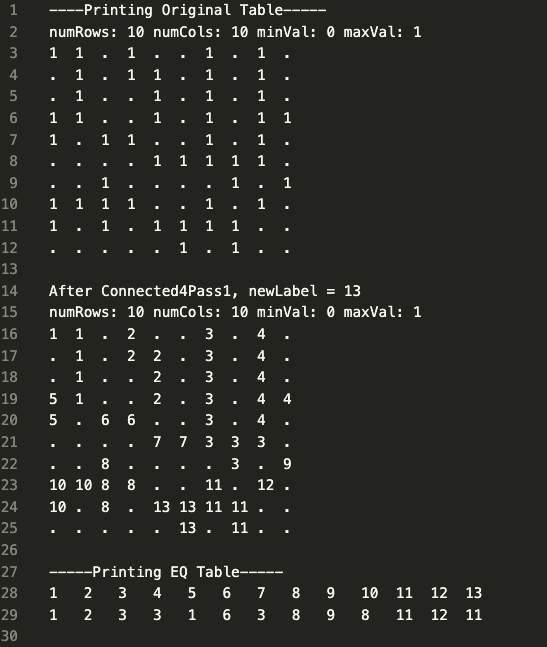
A picture containing text

Description automatically generated



CONN 4 Data 1.txt

PrettyPrint.txt



Calendar

Description automatically generated

Calendar

Description automatically generated with medium confidence

Label.txt

Calendar

Description automatically generated

Properties.txt

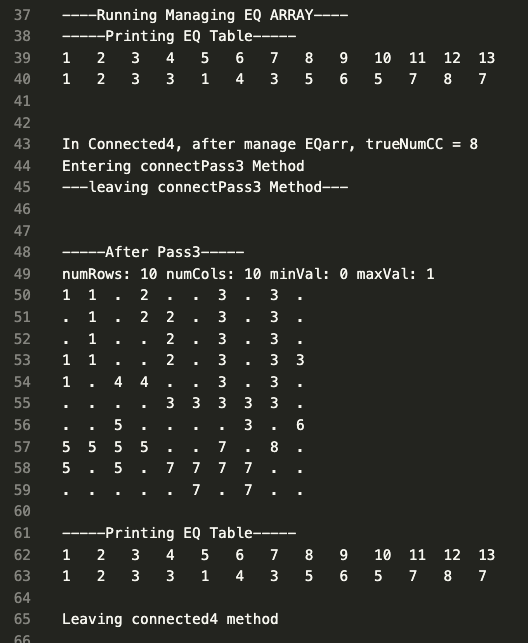
A screenshot of a computer

Description automatically generated with low confidence

Debug.txt

Calendar

Description automatically generated



CONN 8 Data 2.txt

PrettyPrint.txt

A picture containing graphical user interface

Description automatically generated

A picture containing background pattern

Description automatically generated

A picture containing calendar

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

A picture containing graphical user interface

Description automatically generated

Label.txt

A picture containing background pattern

Description automatically generated

Propert.txt

Calendar

Description automatically generated with low confidence

debug.txt

Graphical user interface

Description automatically generated

Graphical user interface

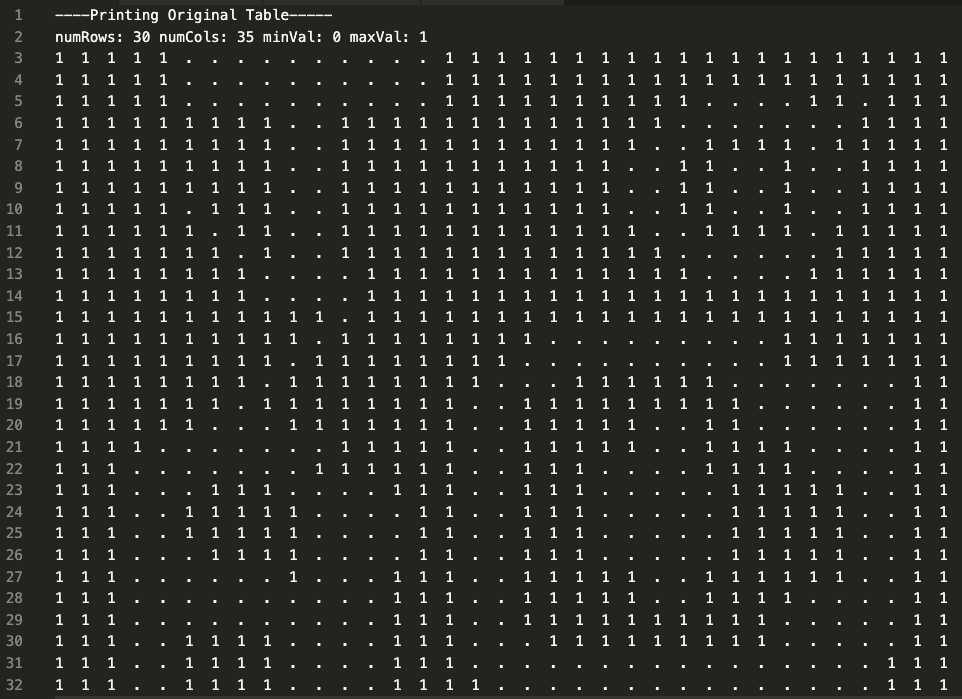
Description automatically generated with medium confidence

Text

Description automatically generated

CONN 4 Data2.txt CONVERSION

PrettyPrint.txt



Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Background pattern

Description automatically generated

Background pattern

Description automatically generated

Label.txt

Background pattern

Description automatically generated with medium confidence

Property.txt

A picture containing table

Description automatically generated

Debug.txt



Graphical user interface

Description automatically generated with low confidence

Main.cpp

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <cstring>

#include <algorithm>

using *namespace* std;

*struct* Property {

*int* label;

*int* numPixels;

*int* minR;

*int* minC;

*int* maxR;

*int* maxC;

Property() {

label = 0;

numPixels = 0;

minR = 0;

minC = 0;

maxR = 0;

maxC = 0;

}

Property(*int* *label*, *int* *numPixels*, *int* *minR*, *int* *minC*, *int* *maxR*, *int* *maxC*) {

this->label = *label*;

this->numPixels = *numPixels*;

this->minR = *minR*;

this->minC = *minC*;

this->maxR = *maxR*;

this->maxC = *maxC*;

}

};

*class* ccLabel {

*public:*

*int* numRows;

*int* numCols;

*int* minVal;

*int* maxVal;

*int* newLabel;

*int* trueNumCC;

*int* newMin;

*int* newMax;

*int*\*\* zeroFramedAry;

*int* NonZeroNeighborAry[5];

*int*\* EQAry;

*char* option;

Property\* CCproperty;

ccLabel(*int* *rows*, *int* *cols*, *int* *min*, *int* *max*, *char* *conver*) {

numRows = *rows*;

numCols = *cols*;

minVal = *min*;

maxVal = *max*;

newLabel = 0;

newMin = 0;

newMax = 0;

option = *conver*;

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

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

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

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

zeroFramedAry[i][j] = 0;

}

}

EQAry = new *int*[(*int*)(numRows \* numCols) / 4];

for (*int* i = 0; i < (numRows \* numCols) / 4; i++) {

EQAry[i] = i;

// cout<<EQAry[i];

}

//Not Sure what to do here come back to it silly goose

}

*void* zero2D(*int*\*\* *arr*, *int* *r*, *int* *c*){

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

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

*arr*[i][j] = 0;

}

}

}

*void* negative1D(*int* \**arr*, *int* *size*){

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

*arr*[i] = -1;

}

}

*void* loadImage(ifstream \**infile*, ofstream \**outfile*){

string line;

*int* val;

*int* r=0;

while (getline(\**infile*, line)){

*int* c=1;

istringstream set(line);

while (set >> val){

// cout << val << " ";

zeroFramedAry[r][c] = val;

c++;

}

// cout << endl;

r++;

// // cout <<endl;

}

}

*void* connectPass3(ofstream \**out*){

\**out* << "Entering connectPass3 Method"<<endl;

//step1

for (*int* i = 1; i <= trueNumCC; i++)

{

CCproperty[i].label = i;

CCproperty[i].numPixels = 0;

CCproperty[i].minR = numRows;

CCproperty[i].maxR = 0;

CCproperty[i].minC = numCols;

CCproperty[i].maxC = 0;

}

*int* r=1;

//step 4

while(r<numRows+1){

*int* c=1;

while (c<numCols+1){

*int* pix = zeroFramedAry[r][c];

//step 3

if(pix > 0){

zeroFramedAry[r][c] = EQAry[pix];

*int* k = zeroFramedAry[r][c];

CCproperty[k].numPixels++;

if(r<CCproperty[k].minR)

CCproperty[k].minR = r;

if(r>CCproperty[k].maxR)

CCproperty[k].maxR = r;

if(c<CCproperty[k].minC)

CCproperty[k].minC = c;

if(c>CCproperty[k].maxC)

CCproperty[k].maxC = c;

}

c++;

}

r++;

}

\**out* << "---leaving connectPass3 Method---"<<endl<<endl;

}

*void* conversion(){

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

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

zeroFramedAry[i][j] = (zeroFramedAry[i][j]+1)%2;

}

}

}

*void* printProperty(ofstream \**property*){

//Image heade

\**property*<<numRows<<" "<<numCols<<" "<<minVal<<" "<<maxVal<<endl;

//number of CCs

// \*property<<"trueNumCC: ";

\**property*<<trueNumCC<<endl;

for (*int* i = 1; i <= trueNumCC; i++){

// \*property<<"Label: ";

\**property*<<CCproperty[i].label<<endl;

// \*property<<"numPix: ";

\**property*<<CCproperty[i].numPixels<<endl;

// \*property<<"minRow and MinCol ";

\**property*<<CCproperty[i].minR-1<<" "<<CCproperty[i].minC-1<<endl;

// \*property<<"maxRow and maxCol ";

\**property*<<CCproperty[i].maxR-1<<" "<<CCproperty[i].maxC-1<<endl;

}

}

*void* imgReformat(ofstream \**output*){

\**output* << "numRows: " << numRows << " numCols: " << numCols << " minVal: " << minVal << " maxVal: " << maxVal<<endl;

*int* r =1;

//step 12

//step 4

while (r<numRows+1){

//step 10

*int* c =1;

while (c<numCols+1){

//step 5

if(zeroFramedAry[r][c]==0)

\**output*<< ".";

else{

\**output*<< zeroFramedAry[r][c];

}

//step 8

if(zeroFramedAry[r][c] < 10){

\**output* << " ";

}

else{

\**output* << " ";

}

//step 9

c++;

}

\**output* << endl;

//step 11

r++;

}

}

*void* printArr(ofstream \**out*){

\**out*<<"\_\_\_\_PRINTING ZERO FRAMED ARR\_\_\_\_"<<endl;

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

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

\**out*<<zeroFramedAry[i][j]<< " ";

}

\**out*<<endl;

}

\**out*<<endl<<endl<<endl;

}

*void* drawBoxes(){

//step 1

*int* index = 1;

// cout<<"trueNumCC is " << trueNumCC<<endl;

while (index <=trueNumCC){

// cout<<"index is " << index<<endl;

//step 2

*int* minRow = CCproperty[index].minR,

minCol = CCproperty[index].minC,

maxRow = CCproperty[index].maxR,

maxCol = CCproperty[index].maxC,

label = CCproperty[index].label;

// cout<<"\tCCproperty["<<index<<"].minR+1: "<< CCproperty[index].minR <<endl;

// cout<<"\tCCproperty["<<index<<"].minC+1: "<< CCproperty[index].minC <<endl;

// cout<<"\tCCproperty["<<index<<"].maxR+1: "<< CCproperty[index].maxR <<endl;

// cout<<"\tCCproperty["<<index<<"].maxC+1 "<< CCproperty[index].maxC <<endl;

// cout<<"\tCCproperty[numPixels] "<< CCproperty[index].numPixels <<endl;

// cout<<"\tCCproperty["<<index<<"].label "<< CCproperty[index].label<<endl;

//step 3

for (*int* r= minRow; r < maxRow+1; r++){

zeroFramedAry[r][minCol] = label;

zeroFramedAry[r][maxCol] = label;

}

for (*int* c = minCol; c < maxCol+1; c++){

zeroFramedAry[minRow][c] = label;

zeroFramedAry[maxRow][c] = label;

}

//step 4

index++;

}

}

*void* printImg(ofstream \**out*){

\**out*<<"-----Printing image-----"<<endl;

\**out*<<numRows<<" " << numCols <<" " << minVal <<" " << maxVal<<endl;

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

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

if(zeroFramedAry[i][j]==0)

\**out* << ".";

else

\**out* << zeroFramedAry[i][j];

if(zeroFramedAry[i][j] < 10){

\**out* << " ";

}

else{

\**out* << " ";

}

}

\**out*<<endl;

}

\**out*<<endl<<endl<<endl;

}

*void* printEQarr(ofstream \**out*){

\**out*<<endl<<"-----Printing EQ Table-----"<<endl;

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

if(i<10)

\**out*<< i << " ";

else

\**out*<< i << " ";

}

\**out*<<endl;

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

if(EQAry[i]<10)

\**out*<<EQAry[i] << " ";

else

\**out*<<EQAry[i] << " ";

}

\**out*<<endl<<endl;

}

*void* printEQAll(){

cout<<endl<<"-----Printing EQ all Table-----"<<endl;

for (*int* i = 0; i < (*int*)(numRows \* numCols) / 4; i++){

if(i<10)

cout<< i << " ";

else

cout<< i << " ";

}

cout<<endl;

for (*int* i = 0; i < (*int*)(numRows \* numCols) / 4; i++){

if(EQAry[i]<10)

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

else

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

}

cout<<endl<<endl;

}

*bool* case2(*int* *a*, *int* *b*, *int* *cl*, *int* *d*, *int* *r*, *int* *c*){

// if(r==5 && c==4){

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

// cout<<endl<<endl<<endl;

// }

*int* arr[] = {*a*,*cl*,*b*,*d*};

*int* last\_non\_0 = 0,

zeroCount = 0,

sum =0;

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

if(arr[i]!=0){

last\_non\_0 = arr[i];

}

if(arr[i]==0){

zeroCount++;

}

sum+=arr[i];

}

// cout<<a<<b<<cl<<d<<sum<<sum/(4-zeroCount)<<last\_non\_0<<endl;

// cout<< (sum/(4-zeroCount)==last\_non\_0)<<"Reee"<<endl;

// if(r==5 && c==4){

// cout<<"sum is: " << sum<<endl;

// cout<<"zeroCOunt is: " << zeroCount<<endl;

// cout<<"last nonzero is: " << last\_non\_0<<endl;

// cout<<"4-zeroCount:" << 4-zeroCount<<endl;

// cout<<"sum/4-zeroCount:" << sum/(4-zeroCount)<<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b <<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

// }

return (*double*) sum/(4-zeroCount)==last\_non\_0;

// return false;

}

*int* case3(*int* *a*, *int* *b*, *int* *cl*, *int* *d*){

*int* arr[] = {*a*,*b*,*cl*,*d*};

*int* min = 99999;

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

{

if(arr[i]<min && arr[i]!=0){

min = arr[i];

}

}

// cout<<"min is: " << min<<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

return min;

}

*bool* case2Pass2(*int* *e*,*int* *f*,*int* *g*,*int* *h*,*int* *pix*,*int* *r*,*int* *c*){

*int* arr[] = {*e*,*f*,*g*,*h*,*pix*};

*int* last\_non\_0 = 0,

zeroCount = 0,

sum =0;

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

if(arr[i]!=0){

last\_non\_0 = arr[i];

}

if(arr[i]==0){

zeroCount++;

}

sum+=arr[i];

}

return (*double*) sum/(5-zeroCount)==last\_non\_0;

}

*int* case3Pass2(*int* *e*,*int* *f*,*int* *g*,*int* *h*,*int* *pix*,*int* *r*,*int* *c*){

*int* arr[] = {*e*,*f*,*g*,*h*,*pix*};

*int* min = 99999;

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

{

if(arr[i]<min && arr[i]!=0){

min = arr[i];

}

}

// cout<<"min is: " << min<<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

return min;

}

*void* connect8pass1(){

//step 0

newLabel = 0;

*int* pix, i=1;

//step 4

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

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

//step 1

pix = zeroFramedAry[r][c];

*int* a = zeroFramedAry[r-1][c-1];

*int* b = zeroFramedAry[r-1][c];

*int* cl = zeroFramedAry[r-1][c+1];

*int* d = zeroFramedAry[r][c-1];

// step 2

if(pix>0){

//case 1

if( a==0 && b==0 && cl==0 && d==0){

newLabel++;

pix = newLabel;

zeroFramedAry[r][c] = pix;

//step 3

// cout<<"case 1"<<endl<< zeroFramedAry[r][c] <<" Row and Col: (" <<r<<","<<c<<")"<<"====>"<<endl;

EQAry[i] = pix;

i++;

// cout<<zeroFramedAry[r][c] <<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

}

//case 2 this is not right comebcak to it

else if(case2(a,b,cl,d, r, c)==true){

// cout<<"case2"<<endl;

// if(r==5 && c==4){

// cout<<"case 2"<<endl<< zeroFramedAry[r][c] <<" Row and Col: (" <<r<<","<<c<<")"<<"====>"<<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

// cout<<"\t\tMIN is: "<<case3(a,b,cl,d)<<endl;

// }

pix = case3(a,b,cl,d);

zeroFramedAry[r][c] = pix;

// if(r==5 && c==4)

// cout<<zeroFramedAry[r][c] <<endl;

}

//case 3

else{

// cout<<"case 3"<<endl<< zeroFramedAry[r][c] <<" Row and Col: (" <<r<<","<<c<<")"<<"====>"<<endl;

pix = case3(a,b,cl,d);

zeroFramedAry[r][c] = pix;

// cout<<zeroFramedAry[r][c] <<endl;

// cout<<"\ta:" << a<<endl;

// cout<<"\tb:" << b<<endl;

// cout<<"\tc:" << cl<<endl;

// cout<<"\td:" << d<<endl;

// //step 3

// // EQAry[i] = pix;

// cout<<"I: "<< i<<endl;

EQAry[a] = pix;

EQAry[b] = pix;

EQAry[cl] = pix;

EQAry[d] = pix;

}

}

}

}

}

*void* connect8pass2(){

*int* pix, i=1;

EQAry[0] = 0;

for (*int* r = numRows; r > 0; r--){

for (*int* c = numCols; c > 0; c--){

//step 1

pix = zeroFramedAry[r][c];

*int* e = zeroFramedAry[r][c+1];

*int* f = zeroFramedAry[r+1][c-1];

*int* g = zeroFramedAry[r+1][c];

*int* h = zeroFramedAry[r+1][c+1];

// step 2

if(pix>0){

//case 1

if(e==0 && f==0 && g==0 && h==0){

//do nothing

}

//case 2 this is not right comebcak to it

else if(case2Pass2(e,f,g,h,pix,r,c)){

// zeroFramedAry[r][c] = pix;

}

//case 3

else{

*int* minL = case3Pass2(e,f,g,h,pix,r,c);

if(pix > minL){

EQAry[pix] = minL;

pix = minL;

zeroFramedAry[r][c] = pix;

}

}

}

//step 3

pix = EQAry[pix];

zeroFramedAry[r][c] = pix;

}

}

}

*void* connect4pass1(){

//step 0

newLabel = 0;

*int* pix;

*int* i=1;

//step 4

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

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

//step 1

pix = zeroFramedAry[r][c];

*int* b = zeroFramedAry[r-1][c];

*int* d = zeroFramedAry[r][c-1];

// step 2

if(pix>0){

//case 1

if( b==0 && d==0){

newLabel++;

pix = newLabel;

zeroFramedAry[r][c] = pix;

//step 3

// i++;

EQAry[i] = pix;

// cout<<" case 1 "<<endl;

// cout<<"EQ[i] = " << EQAry[i];

// cout<<" i = " << i<<endl;

i++;

// cout<<"case 1"<<endl;

}

//case 2 this is not right comebcak to it

else if(b==d || b==0 || d==0){

pix = max(b,d);

zeroFramedAry[r][c] = max(b,d);

// cout<<" case 2 "<<endl;

// cout<<"EQ[i] = " << EQAry[i];

// cout<<" i = " << i<<endl;

}

//case 3

else if(b!=d){

pix = min(d,b);

zeroFramedAry[r][c] = pix;

//step 3

// i++;

EQAry[i] = pix;

EQAry[max(b,d)] = pix;

// cout<<" case 3 "<<endl;

// cout<<"EQ[i] = " << EQAry[i];

// cout<<" i = " << i<<endl;

// i++;

}

//step 3

// cout<<"ROW: "<<r<< " COL: "<<c<<" "<<zeroFramedAry[r][c]<<endl;

}

}

}

// cout<<endl<<"I AM DONE" << endl;

}

*void* connect4pass2(){

//step 0

*int* pix, i=1;

//step 4

for (*int* r = numRows; r > 0; r--){

for (*int* c = numCols; c > 0; c--){

//step 1

pix = zeroFramedAry[r][c];

*int* e = zeroFramedAry[r][c+1];

*int* g = zeroFramedAry[r+1][c];

// step 2

if(pix>0){

//case 1

if(e==0 && g==0){

//do nothing

}

//case 2 this is not right comebcak to it

else if( (e==g && g==pix) || (e==pix && g==0) || (g==pix && e==0) || (e==0 || g==0)){

zeroFramedAry[r][c] = pix;

}

//case 3

else{

*int* minL = min(e,g);

if(pix > minL){

EQAry[pix] = minL;

pix = minL;

zeroFramedAry[r][c] = pix;

}

}

}

//step 3

pix = EQAry[pix];

zeroFramedAry[r][c] = pix;

}

}

}

*int* manageEq(){

// step 0

*int* readLabel = 0;

// step 1

*int* index = 1;

// step 4

while (index <= newLabel){

// step 2

if(index != EQAry[index]){

EQAry[index] = EQAry[EQAry[index]];

}else{

readLabel++;

EQAry[index] = readLabel;

}

//step 3

index++;

}

return readLabel;

}

*void* connected4(ofstream \**pretty*, ofstream \**debug*){

\**debug* << "------entering connected 4 method------"<<endl;

\**pretty*<<"----Printing Original Table-----"<<endl;

imgReformat(*pretty*);

//step 1

connect4pass1();

\**debug* << endl<<"After Connected4Pass1, newLabel = " << newLabel<<endl;

\**pretty* << endl<< "After Connected4Pass1, newLabel = " << newLabel<<endl;

imgReformat(*pretty*);

printEQarr(*pretty*);

imgReformat(*debug*);

printEQarr(*debug*);

//step 2

connect4pass2();

\**debug* << "After Connected4Pass2, newLabel = " << newLabel<<endl;

\**pretty* << "After Connected4Pass2, newLabel = " << newLabel<<endl;

imgReformat(*pretty*);

printEQarr(*pretty*);

imgReformat(*debug*);

printEQarr(*debug*);

//step 3

\**debug* << "----Running Managing EQ ARRAY---- ";

\**pretty* << "----Running Managing EQ ARRAY---- ";

trueNumCC = manageEq();

printEQarr(*pretty*);

printEQarr(*debug*);

newMin = 0;

newMax = trueNumCC;

CCproperty = new Property[trueNumCC+1];

\**debug* <<endl<< "In Connected4, after manage EQarr, trueNumCC = " << trueNumCC <<endl;

\**pretty* <<endl<< "In Connected4, after manage EQarr, trueNumCC = " << trueNumCC <<endl<<endl;

//step 4

connectPass3(*debug*);

//step 5

\**debug* <<endl<< "-----After Pass3-----"<<endl;

\**pretty* <<endl<< "-----After Pass3-----"<<endl;

imgReformat(*pretty*);

imgReformat(*debug*);

//step 6

printEQarr(*pretty*);

printEQarr(*debug*);

//step 7

\**debug* << "Leaving connected4 method" <<endl;

}

*void* connected8(ofstream \**pretty*, ofstream \**debug*){

\**debug* << "------entering connected 8 method------"<<endl;

\**pretty*<<"----Printing Original Table-----"<<endl;

//step 1

connect8pass1();

\**debug* << "After Connected8Pass1, newLabel = " << newLabel<<endl;

\**pretty* << endl<< "After Connected4Pass1, newLabel = " << newLabel<<endl;

imgReformat(*pretty*);

printEQarr(*pretty*);

imgReformat(*debug*);

printEQarr(*debug*);

//step 2

connect8pass2();

\**debug* << "After Connected8Pass2, newLabel = " << newLabel<<endl;

\**pretty* << "After Connected4Pass2, newLabel = " << newLabel<<endl;

imgReformat(*pretty*);

printEQarr(*pretty*);

imgReformat(*debug*);

printEQarr(*debug*);

//step 3

\**debug* << "----Running Managing EQ ARRAY---- ";

\**pretty* << "----Running Managing EQ ARRAY---- ";

trueNumCC = manageEq();

printEQarr(*pretty*);

printEQarr(*debug*);

newMin = 0;

newMax = trueNumCC;

CCproperty = new Property[trueNumCC+1];

\**debug* << "In Connected8, after manageEQarr, trueNumCC = " << trueNumCC <<endl;

\**pretty* <<endl<< "In Connected4, after manage EQarr, trueNumCC = " << trueNumCC <<endl<<endl;

//step 4

connectPass3(*debug*);

//step 5

\**debug* <<endl<< "-----After Pass3-----"<<endl;

\**pretty* <<endl<< "-----After Pass3-----"<<endl;

imgReformat(*pretty*);

imgReformat(*debug*);

//step 6

printEQarr(*pretty*);

printEQarr(*debug*);

//step 7

\**debug* << "Leaving connected8 method" <<endl;

}

};

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

ifstream infile;

infile.open(*argv*[1]);

*int* connectness = stoi(*argv*[2]);

*char* option = \**argv*[3];

ofstream pretty, label, property, debug;

pretty.open(*argv*[4]); //pretty print

label.open(*argv*[5]); //label print

property.open(*argv*[6]); //property print

debug.open(*argv*[7]); //debug print

*int* numRows, numCols, minVal, maxVal, newLabel = 0;

infile >> numRows >> numCols>>minVal>>maxVal;

ccLabel \*proj = new ccLabel(numRows, numCols, minVal, maxVal ,option);

//step 2

proj->loadImage(&infile, &pretty);

//step 3

if(option == 'y' || option == 'Y'){

proj->conversion();

}

//step 4

if(connectness == 4){

proj->connected4(&pretty, &debug);

}

//step 5

if(connectness == 8){

proj->connected8(&pretty, &debug);

}

//step 6

label << "numRows: " << numRows<< " numCols: " << numCols<< " minVal: " << minVal << " maxVal: "<< maxVal<<endl;

//step 7

proj->printImg(&label);

//step 8

property << "CCProperties"<<endl;

pretty <<endl<<"Number of Conected Components: " << proj->trueNumCC<<endl<<endl;

//step 9

proj->drawBoxes();

debug<<endl<<"-----Result of Drawing Boxes-----"<<endl;

pretty<<endl<<"-----Result of Drawing Boxes-----"<<endl;

proj->imgReformat(&debug);

proj->printProperty(&property);

//step 10

proj->imgReformat(&pretty);

//step 11

proj->connect8pass1();

//step 12

pretty.close(); //pretty print

label.close(); //label print

property.close(); //property print

debug.close();

}