**CS381-37: Project 3 (JAVA)**

**Yida Tao**

**Due date: Feb. 22, 2018**

Algorithm Steps:

step 0: read the image header

dynamically allocate zeroFramedAry and all other arrays

step 1: load the input image onto zeroFramedAry

step 2: - 8ConnectCC\_Pass1 // as taught in class

- prettyprint the result of pass1// with proper caption

- print EQAry // with index up to newLable with proper caption

step 3: - 8ConnectCC\_Pass2 // as taught in class

- prettyprint the result of pass2// with caption

- print EQAry // with index up to newLable with caption

step 4: - manageEQAry // as taught in class.

- print EQAry // with index up to newLable with caption

step 5: - 8ConnectCC\_Pass3 // In the pass3, you will use the EQAry to relabel the components;

// keep track the newMin newMax

// as well as compute the property of each c.c.

// and store the c. c. properties

- prettyprint the result of pass3 of the connected c.c. // with caption

- Output the result of pass3 to outFile2 with updated image header

- print the propertis of the connected c.c. // with proper caption

**Source Code**

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**public** **class** EightCC {

**class** Property{

**int** numpx;

**int** minCol;

**int** maxCol;

**int** minRow;

**int** maxRow;

Property(){

numpx = 0;

minCol = 99;

maxCol = 0;

minRow = 99;

maxRow = 0;

}

}

**int** numRows;

**int** numCols;

**int** minVal;

**int** maxVal;

**int** newMin;

**int** newMax;

**int** newLabel;

**int**[][] zeroFramedAry;

**int**[] neighborAry;

**int**[] EQAry;

Property[] properties;

**public** EightCC(String input){

Scanner sc = **null**;

newMin = 1;

newMax = 0;

newLabel = 0;

neighborAry = **new** **int**[5];

**try** {

sc = **new** Scanner(**new** File(input));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

System.***out***.println("Cant find the file" + input);

}

**try** {

numRows = sc.nextInt();

numCols = sc.nextInt();

minVal = sc.nextInt();

maxVal = sc.nextInt();

//zeroFramed

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

EQAry = **new** **int**[numRows \* numCols / 2];

**for**(**int** i = 0; i < EQAry.length; i++){

EQAry[i] = i;

}

//loadimage

**int** counter = 0;

**int** r = 0;

**int** c = 0;

**while**(sc.hasNextInt()){

r = counter/numCols + 1;

c = counter%numCols + 1;

zeroFramedAry[r][c] = sc.nextInt();

counter++;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

sc.close();

}

}

**private** **void** loadNeighbors1(**int** i, **int** j){

neighborAry[0] = **this**.zeroFramedAry[i - 1][j - 1];

neighborAry[1] = **this**.zeroFramedAry[i - 1][j];

neighborAry[2] = **this**.zeroFramedAry[i - 1][j + 1];

neighborAry[3] = **this**.zeroFramedAry[i][j - 1];

}

**public** **void** eightCC\_Pass1(){

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

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

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

loadNeighbors1(i,j);

//case 1

**if**(neighborAry[0]==0 && neighborAry[1]==0 && neighborAry[2]==0 && neighborAry[3]==0){

newLabel ++;

zeroFramedAry[i][j] = newLabel;

}

//case 2 & 3

**else** {

**int** min = minNeighbor1();

**int** max = maxNeighbor1();

//case 2

**if**(min == max){

zeroFramedAry[i][j] = min;

}

//case 3

**else**{

zeroFramedAry[i][j] = min;

updateEQAry1(min);

}

}

}

}

}

}

**private** **int** minNeighbor1(){

//find non-zero min

**int** min = 99;

**for**(**int** i = 0; i < neighborAry.length - 1; i++){

**if**(neighborAry[i] < min && neighborAry[i] > 0) min = neighborAry[i];

}

**return** min;

}

**private** **int** maxNeighbor1(){

**int** max = neighborAry[0];

**for**(**int** i = 1; i < neighborAry.length - 1; i++){

**if**(neighborAry[i] > max) max = neighborAry[i];

}

**return** max;

}

**private** **void** updateEQAry1(**int** m){

**for**(**int** i = 0; i < neighborAry.length - 1; i++){

**if**(neighborAry[i] > m){

EQAry[neighborAry[i]] = m;

}

}

}

**private** **void** loadNeighbors2(**int** i, **int** j){

neighborAry[0] = **this**.zeroFramedAry[i][j]; //itself

neighborAry[1] = **this**.zeroFramedAry[i][j + 1];

neighborAry[2] = **this**.zeroFramedAry[i + 1][j - 1];

neighborAry[3] = **this**.zeroFramedAry[i + 1][j];

neighborAry[4] = **this**.zeroFramedAry[i + 1][j + 1];

}

**public** **void** eightCC\_Pass2(){

**for**(**int** i = numRows; i > 1; i--){

**for**(**int** j = numCols; j > 1; j--){

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

loadNeighbors2(i,j);

//case 1

**if**(neighborAry[1]==0 && neighborAry[2]==0 && neighborAry[3]==0 && neighborAry[4]==0){

**continue**;

}

//case 2 & 3

**else** {

**int** min = minNeighbor2();

**int** max = maxNeighbor2();

//case 2

**if**(min == max){

zeroFramedAry[i][j] = min;

}

//case 3

**else**{

zeroFramedAry[i][j] = min;

updateEQAry2(min);

}

}

}

}

}

}

**private** **int** minNeighbor2(){

//find non-zero min

**int** min = 99;

**for**(**int** i = 0; i < neighborAry.length; i++){

**if**(neighborAry[i] < min && neighborAry[i] > 0) min = neighborAry[i];

}

**return** min;

}

**private** **int** maxNeighbor2(){

**int** max = neighborAry[0];

**for**(**int** i = 1; i < neighborAry.length; i++){

**if**(neighborAry[i] > max) max = neighborAry[i];

}

**return** max;

}

**private** **void** updateEQAry2(**int** m){

**for**(**int** i = 0; i < neighborAry.length; i++){

**if**(neighborAry[i] > m){

EQAry[neighborAry[i]] = m;

}

}

}

**public** **void** manageEQAry(){

**int** count = 0;

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

**if**(EQAry[i] == i){

count ++;

EQAry[i] = count;

}

**else**{

EQAry[i] = EQAry[EQAry[i]];

}

}

newMax = count;

//initialize property table

properties = **new** Property[count + 1];

**for**(**int** i = 1; i < properties.length; i++){

properties[i] = **new** Property();

}

}

**public** **void** eightCC\_Pass3(){

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

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

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

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

properties[zeroFramedAry[i][j]].numpx++;

**if**(i > properties[zeroFramedAry[i][j]].maxRow){

properties[zeroFramedAry[i][j]].maxRow = i;

}

**if**(i < properties[zeroFramedAry[i][j]].minRow){

properties[zeroFramedAry[i][j]].minRow = i;

}

**if**(j > properties[zeroFramedAry[i][j]].maxCol){

properties[zeroFramedAry[i][j]].maxCol = j;

}

**if**(j < properties[zeroFramedAry[i][j]].minCol){

properties[zeroFramedAry[i][j]].minCol = j;

}

}

}

}

}

**public** **void** prettyPrint(){

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

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

**if**(zeroFramedAry[i][j] > 0 && zeroFramedAry[i][j] < 10){

System.***out***.print(zeroFramedAry[i][j] + " ");

}

**else** **if**(zeroFramedAry[i][j] >= 10){

System.***out***.print(zeroFramedAry[i][j] + " ");

}

**else**{

System.***out***.print(" ");

}

}

System.***out***.println();

}

System.***out***.println();

}

**public** **void** createImg(){

System.***out***.println(numRows + " " + numCols + " " + newMin + " " + newMax);

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

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

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

System.***out***.print(zeroFramedAry[i][j] + " ");

}

**else**{

System.***out***.print(zeroFramedAry[i][j] + " ");

}

}

System.***out***.println();

}

System.***out***.println();

}

**public** **void** printEQ(){

System.***out***.print("EQ Array: ");

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

System.***out***.print(EQAry[i] + " ");

}

System.***out***.println();

System.***out***.println();

}

**public** **void** printCCProperty(){

System.***out***.println(numRows + " " + numCols + " " + newMin + " " + newMax);

System.***out***.println(properties.length - 1);

**for**(**int** i = 1; i < properties.length; i++){

System.***out***.println("-----------------");

System.***out***.println(i);

System.***out***.println(properties[i].numpx);

System.***out***.println((properties[i].minRow-1) + " " + (properties[i].minCol-1));

System.***out***.println((properties[i].maxRow-1) + " " + (properties[i].maxCol-1));

}

}

}

**import** java.io.\*;

**public** **class** project3 {

**public** **static** **void** main(String[] args) {

FileOutputStream fos = **null**;

//step 0,1

EightCC ec = **new** EightCC(args[0]);

**try** {

fos = **new** FileOutputStream(args[1]);

System.*setOut*(**new** PrintStream(fos));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

//step 2

System.***out***.println("------- Pass 1 -------");

ec.eightCC\_Pass1();

ec.prettyPrint();

ec.printEQ();

//step 3

System.***out***.println("------- Pass 2 -------");

ec.eightCC\_Pass2();

ec.prettyPrint();

ec.printEQ();

//step 4

System.***out***.println("------- ManageEQ -------");

ec.manageEQAry();

ec.printEQ();

//step 5

System.***out***.println("------- Pass 3 -------");

ec.eightCC\_Pass3();

ec.prettyPrint();

**try** {

fos = **new** FileOutputStream(args[2]);

System.*setOut*(**new** PrintStream(fos));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

ec.createImg();

**try** {

fos = **new** FileOutputStream(args[3]);

System.*setOut*(**new** PrintStream(fos));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

ec.printCCProperty();

//done

**try** {

fos.close();

System.*setOut*(**new** PrintStream(**new** FileOutputStream(FileDescriptor.***out***)));

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("Done");

}

}

**Input**

20 25 0 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Output 1**

------- Pass 1 -------

1 1 2

3 3 4 1 1 5 5 5 2

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

3 3 3 3 3 1 1 6 8 5 5 7 2

3 3 3 1 1 6 8 5 5 5 9 7 7 2

3 3 3 3 1 1 6 8 5 5 5 7 7 7

3 3 3 3 1 1 6 8 5 5 5 7

3 3 3 1 1 6 8 5 5 5 7 10 11

3 3 3 1 1 6 8 5 5 5 7 10 11

3 3 3 1 1 6 8 5 5 5 5 7 7 7 11

3 5 11 11

3 3 12 12 12 12 12 5 5 5 13 14 11

3 3 12 12 12 12 12 5 5 5 13 13 11 15

3 3 12 12 12 12 12 5 5 5 11 15

3 3 12 12 12 16 17 5 5 5 18 11 11 11

12 12 16 11

19 19 20 12 12 16 21 21 21 11

19 19 20 12 12 16 16 21 21 21 22 22 22 11 11

19 19 19 19 12 12 16 16 21 21 22 22 11

19 19 19 19 12 12 12 16 16 16 16 16 16 22 22 22

EQ Array: 0 1 2 3 3 5 6 7 5 7 7 11 12 11 13 11 16 17 18 12 19 16 22

------- Pass 2 -------

1 1 2

3 3 3 1 1 5 5 5 2

3 3 3 1 1 6 5 5 5 7 7 7 2

3 3 3 3 3 1 1 6 5 5 5 7 2

3 3 3 1 1 6 5 5 5 5 7 7 7 2

3 3 3 3 1 1 6 5 5 5 5 7 7 7

3 3 3 3 1 1 6 5 5 5 5 7

3 3 3 1 1 6 5 5 5 5 7 7 11

3 3 3 1 1 6 5 5 5 5 7 7 11

3 3 3 1 1 6 5 5 5 5 5 7 7 7 11

3 5 11 11

3 3 12 12 12 12 12 5 5 5 11 11 11

3 3 12 12 12 12 12 5 5 5 11 11 11 11

3 3 12 12 12 12 12 5 5 5 11 11

3 3 12 12 12 16 17 5 5 5 18 11 11 11

12 12 16 11

12 12 12 12 12 16 16 16 16 11

12 12 12 12 12 16 16 16 16 16 22 22 22 11 11

12 12 12 12 12 12 16 16 16 16 22 22 11

19 19 12 12 12 12 12 16 16 16 16 16 16 22 22 22

EQ Array: 0 1 2 3 3 5 6 7 5 7 7 11 12 11 11 11 16 17 18 12 12 16 22

------- ManageEQ -------

EQ Array: 0 1 2 3 3 4 5 6 4 6 6 7 8 7 7 7 9 10 11 8 8 9 12

------- Pass 3 -------

1 1 2

3 3 3 1 1 4 4 4 2

3 3 3 1 1 5 4 4 4 6 6 6 2

3 3 3 3 3 1 1 5 4 4 4 6 2

3 3 3 1 1 5 4 4 4 4 6 6 6 2

3 3 3 3 1 1 5 4 4 4 4 6 6 6

3 3 3 3 1 1 5 4 4 4 4 6

3 3 3 1 1 5 4 4 4 4 6 6 7

3 3 3 1 1 5 4 4 4 4 6 6 7

3 3 3 1 1 5 4 4 4 4 4 6 6 6 7

3 4 7 7

3 3 8 8 8 8 8 4 4 4 7 7 7

3 3 8 8 8 8 8 4 4 4 7 7 7 7

3 3 8 8 8 8 8 4 4 4 7 7

3 3 8 8 8 9 10 4 4 4 11 7 7 7

8 8 9 7

8 8 8 8 8 9 9 9 9 7

8 8 8 8 8 9 9 9 9 9 12 12 12 7 7

8 8 8 8 8 8 9 9 9 9 12 12 7

8 8 8 8 8 8 8 9 9 9 9 9 9 12 12 12

**Output 2**

20 25 1 12

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

0 3 3 0 3 0 0 1 1 0 0 0 0 0 4 4 4 0 0 0 0 0 0 0 2

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

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

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

0 3 3 3 3 0 0 1 1 0 5 0 4 0 4 4 4 0 0 6 6 6 0 0 0

3 3 3 0 3 0 0 1 1 0 5 0 4 0 4 4 4 0 0 6 0 0 0 0 0

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

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

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

3 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 7 7

0 3 3 0 8 8 8 8 8 0 0 0 0 0 4 4 4 0 7 0 7 0 7 0 0

0 3 3 0 8 8 8 8 8 0 0 0 0 0 4 4 4 0 0 7 7 0 7 0 7

0 3 3 0 8 8 8 8 8 0 0 0 0 0 4 4 4 0 0 0 0 7 0 0 7

0 3 3 0 8 0 0 8 8 0 9 0 10 0 4 4 4 0 0 11 0 0 7 7 7

0 0 0 0 0 0 0 8 8 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 7

0 8 8 0 8 0 0 8 8 0 0 9 0 0 9 9 9 0 0 0 0 0 0 0 7

0 8 8 0 8 0 0 8 8 0 9 0 9 0 9 9 9 0 0 12 12 12 0 7 7

0 8 8 8 8 0 0 8 8 0 9 0 9 0 9 0 9 0 0 0 12 12 0 0 7

0 8 8 0 8 8 8 8 8 0 9 0 9 9 9 9 9 0 0 12 12 12 0 0 0

**Output 3**

20 25 1 12

12

-----------------

1

20

0 7

9 8

-----------------

2

5

0 24

4 24

-----------------

3

40

1 0

14 4

-----------------

4

47

1 12

14 16

-----------------

5

8

2 10

9 10

-----------------

6

18

2 19

9 21

-----------------

7

22

7 18

18 24

-----------------

8

43

11 1

19 8

-----------------

9

21

14 10

19 16

-----------------

10

1

14 12

14 12

-----------------

11

1

14 19

14 19

-----------------

12

8

17 19

19 21