Project 3 Euclean Distance Transform (JAVA)

CSC 381 Rafael Carmilema

Due Date: Sep / 22 /2016

III. Algorithms  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: read the image header  
           dynamically allocate zerpFramedAry  
  
step 1: 1.1 zeroFrame the array.  
  
           1.2 read from the input file onto zeroFramedAry  
  
step 2: 2.1 firstPassEucleanDistance // algorithm taught in class  
  
           2.2 prettyPrintDistace of the result of Pass-1  
                with caption to Out2  
  
step 3: 3.1 secondPassEucleanDistance // algorithm taught in class  
  
           3.2 prettyPrintDistace of the result of Pass-2  
                with caption to Out2  
  
step 4: Create a distance transform image from the result of Pass-2  
                with image header, to Out1, for future processing.  
  
step 5: Close files

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.io.PrintStream;

import java.util.Scanner;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

// TODO Auto-generated method stub

if ( args.length < 3 ){ // the number of arguments should be 4

System.err.println("Invalid number of arguments.");

System.exit(1);

}

else {

File file = new File(args[0]);

try {

int row;

int col;

int min;

int max;

int data;

int[][] inputArray;

PrintStream outfile1 = new PrintStream(args[1]);

PrintStream outfile2 = new PrintStream(args[2]);

Scanner infile = new Scanner(file);

// Get the header values

row=infile.nextInt();

col= infile.nextInt();

min=infile.nextInt();

max=infile.nextInt();

inputArray = new int[row][col];

//pass the header values into the constructor

Euclean test = new Euclean(row,col,min,max);

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

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

while(infile.hasNext()){

inputArray[r][c]=infile.nextInt();

test.loadImage(inputArray[r][c],r ,c);

break;

}

}

}// outermost for

test.PrintZeroFramed();

test.FirstPassEucleanDistance();

System.out.println();

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

test.PrintZeroFramed();

//outfile2<<"PASS 1"<<endl;

test.PrettyPrint(outfile2);

test.SecondPassEucleanDistance();

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

test.PrintZeroFramed();

//outfile1<<"PASS 2"<<endl;

test.PP(outfile1);

//outfile2<<"PASS 2"<<endl;

test.PrettyPrint(outfile2);

infile.close();

outfile1.close();

outfile2.close();

}

catch(FileNotFoundException e){

System.out.println("File not Found");

}

}//else

}

}

**import** java.io.PrintStream;

**public** **class** Euclean{

**int** row;

**int** col;

**int** min;

**int** max;

**double** newmin;

**double** newmax;

**double**[][] zeroFramedAry;

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

**public** Euclean(**int** r, **int** c, **int** mi, **int** ma){

row=r;

col=c;

min=mi;

max=ma;

zeroFramedAry = **new** **double**[r+2][c+2];

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

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

zeroFramedAry[rows][cols]=0;

}

}

**for**(**int** i=0; i <5;i++){

neighborAry[i]=0;

}

}

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

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

}

**void** loadNeighborsPass1(**int** a, **int** b){

//uppper left

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

//upper middle

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

//upper right

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

//left next to p(i,j)

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

}

**double** findMin(){

**double** minimumd;

**double** euclean= Math.*sqrt*(2);

neighborAry[0]=neighborAry[0]+euclean;

//cout<<neighborAry[0]<<endl; //upper middle

neighborAry[1]=neighborAry[1]+1;

//upper right

neighborAry[2]=neighborAry[2]+euclean;

//left next to p(i,j)

neighborAry[3]=neighborAry[3]+1;

minimumd=neighborAry[0];

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

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

minimumd=neighborAry[i];

}

//return minimumd;

}

**return** minimumd;

}

**void** FirstPassEucleanDistance(){

newmin=0;

newmax=0;

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

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

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

loadNeighborsPass1(i,j);

zeroFramedAry[i][j]=findMin();

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

newmin=zeroFramedAry[i][j];

}

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

newmax=zeroFramedAry[i][j];

}

}

}

}// outer most for loop

}

**void** loadNeighborsPass2(**int** a, **int** b){

//right of p(i,j)

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

//lower left

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

//lower middle

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

//lower right

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

//itself

neighborAry[4]=zeroFramedAry[a][b];

}

**double** findMinPass2(){

**double** minimumd;

**double** euclean= Math.*sqrt*(2);

//right next to p(ij)

neighborAry[0]=neighborAry[0]+1;

//lower left

neighborAry[1]=neighborAry[1]+euclean;

//lower middle

neighborAry[2]=neighborAry[2]+1;

//lower right

neighborAry[3]=neighborAry[3]+euclean;

minimumd=neighborAry[0];

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

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

minimumd=neighborAry[i];

}

}

//check with p(i,j)

**if**(minimumd>neighborAry[4]){

minimumd=neighborAry[4];

}

**return** minimumd;

}

**void** SecondPassEucleanDistance(){

newmin=0;

newmax=0;

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

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

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

loadNeighborsPass2(i,j);

zeroFramedAry[i][j]=findMinPass2();

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

newmin=zeroFramedAry[i][j];

}

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

newmax=zeroFramedAry[i][j];

}

}

}

}//outer for loop

}

**void** PrintZeroFramed(){

System.***out***.println(row+" "+col+" "+newmin+" "+newmax);

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

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

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

}

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

}

}

**void** PrettyPrint(PrintStream outfile2){

outfile2.println(row+" "+col+" "+newmin+" "+newmax);

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

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

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

outfile2.print(" ");

}

**else**{

outfile2.print((**int**) (zeroFramedAry[i][j]+0.5));

}

}

outfile2.println();

}

}

**public** **void** PP(PrintStream outfile2){

outfile2.println(row+" "+col+" "+newmin+" "+newmax);

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

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

outfile2.printf("%.1f",zeroFramedAry[i][j]);

outfile2.print(" ");

}

outfile2.println();

}

}

};

Out1

42 31 0.0 6.65685424949238

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 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 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 1.0 1.4 1.0 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 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 1.4 1.0 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 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 2.8 2.4 1.4 1.0 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 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.4 2.4 2.8 3.8 2.8 2.4 1.4 1.0 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 1.0 0.0 1.0 0.0 0.0 1.0 1.4 2.4 2.8 3.8 4.2 3.8 2.8 2.4 1.4 1.0 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 1.0 0.0 0.0 1.0 0.0 1.0 1.4 2.4 2.8 3.8 4.2 5.2 4.2 3.8 2.8 2.4 1.4 1.0 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 1.0 0.0 1.0 0.0 1.0 2.0 2.8 3.8 4.2 5.2 5.7 5.2 4.2 3.8 2.8 2.0 1.0 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 1.0 0.0 0.0 1.0 0.0 1.0 2.0 3.0 4.0 5.0 5.7 6.7 5.7 5.0 4.0 3.0 2.0 1.0 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 1.0 0.0 1.0 1.0 0.0 1.0 2.0 3.0 4.0 5.0 6.0 6.7 6.0 5.0 4.0 3.0 2.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0

0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 1.0 2.0 3.0 4.0 4.8 5.2 5.7 5.2 4.8 4.2 3.4 2.4 1.4 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0

0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.4 2.4 2.8 3.4 3.8 4.2 5.2 4.2 3.8 3.4 2.8 2.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0 1.4 2.0 2.4 2.8 3.8 4.2 3.8 2.8 2.4 2.0 1.4 1.0 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 1.0 1.0 1.4 2.4 2.8 3.8 2.8 2.4 1.4 1.0 1.0 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 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.4 2.4 2.8 2.4 1.4 1.0 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 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 1.4 1.0 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 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.4 1.0 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 1.0 1.0 0.0 1.0 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 0.0

0.0 0.0 0.0 0.0 1.0 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 0.0 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 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 1.0 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 0.0

0.0 0.0 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 1.4 1.0 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 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0 1.0 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 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 1.4 1.0 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 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 2.8 2.4 1.4 1.0 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 1.0 0.0 1.0 0.0 0.0 0.0 1.0 1.4 2.4 2.8 3.8 2.8 2.4 1.4 1.0 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 1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.4 2.4 2.8 3.8 4.2 3.8 2.8 2.4 1.4 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 1.0 1.4 2.4 2.8 3.8 4.2 5.2 4.2 3.8 2.8 2.4 1.4 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 1.0 2.0 2.8 3.8 4.2 5.2 5.7 5.2 4.2 3.8 2.8 2.0 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 1.0 2.0 3.0 4.0 5.0 5.7 6.7 5.7 5.0 4.0 3.0 2.0 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 1.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0 3.0 4.0 5.0 6.0 6.7 6.0 5.0 4.0 3.0 2.0 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 1.0 0.0 0.0 0.0 0.0 1.0 2.0 3.0 4.0 4.8 5.2 5.7 5.2 4.8 4.0 3.0 2.0 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 1.0 0.0 0.0 0.0 0.0 1.0 2.0 2.8 3.4 3.8 4.2 5.2 4.2 3.8 3.4 2.8 2.0 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 1.0 0.0 0.0 0.0 1.0 1.4 2.0 2.4 2.8 3.8 4.2 3.8 2.8 2.4 2.0 1.4 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 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 1.4 2.4 2.8 3.8 2.8 2.4 1.4 1.0 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 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 2.8 2.4 1.4 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 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.4 2.4 1.4 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 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0 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 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.4 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 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 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Out2

42 31 0.0 8.071067811865476

1

1 111 1

1 1 11211 1

1 1 1123211 1

1 1 112343211 1

1 1 11234443211 1

1 1 1123445443211 1

1 1 1234456544321 1

1 1 1234567654431 1

1 11 1234567765431 1

111 1 1234567776431111111

1 1 11234567876432

1 111 1234567876441

1 12345677653 1

1 1 1234567 1

1 1 12345 1

1 1 123 1

11 1 1

1 1

1 1

1 1

111 1

1 1 121 1

11 11211 1

1 1123211 1

1 1 112343211 1

1 1 11234443211

1123445443211

1234456544321

1234567654431

1 1234567765431

1 1234567776431

1 1234567876431

1 1234567876431

1 1 12345677643

1 1 1234566

1 11 12345

1 1 123

1 121

1

42 31 0.0 6.65685424949238

1

1 111 1

1 1 11211 1

1 1 1123211 1

1 1 112343211 1

1 1 11234443211 1

1 1 1123445443211 1

1 1 1234456544321 1

1 1 1234567654321 1

1 11 1234567654321 1

111 1 1234556554321111111

1 1 11233445443321

1 111 1122344432211

1 11123432111 1

1 1 1123211 1

1 1 11211 1

1 1 111 1

11 1 1

1 1

1 1

1 1

111 1

1 1 121 1

11 11211 1

1 1123211 1

1 1 112343211 1

1 1 11234443211

1123445443211

1234456544321

1234567654321

1 1234567654321

1 1234556554321

1 1233445443321

1 1122344432211

1 1 11123432111

1 1 1123211

1 11 11211

1 1 121

1 111

1