CSCI 381 (Java)

Section 31

Project 4 Distance Transform

Essam Yousry

Due Date of soft copy: 02/23/2018

Due Date of hard copy: 02/27/2018

Algorithm Steps for Distance Transform:

Step 0: Read the image header

Dynamically allocate zeroFramedAry

Step 1: zeroFrame the zeroFramedAry

Step 2: loadImage

Step 3: firstPassDistance

Step 4: prettyPrint of result of pass -1 to outfile2 with caption indicating the result of pass-1

Step 5: secondPassDistance

Step 6: prettyPrint of the result of pass-2 to outfile2

Step 7: Write the result of pass-2 to outfile1 with updated image header

Step 8: Close all files

Source Code

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void zeroFrame(int \*\*array, int row, int col){

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

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

            array[0][j] = 0;

            array[row + 1][j] = 0;

            array[i][0] = 0;

            array[i][col + 1] = 0;

        }

    }

    array[0][0] = 0;

    array[0][col + 1] = 0;

    array[row + 1][0] = 0;

    array[row + 1][col + 1] = 0;

}

void loadImage(int \*\*array, ifstream &myfile, int row, int col){

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

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

            myfile >> array[i][j];

        }

    }

}

int findMin(int x, int y){

    if (x < y) return x;

    else return y;

}

void prettyPrint(int \*\*array, int row, int col, ofstream &myfile){

    for(int x = 1; x < row + 1; x++){

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

            if (array[x][y] > 0){

                myfile << (array[x][y]);

                myfile << ' ';

            }

            else {

                myfile << ' ';

                myfile << ' ';

            }

        }

        myfile << endl;

    }

}

int getMin(int array[], int size){

    int min = array[0];

    for (int i = 0; i < size; i++){

        if (array[i] < min) min = array[i];

    }

    return min;

}

int getMaxD(int array[], int size){

    int max = array[1];

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

        if (array[i] > max) max = array[i];

    }

    return max;

}

void loadNeighborsPass1(int neighborAry[], int \*\*mirrorFramedAry, int i, int j){

    neighborAry[0] = mirrorFramedAry[i - 1][j - 1];

    neighborAry[1] = mirrorFramedAry[i - 1][j];

    neighborAry[2] = mirrorFramedAry[i - 1][j + 1];

    neighborAry[3] = mirrorFramedAry[i][j - 1];

}

void loadNeighborsPass2(int neighborAry[], int \*\*mirrorFramedAry, int i, int j){

    neighborAry[0] = mirrorFramedAry[i][j + 1];

    neighborAry[1] = mirrorFramedAry[i + 1][j - 1];

    neighborAry[2] = mirrorFramedAry[i + 1][j];

    neighborAry[3] = mirrorFramedAry[i + 1][j + 1];

}

void distancePass1(int \*\*array, int row, int col, ofstream &myfile){

    int neighborAry[4];

    myfile << "Pass 1";

    myfile << endl;

    myfile << endl;

    for(int i = 1; i < row + 1; i++){

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

            loadNeighborsPass1(neighborAry, array, i, j);

            if (array[i][j] > 0){

                array[i][j] = getMin(neighborAry, 4) + 1;

            }

        }

    }

    prettyPrint(array, row, col, myfile);

    myfile << endl;

}

void Print(int \*\*array, int row, int col, ofstream &myfile){

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

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

            myfile << array[i][j];

            myfile << ' ';

        }

        myfile << endl;

    }

}

void distancePass2(int \*\*array, int row, int col, ofstream &myfile, ofstream &myfile2){

    int neighborAry[4];

    myfile2 << "Pass 2";

    myfile2 << endl;

    myfile2 << endl;

    for(int i = 1; i < row + 1; i++){

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

            loadNeighborsPass2(neighborAry, array, i, j);

            if (array[i][j] > 0){

                array[i][j] = findMin(array[i][j], getMin(neighborAry, 4) + 1);

            }

        }

    }

    myfile << row;

    myfile << ' ';

    myfile << col;

    myfile << ' ';

    int colValue [row + 1];

    int countCol;

    for(int i = 1; i < row + 1; i++){

        countCol = 0;

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

            if (array[i][j] > 0) countCol++;

        }

        colValue[i] = countCol;

    }

    int newColValue = getMaxD(colValue, row);

    int rowValue [col + 1];

    int countRow;

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

        countRow = 0;

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

            if (array[i][j] > 0) countRow++;

        }

        rowValue[j] = countRow;

    }

    int newRowValue = getMaxD(rowValue, col);

    myfile << newRowValue;

    myfile << ' ';

    myfile << newColValue;

    myfile << endl;

    Print(array, row, col, myfile);

    prettyPrint(array, row, col, myfile2);

}

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

{

    int vars[4], row, col, min, max;

    ifstream myfile;

    myfile.open(argv[1]);

    for(int i = 0; i < 4; i++)

        myfile >> vars[i];

    cout << endl;

    row = vars[0];

    col = vars[1];

    min = vars[2];

    max = vars[3];

    string fileName = argv[1];

    string fileNameWithoutExtension = fileName.substr(0, fileName.rfind("."));

    ofstream myfile2;

    myfile2.open(string(fileNameWithoutExtension + "\_Image" + ".txt"));

    ofstream myfile3;

    myfile3.open(string(fileNameWithoutExtension + "\_PP" + ".txt"));

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

    for (int i = 0; i < row + 2; i++)

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

    loadImage(zeroFramedAry, myfile, row, col);

    zeroFrame(zeroFramedAry, row, col);

    distancePass1(zeroFramedAry, row, col, myfile3);

    distancePass2(zeroFramedAry, row, col, myfile2, myfile3);

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

        delete zeroFramedAry[i];

    }

    delete[] zeroFramedAry;

    myfile.close();

    myfile2.close();

    myfile3.close();

    return 0;

}

Output















