Student: Michael Grossman

Project Due Date: 3/17/2022

Algorithm Steps for firstPass8Dist given an array:

1. newMin ←999; newMax←0
2. Scan array from L to R & T to B starting at (1,1), where i and j are the indices respectively
3. If array[i,j] > 0:
4. Array[i,j] ←1 + min(upper 3 neighbors, and left neighbor)
5. End-if
6. If newMin > Array[i,j]:
7. newMin ← Array[i,j]
8. End-if
9. If newMax < Array[i,j]:
10. newMax ← Array[i,j]
11. repeat 1 to 9 until all pixels are processed

Algorithm steps for secondPass8Dist given an array:

1. newMin ←999; newMax←0
2. Scan array from R to L & B to T starting at (numRows,numCols), where i and j are the indices respectively
3. If array[i,j] > 0:
4. Array[i,j] ←min (upper 3 neighbors + 1 to each, left neighbor + 1, array[i,j])
5. End-if
6. If newMin > Array[i,j]:
7. newMin ← Array[i,j]
8. End-if
9. If newMax < Array[i,j]:
10. newMax ← Array[i,j]
11. repeat 1 to 9 until all pixels are processed

#include <iostream>

#include <fstream>

#include <string>

#include <cmath>

using namespace std;

class imageProcessing{

    public:

        int numRows, numCols, minVal, maxVal;

        int newMin, newMax;

        int\*\* ZFArray;

        imageProcessing(int \*h);

        ~imageProcessing();

        void setZero(int\*\* zfarray);

        void loadImage(ifstream& in, int\*\* ary);

        void firstPass8Distance(int\*\* ary);

        void secondPass8Dustance(int\*\* ary);

        void reformatPrettyPrint(int\*\* array, int min, int max, ofstream& out);

};

int main(int argc, char\*\* argv){

    string inputFileName = argv[1], outputFileName = argv[2];

    ifstream input(inputFileName);

    ofstream output(outputFileName);

    int header[4];

    for(int i = 0; i < 4; ++i){

        input >> header[i];

    }

    imageProcessing imageprocessing(header);

    imageprocessing.setZero(imageprocessing.ZFArray);

    output << "Input Image \n";

    imageprocessing.loadImage(input, imageprocessing.ZFArray);

    imageprocessing.reformatPrettyPrint(imageprocessing.ZFArray, imageprocessing.minVal, imageprocessing.maxVal, output);

    imageprocessing.firstPass8Distance(imageprocessing.ZFArray);

    output << "First Pass image \n";

    imageprocessing.reformatPrettyPrint(imageprocessing.ZFArray, imageprocessing.minVal, imageprocessing.maxVal, output);

    imageprocessing.secondPass8Dustance(imageprocessing.ZFArray);

    output << "Second Pass Image \n";

    imageprocessing.reformatPrettyPrint(imageprocessing.ZFArray, imageprocessing.minVal, imageprocessing.maxVal, output);

    input.close();

    output.close();

}

imageProcessing::imageProcessing(int \*h){

    numRows = h[0];

    numCols = h[1];

    minVal = h[2];

    maxVal = h[3];

    ZFArray = new int\*[numRows + 2];

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

        ZFArray[i] = new int[numCols + 2];

    }

}

imageProcessing::~imageProcessing(){

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

        delete[] ZFArray[i];

    }

    delete[] ZFArray;

}

void imageProcessing::loadImage(ifstream& in, int\*\* ary){

    int rows = numRows+1, cols = numCols + 1;

    for(int i = 1; i < rows; ++i){

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

            in >> ary[i][j];

        }

    }

}

void imageProcessing::setZero(int\*\* zfarray){

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

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

            zfarray[i][j] = 0;

        }

    }

}

void imageProcessing::reformatPrettyPrint(int\*\* array, int min, int max, ofstream& out){

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

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

            if(array[i][j] > 0){

                out << to\_string(array[i][j]) + " ";

            }

            else{

                out << ". ";

            }

        }

        out << "\n";

    }

    out << "\n\n";

}

void imageProcessing::firstPass8Distance(int\*\* ary){

    int newMin = 99999, newMax = 0;

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

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

            if(ary[i][j] > 0){

                int\* inp = new int[4];

                inp[0] = ary[i][j-1];

                int minimum = inp[0];

                inp[1] = ary[i-1][j+1];

                inp[2] = ary[i-1][j];

                inp[3] = ary[i-1][j-1];

                for(int i = 1; i < 4; ++i){

                    if(inp[i] < minimum){

                        minimum = inp[i];

                    }

                }

                ary[i][j] = minimum + 1;

            }

            newMin = newMin > ary[i][j] ? ary[i][j] : newMin;

            newMax = newMax < ary[i][j] ? ary[i][j] : newMax;

        }

    }

}

void imageProcessing::secondPass8Dustance(int\*\* ary){

    int newMin = 99999, newMax = 0;

    for(int i = numRows; i > 0; --i){

        for(int j = numCols; j > 0; --j){

            if(ary[i][j] > 0){

                int\* inp = new int[5];

                inp[0] = ary[i][j+1]+1;

                int minimum = inp[0];

                inp[1] = ary[i+1][j+1] + 1;

                inp[2] = ary[i+1][j] + 1;

                inp[3] = ary[i+1][j-1] + 1;

                inp[4] = ary[i][j];

                for(int i = 1; i < 5; ++i){

                    if(inp[i] < minimum){

                        minimum = inp[i];

                    }

                }

                ary[i][j] = minimum;

            }

            newMin = newMin > ary[i][j] ? ary[i][j] : newMin;

            newMax = newMax < ary[i][j] ? ary[i][j] : newMax;

        }

    }

}





