**CS381-37: Project 9.1 (CPP)**

**Yida Tao**

**Due date: May. 10, 2018**

Algorithm Steps:

step 0: open all files

( numRowsImg, numColsImg, minImg, maxImg ) 🡨 get from input1

( numRowsStrctElem, numColsStrctElem, minStrctElem, maxStrctElem ) 🡨 get from input2

( rowOrigin, colOrigin) 🡨 get from input2

step 1: computeFrameSize

step 2: - dynamically allocate imgAry with extra rows and extra columns

- loadImage // load input file to imgAry

- zeroFrameImg ()

- prettyPrint (imgAry) // pretty print imgAry to the \*console\* of "Input Image"

- dynamically allocate morphAry with extra rows and extra columns

step 3:

- dynamically allocate structElemAry

- loadstruct // load input2 file to structElem array

- prettyPrint (structElemAry)// pretty print to the \*console\* of "Structuring Element"

step 4: - initMorphAry( ) // initialize morphAry to zero

- call dilation // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of dilation

- outputResult //write the delation result to Output1 (argv[3])

step 5: - initMorphAry( ) // initialize morphAry to zero

- call erosion // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of erosion

- outputResult //write the delation result to Output2 (argv[4])

step 6: - initMorphAry( ) // initialize morphAry to zero

- call closing // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* the result of closing

- outputResult //write the closing result to Output3 (argv[5])

step 7: - initMorphAry( ) // initialize morphAry to zero

- call opening // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* with "Opening Result"

- outputResult //write the opening result to Output4 (argv[6])

step 8: close all files

**Source Code**

#include <iostream>

#include <fstream>

#include <string>

#include <cmath>

using namespace std;

class morphology{

public:

int numRowsImg;

int numColsImg;

int minImg;

int maxImg;

int numRowsStructElem;

int numColsStructElem;

int minStructElem;

int maxStructElem;

int rowOrigin;

int colOrigin;

int rowFrameSize;

int colFrameSize;

int \*\*imgAry;

int \*\*morphAry;

int \*\*structElemAry;

morphology(string input1, string input2){

//load struct

ifstream inFile;

inFile.open(input2);

inFile >> numRowsStructElem;

inFile >> numColsStructElem;

inFile >> minStructElem;

inFile >> maxStructElem;

inFile >> rowOrigin;

inFile >> colOrigin;

//compute frame size

rowFrameSize = numRowsStructElem;

colFrameSize = numColsStructElem;

structElemAry = new int\*[numRowsStructElem];

for(int i = 0; i < numRowsStructElem; i++){

structElemAry[i] = new int[numColsStructElem];

for(int j = 0; j < numColsStructElem; j++){

structElemAry[i][j] = 0;

}

}

int r = 0;

int c = 0;

int num = 0;

int counter = 0;

while(inFile>>num){

r = counter/numColsStructElem;

c = counter%numColsStructElem;

structElemAry[r][c] = num;

counter ++;

}

inFile.close();

//load Image

inFile.open(input1);

inFile >> numRowsImg;

inFile >> numColsImg;

inFile >> minImg;

inFile >> maxImg;

//zero frame

imgAry = new int\*[numRowsImg + rowFrameSize];

morphAry = new int\*[numRowsImg + rowFrameSize];

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

imgAry[i] = new int[numColsImg + colFrameSize];

morphAry[i] = new int[numColsImg + colFrameSize];

for(int j = 0; j < numColsImg + colFrameSize; j++){

imgAry[i][j] = 0;

morphAry[i][j] = 0;

}

}

counter = 0;

while(inFile>>num){

r = counter/numColsImg + rowFrameSize/2;

c = counter%numColsImg + colFrameSize/2;

imgAry[r][c] = num;

counter ++;

}

inFile.close();

}

~morphology(){

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

delete imgAry[i];

delete morphAry[i];

}

delete[] imgAry;

delete[] morphAry;

for(int i = 0; i < numRowsStructElem; i++){

delete structElemAry[i];

}

delete[] structElemAry;

}

void initMorphAry(){

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

for(int j = 0; j < numColsImg + colFrameSize; j++){

morphAry[i][j] = 0;

}

}

}

void dilation(){

int rows = numRowsStructElem/2;

int cols = numColsStructElem/2;

for(int i = rowFrameSize/2; i < numRowsImg + rowFrameSize/2; i++){

for(int j = colFrameSize/2; j < numColsImg + colFrameSize/2; j++){

if(imgAry[i][j]==1){

for(int x = -rows; x <= rows; x++){

for(int y = -cols; y <= cols;y++){

if(structElemAry[x+rows][y+cols]>0)

morphAry[i+x][j+y] = structElemAry[x+rows][y+cols];

}

}

}

}

}

}

void erosion(){

int rows = numRowsStructElem/2;

int cols = numColsStructElem/2;

bool check = false;

for(int i = rowFrameSize/2; i < numRowsImg + rowFrameSize/2; i++){

for(int j = colFrameSize/2; j < numColsImg + colFrameSize/2; j++){

if(imgAry[i][j]==1){

check = true;

for(int x = -rows; x <= rows; x++){

for(int y = -cols; y <= cols;y++){

if(structElemAry[x+rows][y+cols]>0 && imgAry[i+x][j+y]==0){

check = false;

}

}

}

if(check){

morphAry[i][j] = 1;

}

}

}

}

}

void closing(){

//dilation

dilation();

int \*\*temp = new int\*[numRowsImg + rowFrameSize];

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

temp[i] = new int[numColsImg + colFrameSize];

for(int j = 0; j < numColsImg + colFrameSize; j++){

temp[i][j] = morphAry[i][j];

}

}

initMorphAry();

//erosion on temp array

int rows = numRowsStructElem/2;

int cols = numColsStructElem/2;

bool check = false;

for(int i = rowFrameSize/2; i < numRowsImg + rowFrameSize/2; i++){

for(int j = colFrameSize/2; j < numColsImg + colFrameSize/2; j++){

if(temp[i][j]==1){

check = true;

for(int x = -rows; x <= rows; x++){

for(int y = -cols; y <= cols;y++){

if(structElemAry[x+rows][y+cols]>0 && temp[i+x][j+y]==0){

check = false;

}

}

}

if(check){

morphAry[i][j] = 1;

}

}

}

}

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

delete temp[i];

}

delete[] temp;

}

void opening(){

//erosion

erosion();

int \*\*temp = new int\*[numRowsImg + rowFrameSize];

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

temp[i] = new int[numColsImg + colFrameSize];

for(int j = 0; j < numColsImg + colFrameSize; j++){

temp[i][j] = morphAry[i][j];

}

}

initMorphAry();

//dilation

int rows = numRowsStructElem/2;

int cols = numColsStructElem/2;

for(int i = rowFrameSize/2; i < numRowsImg + rowFrameSize/2; i++){

for(int j = colFrameSize/2; j < numColsImg + colFrameSize/2; j++){

if(temp[i][j]==1){

for(int x = -rows; x <= rows; x++){

for(int y = -cols; y <= cols;y++){

if(structElemAry[x+rows][y+cols]>0)

morphAry[i+x][j+y] = structElemAry[x+rows][y+cols];

}

}

}

}

}

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

delete temp[i];

}

delete[] temp;

}

void prettyPrint(int\*\* array){

if(array == imgAry || array == morphAry){

for(int i = 0; i < numRowsImg + rowFrameSize; i++){

for(int j = 0; j < numColsImg + colFrameSize; j++){

if(array[i][j] > 0)

cout << array[i][j] <<" ";

else

cout <<" ";

}

cout << endl;

}

}

if(array == structElemAry){

for(int i = 0; i < numRowsStructElem; i++){

for(int j = 0; j < numColsStructElem; j++){

if(array[i][j] > 0)

cout << array[i][j] <<" ";

else

cout <<" ";

}

cout << endl;

}

}

}

void outputResult(){

cout<< numRowsImg << " " << numColsImg << " " << minImg << " " << maxImg << endl;

for(int i = rowFrameSize/2; i < numRowsImg + rowFrameSize/2; i++){

for(int j = colFrameSize/2; j < numColsImg + colFrameSize/2; j++){

if(morphAry[i][j] > 0){

cout << morphAry[i][j] <<" ";

}

else{

cout << " ";

}

}

cout << endl;

}

}

};

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

if(argv[1]==NULL) {

cout<<"no parameter"<<endl;

return 0;

}

ofstream out1;

out1.open(argv[3]);

ofstream out2;

out2.open(argv[4]);

ofstream out3;

out3.open(argv[5]);

ofstream out4;

out4.open(argv[6]);

streambuf \*console = cout.rdbuf();

//step 0,1,2,3

morphology \*mp = new morphology(argv[1],argv[2]);

cout<<"prettyPrint (imgAry):"<<endl;

mp->prettyPrint(mp->imgAry);

cout<<"prettyPrint (structElemAry):"<<endl;

mp->prettyPrint(mp->structElemAry);

int srows = mp->rowFrameSize/2;

int scols = mp->colFrameSize/2;

int imgrows = mp->numRowsImg;

int imgcols = mp->numColsImg;

//step 4

mp->initMorphAry();

mp->dilation();

cout<<"prettyPrint (morphAry): dilation"<<endl;

mp->prettyPrint(mp->morphAry);

cout.rdbuf(out1.rdbuf());

mp->outputResult();

cout.rdbuf(console);

//step 5

mp->initMorphAry();

mp->erosion();

cout<<"prettyPrint (morphAry): erosion"<<endl;

mp->prettyPrint(mp->morphAry);

cout.rdbuf(out2.rdbuf());

mp->outputResult();

cout.rdbuf(console);

//step 6

mp->initMorphAry();

mp->closing();

cout<<"prettyPrint (morphAry): closing"<<endl;

mp->prettyPrint(mp->morphAry);

cout.rdbuf(out3.rdbuf());

mp->outputResult();

cout.rdbuf(console);

//step 7

mp->initMorphAry();

mp->opening();

cout<<"prettyPrint (morphAry): opening"<<endl;

mp->prettyPrint(mp->morphAry);

cout.rdbuf(out4.rdbuf());

mp->outputResult();

cout.rdbuf(console);

cout<<"done"<<endl;

out1.close();

out2.close();

out3.close();

out4.close();

delete mp;

return 0;

}

**Input1.1: Img1.txt**

42 31 0 1

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

1 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 1 0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Input1.2: StrucElem1.txt**

3 3 0 1

1 1

0 1 0

1 1 1

0 1 0

**Input2.1: Img2.txt**

21 21 0 1

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

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

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

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

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**Input1.2: StrucElem2.txt**

3 3 1 1

1 1

1 1 1

1 1 1

1 1 1

**Output: dilation1**

42 31 0 1

1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1

**Output: erosion1**

42 31 0 1

1

1 1 1

1 1 1

1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1

1

1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1

1 1

1 1

1

1

1 1

1

1 1 1

1 1 1

1 1 1

1 1

1 1

1 1 1

1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1

1

1

**Output: closing1**

42 31 0 1

1 1

1 1 1 1

1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1

1

1 1

1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1

**Output: opening1**

42 31 0 1

1

1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1

1 1 1

1

1

1 1 1 1

1 1 1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1

1 1 1

1

**Output: dilation2**

21 21 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**Output: erosion2**

21 21 0 1

1 1

1 1

1 1

1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1

1 1

1 1

1 1

1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1

1 1

1 1

1 1

1 1

1 1

1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**Output: closing2**

21 21 0 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**Output: opening2**

21 21 0 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1