**CSCI 381 – Computer Vision (C++)**

**Program: Project 6: DistanceTransformSkeletonCompression**

**Name: Isaac Gordon**

**Due Date:**

**Soft copy: 4/05/2019 Friday before midnight**

**Hard copy: 4/09/2019 Tuesday in class**

step 0: inFile 🡨 open input file

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate zeroFramedAry with extra 2 rows and 2 cols

dynamically allocate skeletonAry with extra 2 rows and 2 cols

open outFile\_1, outFile\_2

Step 1: skeletonFileName 🡨 argv[1] + “\_skeleton”

Step 2: skeletonFile 🡨 open (skeletonFileName)

Step 3: deCompressedFileName 🡨 argv[1] + “\_deCompressed”

Step 4: deCompressFile 🡨 open (deCompressedFileName)

step 5: setZero (ZeroFramedAry)

setZero (skeletonAry)

Step 6: loadImage (inFile, ZeroFramedAry) // begins at ZeroFramedAry(1,1)

Step 7: compute8Distance(ZeroFramedAry, outFile1) // Perform distance transform

Step 8: skeletonExtraction (ZeroFramedAry, skeletonAry, skeletonFile, outFile1)

// perform compression

Step 9: skeletonExpansion(ZeroFramedAry, skeletonFile, outFile2)

// perform decompression

step 10: Output numRows, numCols, newMinVal, newMaxVal to deCompressFile

Step 11: ary2File(ZeroFramedAry, deCompressFile)

// dump ZeroFramedAry to deCompressFile

Step 12: close all files

**CODE**

#include <iostream>

#include <fstream>

#include <cmath>

#include <string>

#include <cstring>

using namespace std;

/\* Data Structures \*/

//files

ifstream inFile1;

ofstream outFile1, outFile2, skeletonFile, decompressionFile;

//data

int numRows, numCols, minVal, maxVal, newMinVal, newMaxVal;

int\*\* zeroFramedAry;

int\*\* skeletonAry;

int\* neighborhood;

int setup(int argc, char \*argv[]);

void setZero(int \*\*&ary);

void loadImage();

void loadNeighbors(int i, int j);

void compute8Distance(int \*\*&ary, ofstream &file);

void firstPass8Distance(int \*\*&ary);

void secondPass8Distance(int \*\*&ary);

void skeletonExtraction();

int isMaxima(int i, int j);

void computeLocalMaxima(int \*\*&zAry, int \*\*&skelAry);

void extractLocalMaxima(int \*\*&skelAry, ofstream &file);

void skeletonExpansion(ifstream &skelfile, int \*\*&ary);

void loadFromSkeleton(ifstream &file, int \*\*ary);

void firstPassExtraction(int \*\*&ary);

void secondPassExtraction(int \*\*&ary);

void sendToFile(int \*\*&ary, ofstream &file);

void prettyPrint(int \*\*&ary, ofstream &file, bool includeZero);

void ary2file(int \*\*&ary, ofstream &file);

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

//Step 0: open files

int notgood = setup(argc, argv);

if(notgood == -1){

return 0;

}//if

//Step 1: skeletonFileName <- argv[1] + “\_skeleton”

string skeletonFileName;

skeletonFileName = argv[1];

string sk = skeletonFileName.substr(0,skeletonFileName.find(".txt",0));

skeletonFileName = sk + "\_skeleton.txt";

//Step 2: skeletonFile <- open (skeletonFileName)

char skCharAry[skeletonFileName.length() + 1];

strcpy(skCharAry, skeletonFileName.c\_str());

skeletonFile.open(skCharAry);

//Step 3: deCompressedFileName <- argv[1] + “\_deCompressed”

string decompressionFileName;

decompressionFileName = argv[1];

string de = decompressionFileName.substr(0,decompressionFileName.find(".txt",0));

decompressionFileName = de + "\_decompressed.txt";

//Step 4: deCompressFile <- open (deCompressedFileName)

char deCharAry[decompressionFileName.length() + 1];

strcpy(deCharAry, decompressionFileName.c\_str());

decompressionFile.open(deCharAry);

//step 5: setZero (ZeroFramedAry), setZero (skeletonAry)

/\* THIS STEP WAS TAKEN CARE OF IN SETUP \*/

//Step 6: loadImage (inFile, ZeroFramedAry) // begins at ZeroFramedAry(1,1)

loadImage();

//Step 7: compute8Distance(ZeroFramedAry, outFile1) // Perform distance transform

compute8Distance(zeroFramedAry, outFile1);

//Step 8: skeletonExtraction (ZeroFramedAry, skeletonAry, skeletonFile, outFile1)

// perform compression

skeletonExtraction();

skeletonFile.close();

//Step 9: skeletonExpansion(ZeroFramedAry, skeletonFile, outFile2)

// perform decompression

ifstream skstream;

skstream.open(skCharAry);

skeletonExpansion(skstream, zeroFramedAry);

//step 10: Output numRows, numCols, newMinVal, newMaxVal to deCompressFile

decompressionFile << numRows << " " << numCols << " " << newMinVal << " " << newMaxVal << endl;

//Step 11: ary2File(ZeroFramedAry, deCompressFile)

// dump ZeroFramedAry to deCompressFile

ary2file(zeroFramedAry, decompressionFile);

//Step 12: close all files

inFile1.close();

outFile1.close();

outFile2.close();

skeletonFile.close();

decompressionFile.close();

}//main

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

string err = "";

err += "Improper arguements. Correct syntax is: \n>> ... <input1.txt> <output1.txt> <output2.txt>";

err += "\n\twhere: \n\tinput1.txt is a binary image with header \n\toutput1.txt is a file to ";

err += " print the result of each pass of the compression \n\toutput2.txt is a file to print the ";

err += "results of each pass in the decompression.\n";

//check for bad num or args

if(argc != 4){

cout << err <<endl;

return -1;

}//if

//get filestreams

inFile1.open(argv[1]);

outFile1.open(argv[2]);

outFile2.open(argv[3]);

//set header vals

inFile1 >> numRows;

inFile1 >> numCols;

inFile1 >> minVal;

inFile1 >> maxVal;

//allocate arys

setZero(zeroFramedAry);

setZero(skeletonAry);

return 0;

}//setup

void setZero(int \*\*&ary){

ary = new int\*[numRows + 2];

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

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

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

ary[i][j] = 0;

}//for

}//for

}//setZero

void loadImage(){

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

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

inFile1 >> zeroFramedAry[i+1][j+1];

}//for

}//for

}//loadImage

void loadNeighbors(int i, int j){

neighborhood = new int[9];

int index = 0;

for(int r = i-1; r <= i+1; r++){

for(int c = j-1; c <= j+1; c++){

neighborhood[index] = zeroFramedAry[r][c];

index++;

}//for

}//for

}//loadNeighbors

void compute8Distance(int \*\*&ary, ofstream &file){

firstPass8Distance(ary);

outFile1 << "1st PASS DISTANCE TRANSFORM\n\n";

prettyPrint(ary, file, false);

secondPass8Distance(ary);

outFile1 << "2nd PASS DISTANCE TRANSFORM\n\n";

prettyPrint(ary, file, false);

}//compute8Distance

void firstPass8Distance(int \*\*&ary){

//for every pixel, get its distance form the edge

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

for(int j = 0; j < numCols + 1; j++){

//if p(i,j) is an object pixel,

//set it equal to minimum neighbor +1

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

loadNeighbors(i,j);

//set

int min = 1000000;

for(int k = 0; k < 4; k++){

if(neighborhood[k] < min) min = neighborhood[k];

}//for

zeroFramedAry[i][j] = min + 1;

}//if

}//for cols

}//for rows

}//firstPass8Distance

void secondPass8Distance(int \*\*&ary){

newMinVal = 1000000;

newMaxVal = -10;

for(int i = numRows; i > 0; i--){

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

//if p(i,j) is an object pixel,

//set it equal to minimum neighbor +1 or itself, whichever is smaller

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

loadNeighbors(i,j);

int min = zeroFramedAry[i][j];

for(int k = 5; k < 9; k++){

if(neighborhood[k]+1 < min) min = neighborhood[k]+1;

}//for

zeroFramedAry[i][j] = min;

//keep track of new min and max values

if(zeroFramedAry[i][j] > newMaxVal) newMaxVal = zeroFramedAry[i][j];

if(zeroFramedAry[i][j] < newMinVal) newMinVal = zeroFramedAry[i][j];

}//if nonzero

}//for rows

}//for cols

}//secondPass8Distance

void skeletonExtraction(){

computeLocalMaxima(zeroFramedAry, skeletonAry);

outFile1 <<"\nLOCAL MAXIMA\n\n";

prettyPrint(skeletonAry, outFile1, false);

extractLocalMaxima(skeletonAry, skeletonFile);

}//skeletonExtraction

void computeLocalMaxima(int \*\*&zAry, int \*\*&skelAry){

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

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

if(zAry[i][j] != 0){

loadNeighbors(i,j);

int max = -1;

for(int k = 0; k < 9; k++){

if(neighborhood[k]> max) max = neighborhood[k];

if(zAry[i][j] == max) skelAry[i][j]=zAry[i][j];

else skelAry[i][j] = 0;

}//for

}//if

}//for cols

}//for rows

}//computeLocalMaxima

void extractLocalMaxima(int \*\*&skelAry, ofstream &file){

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

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

if(skelAry[i][j] != 0){

file << i << " " << j << " " << zeroFramedAry[i][j] << endl;

}//if

}//for cols

}//for rows

}//extractLocalMaxima

void skeletonExpansion(ifstream &skelfile,int \*\*&ary){

//set and load

setZero(ary);

loadFromSkeleton(skelfile, ary);

//first pass

firstPassExtraction(ary);

outFile2 << "1st PASS EXPANSION\n\n";

prettyPrint(ary, outFile2, false);

//second pass

secondPassExtraction(ary);

outFile2 << "2nd PASS EXPANSION\n\n";

prettyPrint(ary, outFile2, false);

}//skeletonExpansion

void loadFromSkeleton(ifstream &file, int \*\*ary){

int r, c, dist;

while(!file.eof()){

file >> r;

file >> c;

file >> dist;

ary[r][c] = dist;

}//while

}//loadFromSkeleton

void firstPassExtraction(int \*\*&ary){

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

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

if(ary[i][j] == 0){

loadNeighbors(i,j);

int max = -10;

for(int k = 0; k < 9;k++){

if(max < neighborhood[k]) max = neighborhood[k];

}//for

if(max - 1 > 0) ary[i][j] = max - 1;

else ary[i][j] = 0;

}//if

}//for

}//for

}//firstPassExtraction

void secondPassExtraction(int \*\*&ary){

for(int i = numRows; i > 0; i--){

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

//if(ary[i][j] == 0){

loadNeighbors(i,j);

int max = -10;

for(int k = 0; k < 9;k++){

if(max < neighborhood[k]) max = neighborhood[k];

}//for

if(max - 1 > ary[i][j]) ary[i][j] = max - 1;

//keep track of new min and max values

if(zeroFramedAry[i][j] > newMaxVal) newMaxVal = zeroFramedAry[i][j];

if(zeroFramedAry[i][j] < newMinVal) newMinVal = zeroFramedAry[i][j];

//}//if

}//for

}//for

}//secondPassExtraction

void prettyPrint(int \*\*&ary, ofstream &file, bool includeZero){

// if Ary(i,j) == 0 print 2 blank space, else print Ary(i,j) use 2 digit space

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

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

if(ary[i][j] == 0) file << " ";

else file << ary[i][j] << " ";

}//for

file << endl;

}//for

}//prettyprint

void ary2file(int \*\*&ary, ofstream &file){

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

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

if(ary[i][j] != 0){

file << 1 << " ";

}

else file << 0 << " ";

}//for

file << endl;

}//for

}//ary2file

**Input File**

30 40 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 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 1 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 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 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 0 0 0 0 0 0 0 0 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 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 1 1 1 1 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 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 0 0 0 0 0 0 0 0 0 0 1 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 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 1 1 0 0

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

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

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

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

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

**Output 1**

1st PASS DISTANCE TRANSFORM

1

1 1 1

1 1 2 1 1

1 1 2 2 2 1 1

1 1 2 2 3 2 2 1 1

1 1 2 2 3 3 3 2 2 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

2nd PASS DISTANCE TRANSFORM

1

1 1 1

1 1 2 1 1

1 1 2 2 2 1 1

1 1 2 2 3 2 2 1 1

1 1 2 2 3 3 3 2 2 1 1

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 2 2 1 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 3 2 2 1 1 1

1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 2 2 2 1 1

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

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

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

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

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

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

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

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

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

1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1

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

LOCAL MAXIMA

1

2

3

4

5

1 2 3 4 5 5 5 4 3 2 1

4

3

2

1

8 8

8 8

8 8

**Output 2**

1st PASS EXPANSION

1

1 1 1

1 2 1

2 2 2 1

1 2 3 2 1

1 3 3 3 2 1

2 3 4 3 2 1

1 2 4 4 4 3 2 1

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

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

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

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

1 1 2 2 2 3 3 3 2 2 2 1 1

1 1 1 2 2 3 2 2 1 1 1

1 1 2 2 2 1 1

1 1 2 1 1

1 1 1

1

7 7 7 7 6 5 4 3 2 1

6 7 8 8 7 6 5 4 3 2 1

5 6 7 8 8 7 6 5 4 3 2 1

4 5 6 7 8 8 7 6 5 4 3 2 1

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

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

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

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

1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1

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

2nd PASS EXPANSION

1

1 1 1

1 1 2 1 1

1 1 2 2 2 1 1

1 1 2 2 3 2 2 1 1

1 1 2 2 3 3 3 2 2 1 1

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 2 2 1 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 3 2 2 1 1 1

1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 2 2 2 1 1

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

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

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

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

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

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

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

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

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

1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1

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

**Skeleton File**

1 31 1

3 31 2

5 31 3

7 31 4

9 31 5

10 22 1

10 24 2

10 26 3

10 28 4

10 30 5

10 31 5

10 32 5

10 34 4

10 36 3

10 38 2

10 40 1

12 31 4

14 31 3

16 31 2

18 31 1

20 11 8

20 12 8

21 11 8

21 12 8

22 11 8

22 12 8

**De-Compressed File**

30 40 0 8

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

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

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

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

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

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