CV Project 4: Connected Components C++

**Student:** Pawan Bhatta

**Project Due Date:** 03/21/2021

**Algorithm for Pass 1:**

Step 0: image ← given the Binary Image

newLabel ← 0 (background is 0)

eqTable ← Equivalency Table

Step 1: scan the image Left→Right and Top→Bottom

P( i,j ) ← next pixel

Step 2: if P( i,j ) > 0 // skip 0s

look at: a, b, c, d

**Case 1**: a = b = c = d = 0

newLabel ++ //increment new label

P( i,j ) ← newLabel

**Case 2**: some/all of a,b,c,d already have the same label

P( i,j ) ← same label //give same label

**Case 3**: some/all of a,b,c,d already have labels but their

labels are NOT the same (excluding 0)

P( i,j ) ← minLabel = min( a, b, c, d) //give smallest

Step 3: if case-1 or case-3, update the Equivalency Table

**Algorithm for Pass 2:**

Step 1: scan the result of Pass-1 Right→Left and Bottom→Top

P( i,j ) ← next pixel

Step 2: if P( i,j ) > 0 // skip 0s

look at: e, f, g, h, P( i,j )

**Case 1**: e = f = g = h = 0

do nothing P( i,j ) keeps its label

**Case 2**: e = f = g = h = P( i,j ) all/some have the same label

(excluding 0) do nothing P( i,j ) keeps its label

**Case 3**: at least 2 among e, f, g, h, P( i,j ) have different labels (excluding 0)

minLabel ← min( e, f, g, h, P( i,j ) ) (excluding 0) //

find smallest label if P( i,j ) > minLabel

EQTable[ P( i,j ) ] ← minLabel

P( i,j ) ← minLabel

Step 3: use the Equivalency Table to update P( i,j ) that was NOT updated in Step-2

Step 4: repeat steps 1 to 3 until ALL pixels are processed

**Algorithm for Equivalency Table Management:**

Step 0: readLabel ← 0

Step 1: index ← 1

Step 2: if index ≠ EQ[ index ]

EQ[ index ] ← EQ[ EQ[ index ] ]

else

readLabel++

EQ[ index ] ← readLabel

Step 3: index++

Step 4: repeat steps 2 to 3 until index > newLabel

**Algorithm for Pass 3:**

Step 0: PropertyFile - array of size maxLabel + 1

image ← result of Pass-2

Step 1: scan the image Left→Right and Top→Bottom

or Right→Left and Bottom→Top

P( i,j ) ← next pixel

Step 2: if P( i,j ) > 0

P( i,j ) ← EquivalencyTable[ P( i,j ) ]

PropertyFile[ P( i,j ) ] PixelCount++

PropertyFile[ P( i,j ) ] ← minRow minCol maxRow maxCol

Step 3: repeat steps 1 to 2 for ALL pixels

**Source Code:**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdarg>

using namespace std;

//helper functions that returns first non zero value seen in a given list of params

template <class T>

T getNonZero(T n)

{

if (n != 0)

{

return n;

}

else

{

return 0;

}

}

template <class T, class... Args>

T getNonZero(T n, Args... args)

{

if (n != 0)

{

return n;

}

return getNonZero(args...);

}

//helper functions that returns minimum value ignoring zero

template <class T>

T getMinVal(T a)

{

return a;

}

template <class T>

T getMinVal(T a, T b)

{

if (a == 0 && b == 0)

{

return 0;

}

if (a == 0)

{

return b;

}

if (b == 0)

{

return a;

}

if (a < b)

{

return a;

}

return b;

}

template <class T, class... Args>

T getMinVal(T a, T b, Args... args)

{

if (a == 0 && b == 0)

{

return getMinVal(args...);

}

if (a == 0)

{

return getMinVal(b, args...);

}

if (b == 0)

{

return getMinVal(a, args...);

}

if (a < b)

{

return getMinVal(a, args...);

}

return getMinVal(b, args...);

}

//helper function that returns true if list of parameters includes all same values ignoring zero else returns false

template <class T>

bool isSameValExZero(T a)

{

return true;

}

template <class T>

bool isSameValExZero(T a, T b)

{

if (b == 0 || a == 0)

{

return true;

}

if (a == b)

{

return true;

}

return false;

}

template <class T, class... Args>

bool isSameValExZero(T a, T b, Args... args)

{

if (a == 0 && b == 0)

{

return isSameValExZero(args...);

}

else if (a == 0)

{

return isSameValExZero(b, args...);

}

else if (b == 0)

{

return isSameValExZero(a, args...);

}

else if (a == b)

{

return isSameValExZero(b, args...);

}

else

{

return false;

}

}

struct Property

{

int label; // The component label

int numPixels; // total number of pixels in the cc.

int minR;

int minC;

int maxR;

int maxC;

Property()

{

label = -1;

numPixels = 0;

minC = 9999;

minR = 9999;

maxC = 0;

maxR = 0;

}

};

class CClabel

{

public:

int numRows;

int numCols;

int minVal;

int maxVal;

int newMin;

int newMax;

int rowFrameSize;

int colFrameSize;

int extraRows;

int extraCols;

int newLabel;

int trueNumCC; // the true number of connected components in the image

int \*\*zeroFramedAry;

int NonZeroNeighborAry[5];

int \*EQAry; // an 1-D array, of size (numRows \* numCols) / 4

Property \*CCproperty;

CClabel(ifstream &input)

{

loadHeader(input);

rowFrameSize = 1;

colFrameSize = 1;

extraRows = 2 \* rowFrameSize;

extraCols = 2 \* colFrameSize;

//dynamic allocation of zeroframeArray

zeroFramedAry = new int \*[numRows + extraRows];

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

{

zeroFramedAry[i] = new int[numCols + extraCols];

}

zero2D(zeroFramedAry, numRows + extraRows, numCols + extraCols);

}

void loadHeader(ifstream &input)

{

input >> numRows >> numCols >> minVal >> maxVal;

}

void loadImage(ifstream &input)

{

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

input >> zeroFramedAry[i][j];

}

}

}

void connect8Pass1()

{

newLabel = 0;

//allocating EQTable

int EQSize = (numRows \* numCols) / 4;

EQAry = new int[EQSize];

for (int i = 0; i < EQSize; i++)

{

EQAry[i] = i;

}

newMax = 0;

newMin = 9999;

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

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

{

int a = zeroFramedAry[i - 1][j - 1];

int b = zeroFramedAry[i - 1][j];

int c = zeroFramedAry[i - 1][j + 1];

int d = zeroFramedAry[i][j - 1];

//Case 1

if (a == 0 && b == 0 && c == 0 && d == 0)

{

newLabel++;

zeroFramedAry[i][j] = newLabel;

//updating EQ table

EQAry[newLabel] = newLabel;

}

//Case 2

else if (isSameValExZero(a, b, c, d))

{

zeroFramedAry[i][j] = getNonZero(a, b, c, d);

}

//Case 3

else

{

int minVal = getMinVal(a, b, c, d);

zeroFramedAry[i][j] = minVal;

//updating EQ Table

EQAry[a] = minVal;

EQAry[b] = minVal;

EQAry[c] = minVal;

EQAry[d] = minVal;

}

//Updating newMax and newMin

if (zeroFramedAry[i][j] < newMin)

{

newMin = zeroFramedAry[i][j];

}

if (zeroFramedAry[i][j] > newMax)

{

newMax = zeroFramedAry[i][j];

}

}

}

}

}

void connect4Pass1()

{

newLabel = 0;

//allocating EQTable

int EQSize = (numRows \* numCols) / 4;

EQAry = new int[EQSize];

for (int i = 0; i < EQSize; i++)

{

EQAry[i] = i;

}

newMax = 0;

newMin = 9999;

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

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

{

int a = zeroFramedAry[i - 1][j];

int b = zeroFramedAry[i][j - 1];

//Case 1

if (a == 0 && b == 0)

{

newLabel++;

zeroFramedAry[i][j] = newLabel;

//updating EQ table

EQAry[newLabel] = newLabel;

}

//Case 2

else if (isSameValExZero(a, b))

{

zeroFramedAry[i][j] = getNonZero(a, b);

}

//Case 3

else

{

int minVal = getMinVal(a, b);

zeroFramedAry[i][j] = minVal;

//updating EQ Table

EQAry[a] = minVal;

EQAry[b] = minVal;

}

//Updating newMax and newMin

if (zeroFramedAry[i][j] < newMin)

{

newMin = zeroFramedAry[i][j];

}

if (zeroFramedAry[i][j] > newMax)

{

newMax = zeroFramedAry[i][j];

}

}

}

}

}

void connect8Pass2()

{

newMax = 0;

newMin = 9999;

for (int i = numRows + rowFrameSize - 1; i >= rowFrameSize; i--)

{

for (int j = numCols + colFrameSize - 1; j >= colFrameSize; j--)

{

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

{

int e = zeroFramedAry[i][j + 1];

int f = zeroFramedAry[i + 1][j - 1];

int g = zeroFramedAry[i + 1][j];

int h = zeroFramedAry[i + 1][j + 1];

//Case 1

if (e == 0 && f == 0 && g == 0 && h == 0)

{

//do nothing

}

//Case 2

else if (isSameValExZero(e, f, g, h, zeroFramedAry[i][j]))

{

//do nothing

}

//Case 3

else

{

int minLabel = getMinVal(e, f, g, h, zeroFramedAry[i][j]);

zeroFramedAry[i][j] = minLabel;

//Updating EQ Table

if (zeroFramedAry[i][j] > minLabel)

{

EQAry[zeroFramedAry[i][j]] = minLabel;

}

EQAry[e] = minLabel;

EQAry[f] = minLabel;

EQAry[g] = minLabel;

EQAry[h] = minLabel;

}

//Updating newMax and newMin

if (zeroFramedAry[i][j] < newMin)

{

newMin = zeroFramedAry[i][j];

}

if (zeroFramedAry[i][j] > newMax)

{

newMax = zeroFramedAry[i][j];

}

}

}

}

}

void connect4Pass2()

{

newMax = 0;

newMin = 9999;

for (int i = numRows + rowFrameSize - 1; i >= rowFrameSize; i--)

{

for (int j = numCols + colFrameSize - 1; j >= colFrameSize; j--)

{

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

{

int c = zeroFramedAry[i][j + 1];

int d = zeroFramedAry[i + 1][j];

//Case 1

if (c == 0 && d == 0)

{

//do nothing

}

//Case 2

else if (isSameValExZero(c, d, zeroFramedAry[i][j]))

{

//do nothing

}

//Case 3

else

{

int minLabel = getMinVal(c, d, zeroFramedAry[i][j]);

zeroFramedAry[i][j] = minLabel;

//Updating EQ Table

if (zeroFramedAry[i][j] > minLabel)

{

EQAry[zeroFramedAry[i][j]] = minLabel;

}

EQAry[c] = minLabel;

EQAry[d] = minLabel;

}

//Updating newMax and newMin

if (zeroFramedAry[i][j] < newMin)

{

newMin = zeroFramedAry[i][j];

}

if (zeroFramedAry[i][j] > newMax)

{

newMax = zeroFramedAry[i][j];

}

}

}

}

}

void connectPass3()

{

newMax = 0;

newMin = 9999;

CCproperty = new Property[trueNumCC + 1]();

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

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

{

zeroFramedAry[i][j] = EQAry[zeroFramedAry[i][j]];

Property \*p = &CCproperty[zeroFramedAry[i][j]];

p->label = zeroFramedAry[i][j];

p->numPixels = p->numPixels + 1;

if (p->minR > i - 1)

{

p->minR = i - 1;

}

if (p->maxR < i - 1)

{

p->maxR = i - 1;

}

if (p->minC > j - 1)

{

p->minC = j - 1;

}

if (p->maxC < j - 1)

{

p->maxC = j - 1;

}

}

//Updating newMax and newMin

if (zeroFramedAry[i][j] < newMin)

{

newMin = zeroFramedAry[i][j];

}

if (zeroFramedAry[i][j] > newMax)

{

newMax = zeroFramedAry[i][j];

}

}

}

}

void manageEQAry()

{

int readLabel = 0;

for (int i = 1; i <= newLabel; i++)

{

if (i != EQAry[i])

{

EQAry[i] = EQAry[EQAry[i]];

}

else

{

readLabel++;

EQAry[i] = readLabel;

}

}

trueNumCC = readLabel;

}

void zero2D(int \*\*ary, int numOfRows, int numOfCols)

{

for (int i = 0; i < numOfRows; i++)

{

for (int j = 0; j < numOfCols; j++)

{

ary[i][j] = 0;

}

}

}

void minus1D(int \*ary, int arrayLength)

{

for (int i = 0; i < arrayLength; i++)

{

ary[i] = -1;

}

}

void print2DArray(int \*\*ary, int numOfRows, int numOfCols)

{

cout << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;

for (int i = 0; i < numOfRows; i++)

{

for (int j = 0; j < numOfCols; j++)

{

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

}

cout << endl;

}

}

void imgReformat(ofstream &outFile)

{

outFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;

string str = to\_string(newMax);

int width = str.length();

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

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

{

outFile << "."

<< " ";

}

else

{

outFile << zeroFramedAry[i][j] << " ";

}

str = to\_string(zeroFramedAry[i][j]);

int ww = str.length();

while (ww < width)

{

outFile << " ";

ww++;

}

}

outFile << endl;

}

}

void printImg(ofstream &outFile)

{

outFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;

string str = to\_string(newMax);

int width = str.length();

for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)

{

for (int j = colFrameSize; j < numCols + colFrameSize; j++)

{

outFile << zeroFramedAry[i][j] << " ";

str = to\_string(zeroFramedAry[i][j]);

int ww = str.length();

while (ww < width)

{

outFile << " ";

ww++;

}

}

outFile << endl;

}

}

void printEQAry(ofstream &outFile)

{

for (int i = 1; i <= newLabel; i++)

{

outFile << i << " " << EQAry[i] << endl;

}

}

void printCCproperty(ofstream &outFile)

{

outFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;

outFile << trueNumCC << endl;

outFile << "---" << endl;

for (int i = 1; i < trueNumCC + 1; i++)

{

Property \*p = &CCproperty[i];

outFile << p->label << endl;

outFile << p->numPixels << endl;

outFile << p->minR << " " << p->minC << endl;

outFile << p->maxR << " " << p->maxC << endl;

outFile << "---" << endl;

}

}

void drawBoxes()

{

int minRow, minCol, maxRow, maxCol, label;

for (int i = 1; i < trueNumCC + 1; i++)

{

label = CCproperty[i].label;

minRow = CCproperty[i].minR + 1;

minCol = CCproperty[i].minC + 1;

maxRow = CCproperty[i].maxR + 1;

maxCol = CCproperty[i].maxC + 1;

//drawing horizontal top line

for (int j = minCol; j <= maxCol; j++)

{

zeroFramedAry[minRow][j] = label;

}

//drawing horizontal bottom line

for (int j = minCol; j <= maxCol; j++)

{

zeroFramedAry[maxRow][j] = label;

}

//drawing vertical left line

for (int i = minRow; i <= maxRow; i++)

{

zeroFramedAry[i][minCol] = label;

}

//drawing vertical right line

for (int i = minRow; i <= maxRow; i++)

{

zeroFramedAry[i][maxCol] = label;

}

}

}

~CClabel()

{

//Cleaning up

delete[] EQAry;

delete[] CCproperty;

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

{

delete[] zeroFramedAry[i];

}

}

};

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

{

//READ

string inputName = argv[1];

ifstream input;

input.open(inputName);

int connectedness = stoi(argv[2]);

//WRITES

string rfPrettyPrintFileName = argv[3], labelFileName = argv[4], propertyFileName = argv[5];

ofstream rfPrettyPrint, labelFile, propertyFile;

rfPrettyPrint.open(rfPrettyPrintFileName);

labelFile.open(labelFileName);

propertyFile.open(propertyFileName);

//Checking if IO operations succeeds

if (input.is\_open())

{

if (rfPrettyPrint.is\_open() && labelFile.is\_open() && propertyFile.is\_open())

{

CClabel cc(input);

cc.loadImage(input);

if (connectedness == 4)

{

cc.connect4Pass1();

rfPrettyPrint << "Pass 1" << endl;

cc.imgReformat(rfPrettyPrint);

rfPrettyPrint << endl

<< "Equivalency Array after: Pass 1" << endl;

cc.printEQAry(rfPrettyPrint);

cc.connect4Pass2();

rfPrettyPrint << endl

<< "Pass 2" << endl;

cc.imgReformat(rfPrettyPrint);

rfPrettyPrint << endl

<< "Equivalency Array after: Pass 2" << endl;

cc.printEQAry(rfPrettyPrint);

}

if (connectedness == 8)

{

cc.connect8Pass1();

rfPrettyPrint << "Pass 1" << endl;

cc.imgReformat(rfPrettyPrint);

rfPrettyPrint << endl

<< "Equivalency Array after: Pass 1" << endl;

cc.printEQAry(rfPrettyPrint);

cc.connect8Pass2();

rfPrettyPrint << endl

<< "Pass 2" << endl;

cc.imgReformat(rfPrettyPrint);

rfPrettyPrint << endl

<< "Equivalency Array after: Pass 2" << endl;

cc.printEQAry(rfPrettyPrint);

}

//Managing EQ table

cc.manageEQAry();

rfPrettyPrint << endl

<< "Equivalency Array after: EQ Management" << endl;

cc.printEQAry(rfPrettyPrint);

//Third Pass

cc.connectPass3();

rfPrettyPrint << endl

<< "Pass 3" << endl;

cc.imgReformat(rfPrettyPrint);

rfPrettyPrint << endl

<< "Equivalency Array after: Pass 3" << endl;

cc.printEQAry(rfPrettyPrint);

//Printing final product of pass 3

cc.printImg(labelFile);

//Printing into CC Property File

cc.printCCproperty(propertyFile);

//drawing bounding box for each components

cc.drawBoxes();

rfPrettyPrint << endl

<< "Drawing Boxes" << endl;

cc.imgReformat(rfPrettyPrint);

}

else

{

cout << "ERROR: Some output files is missing or couldnt be opened." << endl;

}

}

else

{

cout << "ERROR: The input file with following name does not exists or there was problem reading it: " << inputName << endl;

}

input.close();

rfPrettyPrint.close();

labelFile.close();

propertyFile.close();

return 0;

}

**Outputs**

**8 Connectedness:**

Pass 1

25 31 1 14

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

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

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

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

. . 5 . . . . . . . . 2 2 2 2 . . 2 2 2 . . . . . 3 . . . . .

. . 5 5 5 5 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . . . . .

. 5 . 5 . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 6 6 . . .

. . 5 . 5 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 6 . . . .

. . . . 5 . 7 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 6 . . . .

. . 8 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 6 . . . .

. . 8 8 8 8 8 8 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 8 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . . . . .

. . . . 8 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . . . . .

. . . 8 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 2 . . . . . . . .

. . . . . . . . . 2 . . 2 2 . 2 2 2 2 . . . . 2 2 2 . . . . .

. . . . . . . . 2 . . . . 2 2 2 2 2 . . . . . . 2 . . . . . .

. . . . . . . 2 . . . . . . 2 2 2 . . 9 . . . . 2 2 . . . . .

. . . . . . 2 . . . . . . . . 2 . . . . 9 . . . . . . . . . .

. . . . . . . . . . . . 10 . . 2 . . . . . . . . . . . . . . .

. . . 11 . . . . . . . . 10 . . 2 . . . . . . . . . . . . . . .

. . 11 11 11 . . . . . . . 10 . . 2 . . . . . . . . . . . . . . .

. . 11 11 11 . . . . 12 12 10 10 10 2 2 2 . . . . . . 13 . 14 . . . . .

. . . 11 . . . . . . . . . . 2 2 2 . . . . . . 13 13 . . . . . .

. . . . . . . . . . . . . 2 2 2 2 2 . . . . . 13 . . . . . . .

Equivalency Array after: Pass 1

1 1

2 2

3 3

4 3

5 5

6 2

7 7

8 2

9 9

10 2

11 11

12 10

13 13

14 13

Pass 2

25 31 1 13

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

. . 1 . . . . . . . . . . . 2 2 2 . . . . . . . . . . 3 3 . .

. . . 1 1 1 . . . . . . . 2 2 2 2 2 . . . . . . . . 3 3 . . .

. . . . 1 1 . . . . . . 2 2 2 2 2 2 2 . . . . . . 3 3 . . . .

. . 5 . . . . . . . . 2 2 2 2 . . 2 2 2 . . . . . 3 . . . . .

. . 5 5 5 5 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . . . . .

. 5 . 5 . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 5 . 5 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 . . . .

. . . . 5 . 7 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 . . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 . . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 8 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . . . . .

. . . . 8 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . . . . .

. . . 8 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 2 . . . . . . . .

. . . . . . . . . 2 . . 2 2 . 2 2 2 2 . . . . 2 2 2 . . . . .

. . . . . . . . 2 . . . . 2 2 2 2 2 . . . . . . 2 . . . . . .

. . . . . . . 2 . . . . . . 2 2 2 . . 9 . . . . 2 2 . . . . .

. . . . . . 2 . . . . . . . . 2 . . . . 9 . . . . . . . . . .

. . . . . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . . 11 . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 11 11 11 . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 11 11 11 . . . . 2 2 2 2 2 2 2 2 . . . . . . 13 . 13 . . . . .

. . . 11 . . . . . . . . . . 2 2 2 . . . . . . 13 13 . . . . . .

. . . . . . . . . . . . . 2 2 2 2 2 . . . . . 13 . . . . . . .

Equivalency Array after: Pass 2

1 1

2 2

3 3

4 3

5 5

6 2

7 7

8 2

9 9

10 2

11 11

12 10

13 13

14 13

Equivalency Array after: EQ Management

1 1

2 2

3 3

4 3

5 4

6 2

7 5

8 2

9 6

10 2

11 7

12 2

13 8

14 8

Pass 3

25 31 0 8

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

. . 1 . . . . . . . . . . . 2 2 2 . . . . . . . . . . 3 3 . .

. . . 1 1 1 . . . . . . . 2 2 2 2 2 . . . . . . . . 3 3 . . .

. . . . 1 1 . . . . . . 2 2 2 2 2 2 2 . . . . . . 3 3 . . . .

. . 4 . . . . . . . . 2 2 2 2 . . 2 2 2 . . . . . 3 . . . . .

. . 4 4 4 4 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . . . . .

. 4 . 4 . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 4 . 4 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 . . . .

. . . . 4 . 5 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 . . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 . . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 2 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . . . . .

. . . . 2 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . . . . .

. . . 2 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 2 . . . . . . . .

. . . . . . . . . 2 . . 2 2 . 2 2 2 2 . . . . 2 2 2 . . . . .

. . . . . . . . 2 . . . . 2 2 2 2 2 . . . . . . 2 . . . . . .

. . . . . . . 2 . . . . . . 2 2 2 . . 6 . . . . 2 2 . . . . .

. . . . . . 2 . . . . . . . . 2 . . . . 6 . . . . . . . . . .

. . . . . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . . 7 . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 7 7 7 . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 7 7 7 . . . . 2 2 2 2 2 2 2 2 . . . . . . 8 . 8 . . . . .

. . . 7 . . . . . . . . . . 2 2 2 . . . . . . 8 8 . . . . . .

. . . . . . . . . . . . . 2 2 2 2 2 . . . . . 8 . . . . . . .

Equivalency Array after: Pass 3

1 1

2 2

3 3

4 3

5 4

6 2

7 5

8 2

9 6

10 2

11 7

12 2

13 8

14 8

Drawing Boxes

25 31 0 8

1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 .

1 1 2 . . 1 . . . . . . . . . 2 . . . . . . . . . 3 . 2 . 3 .

1 . 2 . . 1 . . . . . . . . 2 2 2 . . . . . . . . 3 . 2 3 3 .

1 . 2 1 1 1 . . . . . . . 2 2 2 2 2 . . . . . . . 3 3 2 . 3 .

1 1 2 1 1 1 . . . . . . 2 2 2 2 2 2 2 . . . . . . 3 3 2 . 3 .

. 4 4 4 4 4 . . . . . 2 2 2 2 . . 2 2 2 . . . . . 3 3 3 3 3 .

. 4 2 4 4 4 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . 2 . . .

. 4 2 4 . 4 . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. 4 2 . 4 4 . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 2 . . .

. 4 4 4 4 4 5 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 2 . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . 2 . . 2 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . 2 . . .

. . 2 . 2 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . 2 . . .

. . 2 2 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 2 . . . . 2 . . .

. . 2 . . . . . . 2 . . 2 2 . 2 2 2 2 . . . . 2 2 2 . 2 . . .

. . 2 . . . . . 2 . . . . 2 2 2 2 2 . . . . . . 2 . . 2 . . .

. . 2 . . . . 2 . . . . . . 2 2 2 . . 6 6 . . . 2 2 . 2 . . .

. . 2 . . . 2 . . . . . . . . 2 . . . 6 6 . . . . . . 2 . . .

. . 2 . . . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 7 7 7 . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 7 7 7 . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 7 7 7 . . . . 2 2 2 2 2 2 2 2 . . . . . . 8 8 8 . 2 . . .

. . 7 7 7 . . . . . . . . . 2 2 2 . . . . . . 8 8 8 . 2 . . .

. . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 8 8 8 2 2 . . .

Property File

25 31 0 8

8

---

1

9

0 0

4 5

---

2

193

0 2

24 27

---

3

9

0 25

5 29

---

4

10

5 1

9 5

---

5

1

9 6

9 6

---

6

2

17 19

18 20

---

7

8

20 2

23 4

---

8

5

22 23

24 25

---

Label File

25 31 0 8

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0

1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0

0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 3 3 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0 0 3 3 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0 3 3 0 0 0 0

0 0 4 0 0 0 0 0 0 0 0 2 2 2 2 0 0 2 2 2 0 0 0 0 0 3 0 0 0 0 0

0 0 4 4 4 4 0 0 0 0 2 2 2 2 2 0 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0

0 4 0 4 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 2 0 0 0

0 0 4 0 4 0 0 0 0 2 2 0 2 2 0 0 2 2 2 0 2 2 0 0 0 0 2 0 0 0 0

0 0 0 0 4 0 5 0 0 2 2 2 2 2 0 0 2 2 0 2 2 2 0 0 0 0 2 0 0 0 0

0 0 2 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 0 0 0 0

0 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0

0 0 0 0 0 2 0 0 0 2 2 2 2 2 2 2 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0

0 0 0 0 2 0 0 0 0 2 2 2 2 0 2 2 2 0 2 2 2 2 0 0 0 0 0 0 0 0 0

0 0 0 2 0 0 0 0 0 0 2 2 2 2 0 2 2 2 2 2 2 0 2 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 0 0 2 2 0 2 2 2 2 0 0 0 0 2 2 2 0 0 0 0 0

0 0 0 0 0 0 0 0 2 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 2 0 0 0 0 0 0

0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 2 2 0 0 6 0 0 0 0 2 2 0 0 0 0 0

0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 2 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 7 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 7 7 7 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 7 7 7 0 0 0 0 2 2 2 2 2 2 2 2 0 0 0 0 0 0 8 0 8 0 0 0 0 0

0 0 0 7 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 8 8 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 8 0 0 0 0 0 0 0

**4 Connectedness:**

Pass 1

25 31 1 40

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

. . 4 . . . . . . . . . . . 5 2 2 . . . . . . . . . . 6 6 . .

. . . 7 7 7 . . . . . . . 8 5 2 2 2 . . . . . . . . 9 6 . . .

. . . . 7 7 . . . . . . 10 8 5 2 2 2 2 . . . . . . 11 9 . . . .

. . 12 . . . . . . . . 13 10 8 5 . . 2 2 2 . . . . . 11 . . . . .

. . 12 12 12 12 . . . . 14 13 10 8 5 . 15 2 2 2 2 . . . . . . . . . .

. 16 . 12 . . . . . 17 14 13 10 8 5 5 5 2 2 2 2 2 . . . . 18 18 . . .

. . 19 . 20 . . . . 17 14 . 10 8 . . 5 2 2 . 2 2 . . . . 18 . . . .

. . . . 20 . 21 . . 17 14 14 10 8 . . 5 2 . 22 2 2 . . . . 18 . . . .

. . 23 . . . . . . 17 14 14 10 8 8 8 5 2 2 2 2 2 . . . . 18 . . . .

. . 23 23 23 23 23 23 23 17 14 . 10 8 8 8 5 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 23 . . . 17 14 14 10 8 8 8 . . 2 2 2 2 . . . . . . . . .

. . . . 24 . . . . 17 14 14 10 . 8 8 8 . 2 2 2 2 . . . . . . . . .

. . . 25 . . . . . . 14 14 10 10 . 8 8 8 2 2 2 . 26 . . . . . . . .

. . . . . . . . . 27 . . 10 10 . 8 8 8 2 . . . . 28 28 28 . . . . .

. . . . . . . . 29 . . . . 10 10 8 8 8 . . . . . . 28 . . . . . .

. . . . . . . 30 . . . . . . 10 8 8 . . 31 . . . . 28 28 . . . . .

. . . . . . 32 . . . . . . . . 8 . . . . 33 . . . . . . . . . .

. . . . . . . . . . . . 34 . . 8 . . . . . . . . . . . . . . .

. . . 35 . . . . . . . . 34 . . 8 . . . . . . . . . . . . . . .

. . 36 35 35 . . . . . . . 34 . . 8 . . . . . . . . . . . . . . .

. . 36 35 35 . . . . 37 37 37 34 34 34 8 8 . . . . . . 38 . 39 . . . . .

. . . 35 . . . . . . . . . . 34 8 8 . . . . . . 38 38 . . . . . .

. . . . . . . . . . . . . 40 34 8 8 8 . . . . . 38 . . . . . . .

Equivalency Array after: Pass 1

1 1

2 2

3 3

4 4

5 2

6 6

7 7

8 8

9 9

10 8

11 9

12 12

13 10

14 10

15 5

16 16

17 14

18 2

19 19

20 20

21 21

22 2

23 17

24 24

25 25

26 26

27 27

28 28

29 29

30 30

31 31

32 32

33 33

34 8

35 35

36 35

37 34

38 38

39 39

40 34

Pass 2

25 31 1 39

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

. . 4 . . . . . . . . . . . 2 2 2 . . . . . . . . . . 6 6 . .

. . . 7 7 7 . . . . . . . 2 2 2 2 2 . . . . . . . . 6 6 . . .

. . . . 7 7 . . . . . . 2 2 2 2 2 2 2 . . . . . . 9 9 . . . .

. . 12 . . . . . . . . 2 2 2 2 . . 2 2 2 . . . . . 11 . . . . .

. . 12 12 12 12 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . . . . .

. 16 . 12 . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 18 . . .

. . 19 . 20 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 . . . .

. . . . 20 . 21 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 . . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 . . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 23 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . . . . .

. . . . 24 . . . . 8 8 8 8 . 2 2 2 . 2 2 2 2 . . . . . . . . .

. . . 25 . . . . . . 8 8 8 8 . 2 2 2 2 2 2 . 26 . . . . . . . .

. . . . . . . . . 27 . . 8 8 . 2 2 2 2 . . . . 28 28 28 . . . . .

. . . . . . . . 29 . . . . 8 8 8 8 8 . . . . . . 28 . . . . . .

. . . . . . . 30 . . . . . . 8 8 8 . . 31 . . . . 28 28 . . . . .

. . . . . . 32 . . . . . . . . 8 . . . . 33 . . . . . . . . . .

. . . . . . . . . . . . 8 . . 8 . . . . . . . . . . . . . . .

. . . 35 . . . . . . . . 8 . . 8 . . . . . . . . . . . . . . .

. . 35 35 35 . . . . . . . 8 . . 8 . . . . . . . . . . . . . . .

. . 35 35 35 . . . . 8 8 8 8 8 8 8 8 . . . . . . 38 . 39 . . . . .

. . . 35 . . . . . . . . . . 8 8 8 . . . . . . 38 38 . . . . . .

. . . . . . . . . . . . . 8 8 8 8 8 . . . . . 38 . . . . . . .

Equivalency Array after: Pass 2

1 1

2 2

3 3

4 4

5 2

6 6

7 7

8 2

9 6

10 8

11 9

12 12

13 10

14 10

15 5

16 16

17 14

18 2

19 19

20 20

21 21

22 2

23 2

24 24

25 25

26 26

27 27

28 28

29 29

30 30

31 31

32 32

33 33

34 8

35 35

36 35

37 34

38 38

39 39

40 34

Equivalency Array after: EQ Management

1 1

2 2

3 3

4 4

5 2

6 5

7 6

8 2

9 5

10 2

11 5

12 7

13 2

14 2

15 2

16 8

17 2

18 2

19 9

20 10

21 11

22 2

23 2

24 12

25 13

26 14

27 15

28 16

29 17

30 18

31 19

32 20

33 21

34 2

35 22

36 22

37 2

38 23

39 24

40 2

Pass 3

25 31 0 24

1 . . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

1 1 . . . . . . . . . . . . . 2 . . . . . . . . . . . . . 3 .

. . 4 . . . . . . . . . . . 2 2 2 . . . . . . . . . . 5 5 . .

. . . 6 6 6 . . . . . . . 2 2 2 2 2 . . . . . . . . 5 5 . . .

. . . . 6 6 . . . . . . 2 2 2 2 2 2 2 . . . . . . 5 5 . . . .

. . 7 . . . . . . . . 2 2 2 2 . . 2 2 2 . . . . . 5 . . . . .

. . 7 7 7 7 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . . . . .

. 8 . 7 . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 9 . 10 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 . . . .

. . . . 10 . 11 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 . . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 . . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . . . . 2 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . . . . .

. . . . 12 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . . . . .

. . . 13 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 14 . . . . . . . .

. . . . . . . . . 15 . . 2 2 . 2 2 2 2 . . . . 16 16 16 . . . . .

. . . . . . . . 17 . . . . 2 2 2 2 2 . . . . . . 16 . . . . . .

. . . . . . . 18 . . . . . . 2 2 2 . . 19 . . . . 16 16 . . . . .

. . . . . . 20 . . . . . . . . 2 . . . . 21 . . . . . . . . . .

. . . . . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . . 22 . . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 22 22 22 . . . . . . . 2 . . 2 . . . . . . . . . . . . . . .

. . 22 22 22 . . . . 2 2 2 2 2 2 2 2 . . . . . . 23 . 24 . . . . .

. . . 22 . . . . . . . . . . 2 2 2 . . . . . . 23 23 . . . . . .

. . . . . . . . . . . . . 2 2 2 2 2 . . . . . 23 . . . . . . .

Equivalency Array after: Pass 3

1 1

2 2

3 3

4 4

5 2

6 5

7 6

8 2

9 5

10 2

11 5

12 7

13 2

14 2

15 2

16 8

17 2

18 2

19 9

20 10

21 11

22 2

23 2

24 12

25 13

26 14

27 15

28 16

29 17

30 18

31 19

32 20

33 21

34 2

35 22

36 22

37 2

38 23

39 24

40 2

Drawing Boxes

25 31 0 24

1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . 3 .

1 1 2 . . . . . . . . . . . . 2 . . . . . . . . . . . 2 . 3 .

. . 4 . . . . . . . . . . . 2 2 2 . . . . . . . . 5 5 5 5 . .

. . 2 6 6 6 . . . . . . . 2 2 2 2 2 . . . . . . . 5 5 2 5 . .

. . 2 6 6 6 . . . . . . 2 2 2 2 2 2 2 . . . . . . 5 5 2 5 . .

. . 7 7 7 7 . . . . . 2 2 2 2 . . 2 2 2 . . . . . 5 5 5 5 . .

. . 7 7 7 7 . . . . 2 2 2 2 2 . 2 2 2 2 2 . . . . . . 2 . . .

. 8 7 7 7 7 . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 9 . 10 . . . . 2 2 . 2 2 . . 2 2 2 . 2 2 . . . . 2 2 . . .

. . 2 . 10 . 11 . . 2 2 2 2 2 . . 2 2 . 2 2 2 . . . . 2 2 . . .

. . 2 . . . . . . 2 2 2 2 2 2 2 2 2 2 2 2 2 . . . . 2 2 . . .

. . 2 2 2 2 2 2 2 2 2 . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 . . .

. . 2 . . 2 . . . 2 2 2 2 2 2 2 . . 2 2 2 2 . . . . . 2 . . .

. . 2 . 12 . . . . 2 2 2 2 . 2 2 2 . 2 2 2 2 . . . . . 2 . . .

. . 2 13 . . . . . . 2 2 2 2 . 2 2 2 2 2 2 . 14 . . . . 2 . . .

. . 2 . . . . . . 15 . . 2 2 . 2 2 2 2 . . . . 16 16 16 . 2 . . .

. . 2 . . . . . 17 . . . . 2 2 2 2 2 . . . . . 16 16 16 . 2 . . .

. . 2 . . . . 18 . . . . . . 2 2 2 . . 19 . . . 16 16 16 . 2 . . .

. . 2 . . . 20 . . . . . . . . 2 . . . . 21 . . . . . . 2 . . .

. . 2 . . . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 22 22 22 . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 22 22 22 . . . . . . . 2 . . 2 . . . . . . . . . . . 2 . . .

. . 22 22 22 . . . . 2 2 2 2 2 2 2 2 . . . . . . 23 23 24 . 2 . . .

. . 22 22 22 . . . . . . . . . 2 2 2 . . . . . . 23 23 . . 2 . . .

. . 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 23 23 2 2 2 . . .

Label File

25 31 0 24

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0

1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0

0 0 4 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 5 5 0 0

0 0 0 6 6 6 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0 0 5 5 0 0 0

0 0 0 0 6 6 0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0 5 5 0 0 0 0

0 0 7 0 0 0 0 0 0 0 0 2 2 2 2 0 0 2 2 2 0 0 0 0 0 5 0 0 0 0 0

0 0 7 7 7 7 0 0 0 0 2 2 2 2 2 0 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0

0 8 0 7 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 2 0 0 0

0 0 9 0 10 0 0 0 0 2 2 0 2 2 0 0 2 2 2 0 2 2 0 0 0 0 2 0 0 0 0

0 0 0 0 10 0 11 0 0 2 2 2 2 2 0 0 2 2 0 2 2 2 0 0 0 0 2 0 0 0 0

0 0 2 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 0 0 0 0

0 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0

0 0 0 0 0 2 0 0 0 2 2 2 2 2 2 2 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0

0 0 0 0 12 0 0 0 0 2 2 2 2 0 2 2 2 0 2 2 2 2 0 0 0 0 0 0 0 0 0

0 0 0 13 0 0 0 0 0 0 2 2 2 2 0 2 2 2 2 2 2 0 14 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 15 0 0 2 2 0 2 2 2 2 0 0 0 0 16 16 16 0 0 0 0 0

0 0 0 0 0 0 0 0 17 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 16 0 0 0 0 0 0

0 0 0 0 0 0 0 18 0 0 0 0 0 0 2 2 2 0 0 19 0 0 0 0 16 16 0 0 0 0 0

0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 2 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 22 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 22 22 22 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 22 22 22 0 0 0 0 2 2 2 2 2 2 2 2 0 0 0 0 0 0 23 0 24 0 0 0 0 0

0 0 0 22 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 23 23 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 23 0 0 0 0 0 0 0

Property File

25 31 0 24

24

---

1

3

0 0

1 1

---

2

180

0 2

24 27

---

3

2

0 29

1 29

---

4

1

2 2

2 2

---

5

7

2 25

5 28

---

6

5

3 3

4 5

---

7

6

5 2

7 5

---

8

1

7 1

7 1

---

9

1

8 2

8 2

---

10

2

8 4

9 4

---

11

1

9 6

9 6

---

12

1

13 4

13 4

---

13

1

14 3

14 3

---

14

1

14 22

14 22

---

15

1

15 9

15 9

---

16

6

15 23

17 25

---

17

1

16 8

16 8

---

18

1

17 7

17 7

---

19

1

17 19

17 19

---

20

1

18 6

18 6

---

21

1

18 20

18 20

---

22

8

20 2

23 4

---

23

4

22 23

24 24

---

24

1

22 25

22 25

---