CV

Project 4: Connected Components

C++

Student: Trisha Espejo
Project Due Date: 3/21/2020

Algorithm for Pass 1 Connect Component

```
Step 0:  for (int r = 1; r < numRows + 1; r++){
                 for (int c = 1; c < numCols + 1; c++){
                        if (inAry[r][c] > 0) Pass1( inAry, r ,c)
                 }
         }
```

Step 1: row = r -1 and col = c -1, temp = 0, min = 0, max = 0, index = 0
Bool isAllzero = true and isequal = false;
Step 2: you place your template over to the inAry. Then from 0 to the last element before P(I,j) you add all the value into NonzeroNeighborAry.
Step 3: iterate to all the values of nonZeroNeighborAry and evalulate if each element in nonZeroNeighborAry != 0 if it is let isAllzero = false
Step 4: create a 2 for loops the outer for loop i starts at 0 to 3 and inner for j loop goes from 1 to 4.
Here let a = nonzeroAry[i] and b = nonzero[j]. If both a and b is not equal to zero and a and b is not equal isEqual is false. ELSE isEqual = true if a = 0 then temp = b; else if b = 0 let temp = a, else if a and b is not 0 let temp = a.
Step 5: if iszero then case 1 true so newLabel + 1 then inAry[i][j] = newLabel
Else if isEqual = true zeroFrameAry = temp;
Step 6: else if isequal is false you perfrom another 2 for loop create a 2 for loops the outer for loop i starts at 0 to 3 and inner for j loop goes from 1 to 4.
If a < b && a and b is not 0 let min = a and max = b. else min = b and max = a.
Step 7: inAry[i][j] = min and
Step 8: then updateEQ table EQAry[max] = min;

Algorithm for Pass 2 Connect Component

```
Step 0:  for (int r = numRows; r > 0 ; r--){
                 for (int c = numCols; c > 0; c--){
                        if (inAry[r][c] > 0) Pass2( inAry, r ,c)
                 }
         }
```

Step 1: row = r -1 and col = c -1, index = 0 i = 1 and j = 1 Bool isAllzero = true and isequal = false;
Step 2: outer while loop while I < 3 and inner for loop while j < 3 you place the value of inAry to NonZeroAry
When j <3 is done with iterating set j = 0
Step 3: iterate to all the values of nonZeroNeighborAry and evalulate if each element in nonZeroNeighborAry != 0 if it is let isAllzero = false
Step 4: create a 2 for loops the outer for loop i starts at 0 to 3 and inner for j loop goes from 1 to 4.
Here let a = nonzeroAry[i] and b = nonzero[j]. If both a and b is not equal to zero and a and b is not equal isEqual is false.
Step 5: if isAllZero is true or isEqual is true inAry[r][c] = onAry[r][c] (it does notting, to make sure that keeps its value I just return the same element
Step 6: else if isEqual is false. minLabel = nonzeroNeighbor[0]. Thwn have for loop that iterate from i = 1 to 5
For each iteration if nozeroNeighAry != 0 minLabel = minimum minLabel and nonzeroNeighbor[i]
Step 7: if current array is greater than minlabel then update the EQtable with minLabel.
Step 8 : inAry[r][c] = minLabel

Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>

using namespace std;
struct Property {
    int label = 0;
    int numPixel = 0;
    int minR = 9999;
    int minC = 9999;
    int maxR = 0;
    int maxC = 0;
};
class CCLabel {
    public:
    int numRows = -1;
    int numCols= -1;
    int minVal = -1;
    int maxVal = -1;
    int newMin = -1;
    int newMax = -1;
    int newLabel = 0;
    int trueNumCC = 0;
    int **zeroFramedAry;
    int NonZeroNeighborAry[5];
    int *EQAry;
    struct Property *CCProperty;

public:

    void constructor(ifstream & input){
        input >> this->numRows >> this->numCols >> this->minVal >>
this->maxVal;
        newMin = minVal;
        newMax = maxVal;
        newLabel = 0;
        this -> zeroFramedAry = new int*[this -> numRows + 2];
        for (int i = 0; i < this -> numRows + 2; ++i){
            zeroFramedAry[i] = new int[ this -> numCols + 2];
        }

        for (int i = 0; i < 5; ++i){
            NonZeroNeighborAry[i] = -1;

        }
        this -> EQAry = new  int[(this->numRows * this -> numCols)/
4];
```

```
        for (int i = 0; i < (this->numRows * this-> numCols)/ 4 ;
++i){
            EQAry[i] = i;

        }
    }
    void zero2D (){
        for (int i = 0; i < this -> numRows + 2; ++i){
            for (int j = 0; j < this -> numCols + 2; ++j)
                zeroFramedAry[i][j] = 0;
        }
    }
    void loadImage (ifstream & input){

        for (int i = 1; i < this -> numRows + 1 ; ++i){
            for (int j = 1; j < this -> numCols + 1 ; ++j)
            {
                input >> zeroFramedAry[i][j];

            }
        }

    }
    void connect4Pass1(){
        newMin = 9999;
        newMax = 0;
        for (int r = 1; r < numRows +  1; r++){
            for (int c = 1; c < numCols + 1; c++){
                if ( zeroFramedAry[r][c] > 0)
                    pass1Connect4(zeroFramedAry, r, c);
                if (newMin > zeroFramedAry[r][c])
                    newMin = zeroFramedAry[r][c];
                if (newMax < zeroFramedAry[r][c])
                    newMax = zeroFramedAry[r][c];

            }
        }

    }

    void pass1Connect4(int **zeroFramedAry, int r, int c){
        int a = zeroFramedAry[r - 1][c];
        int b = zeroFramedAry [r][c - 1];
        int min = 0;
        int max = 0;
        //case 1 : a = b = 0
        if ( a == 0 && b == 0) {
            newLabel++;
            zeroFramedAry[r][c] = newLabel;
        }
        //case 2: a == b or a or b already has label
        else if( a == b && a != 0 && b != 0)
            zeroFramedAry[r][c] = a;
        else if ( a != 0 && b == 0)
            zeroFramedAry[r][c] = a;
        else if (a == 0 && b != 0)
            zeroFramedAry[r][c] = b;
        //case 3 a != b and a and b has label
        else
        {
            if (a < b){
                min =  a;
                max = b;
            }
            else
            {
                min = b;
                max = a;
            }
            zeroFramedAry[r][c] = min;
            updateEQ(max, min);
        }
    }
    void connect4Pass2(){
        newMin = 9999;
        newMax = 0;
        for (int r = numRows; r > 0; r--){
            for (int c = numCols; c > 0; c--){
                if ( zeroFramedAry[r][c] > 0)
                    pass2Connect4(zeroFramedAry, r, c);
                if (newMin > zeroFramedAry[r][c])
                    newMin = zeroFramedAry[r][c];
                if (newMax < zeroFramedAry[r][c])
                    newMax = zeroFramedAry[r][c];
            }
        }
     }
    void pass2Connect4 (int **zeroFramedAry, int i, int j){
        int c = zeroFramedAry[i][j + 1];
        int d = zeroFramedAry[i + 1][j];
        int x = zeroFramedAry[i][j];
        int minLabel = 0;
        //case 1 : a = b = 0
        if ( c == 0 && d == 0) {
            zeroFramedAry[i][j] = zeroFramedAry[i][j];

        }
        //case 2: a == b or a or b already has label
        else if( c == d == x && c != 0 && d != 0)
        {
            zeroFramedAry[i][j] = zeroFramedAry[i][j];
        }
        else if ( c != 0 && d == 0 && c == x){
            zeroFramedAry[i][j] = zeroFramedAry[i][j];

        }
        else if (c == 0 && d != 0 && d == x){
            zeroFramedAry[i][j] = zeroFramedAry[i][j];

        }
        //case 3 a != b and a and b has label
        else
        {
            if (c == 0) minLabel = min( d, x);
            else if (d == 0) minLabel =  min( c, x);
            else {
                minLabel = min(c, d);
                minLabel = min (minLabel, x);
            }
            if ( x > minLabel)
                updateEQ(x, minLabel);


            zeroFramedAry[i][j] = minLabel;
        }
    }
    void connect8Pass1() {
        newMin = 9999;
        newMax = 0;
        for (int r = 1; r < numRows + 1; r++){
            for (int c = 1; c < numCols + 1; c++){
                if ( zeroFramedAry[r][c] > 0)
                    pass1Connect8(zeroFramedAry, r, c);
                if (newMin > zeroFramedAry[r][c])
                    newMin = zeroFramedAry[r][c];
                if (newMax < zeroFramedAry[r][c])
                    newMax = zeroFramedAry[r][c];
            }
        }
    }
    void pass1Connect8(int **zeroFramedAry, int r, int c){
        int row = r - 1;
        int col = c - 1;
        int temp = 0;
        int min =  0;
        int max = 0;
        int index = 0;
        bool stop = false;
        bool isAllZero = true;
        bool isEqual = false;
        for (int i = 0; i < 2 && !stop; i++){
            for (int j = 0; j < 3 && !stop; j++){
                //cout << j << " " << j;
                if ( i == 1 && j == 1) stop = true;
                else{
                    NonZeroNeighborAry[index] = zeroFramedAry[i +
row][j + col];
                    index++;
                }
            }
        }
        for (int i = 0; i < 4; i++){
            if (NonZeroNeighborAry[i] != 0) {
                isAllZero = false;
            }
        }
        //cout << count;
        for (int i = 0; i < 3; i++){
            for (int j = 1; j < 4; j++){
                int a = NonZeroNeighborAry[i];
                int b = NonZeroNeighborAry[j];
                if(a != 0 && b != 0 && a != b){
                    isEqual = false;
                }
                else
                {
                    isEqual = true;
                    if ( a == 0 && b != 0)
                        temp = b;
                    else if ( b == 0 && a != 0)
                        temp = a;
                    else if (a == b && a != 0 && b != 0 )
                        temp = a;
                }
            }
        }
        // case 1
        if (isAllZero)
        {
            newLabel++;
            zeroFramedAry[r][c] = newLabel;
        }
        //case 2
        else if (isEqual){
            zeroFramedAry[r][c] = temp;
        }
        // case 3
        else {
            for (int i = 0; i < 3; i++){
                for (int j = 1; j < 4; j++){
```

```
                int a = NonZeroNeighborAry[i];
                int b = NonZeroNeighborAry[j];
                if (a < b && a != 0 && b != 0)
                {
                    min = a;
                    max = b;
                }
                else
                {
                    min = b;
                    max = a;
                }
            }
        }
        zeroFramedAry[r][c] = min;
        updateEQ(max, min);
    }
}
void connect8Pass2() {
    newMin = 9999;
    newMax = 0;
    for (int r = numRows; r > 0; r--){
       for (int c = numCols; c > 0; c--){
            if ( zeroFramedAry[r][c] > 0)
                pass2Connect8(zeroFramedAry, r, c);
           if (newMin > zeroFramedAry[r][c])
               newMin = zeroFramedAry[r][c];
           if (newMax < zeroFramedAry[r][c])
               newMax = zeroFramedAry[r][c];
        }
    }
}
void pass2Connect8(int **Ary, int r, int c){
    int row = r - 1;
    int col = c - 1;
    int index = 0;
    bool isAllZero = true;
    bool isEqual = true;
    int minLabel = 0;
    int i = 1;
    int j = 1;
    while (i < 3)
    {
        while (j < 3 ){
            NonZeroNeighborAry[index] = Ary[i + row][j + col];
            index++;
            j++;
        }
        j = 0;
        i++;
    }
    for (int i = 1; i < 5; i++){
        if (NonZeroNeighborAry[i] != 0) {
            isAllZero = false;
        }
    }
    for (int i = 0; i < 4; i++){
        for (int j = 1; j < 5; j++){
            int a = NonZeroNeighborAry[i];
            int b = NonZeroNeighborAry[j];
            if(a != 0 && b != 0 && a != b){
                isEqual = false;
            }

        }
    }
    if(isAllZero)
        Ary[r][c] = Ary[r][c];
    else if (isEqual)
        Ary[r][c] = Ary[r][c];
    else
    {
        minLabel = NonZeroNeighborAry[0];
        for (int i = 1; i < 5; i++){
            if (NonZeroNeighborAry[i] != 0)
                minLabel = min (minLabel, NonZeroNeighborAry[i]);
        }
        if ( Ary[r][c] > minLabel)
            updateEQ(Ary[r][c], minLabel);
        Ary[r][c] = minLabel;

    }

}

void connectPass3(){
    newMin = 9999;
    newMax = 0;
    for (int r = 1; r < numRows +  1; r++){
        for (int c = 1; c < numCols + 1; c++){
            if( zeroFramedAry[r][c] > 0)
                zeroFramedAry[r][c] = EQAry[zeroFramedAry[r][c]];
            if (newMin > zeroFramedAry[r][c])
                newMin = zeroFramedAry[r][c];
            if (newMax < zeroFramedAry[r][c])
                newMax = zeroFramedAry[r][c];
        }
    }
}
void genProperty()
```

```
{
    this -> CCProperty = new struct Property[this->trueNumCC + 1];
    int x = 0;

    for (int a = 1; a < trueNumCC + 1; a++){
        CCProperty[a].label =  a;
        for (int i = 1; i < numRows + 1; i++){
            for (int j = 1; j < numCols + 1; j++){
                x = zeroFramedAry[i][j];

                if ( x == CCProperty[a].label)
                {
                    CCProperty[a].numPixel++;
                    if (CCProperty[a].minR > i)
                        CCProperty[a].minR = i;
                    if (CCProperty[a].minC > j)
                        CCProperty[a].minC = j;
                    if (CCProperty[a].maxR < i)
                        CCProperty[a].maxR = i;
                    if (CCProperty[a].maxC < j)
                        CCProperty[a].maxC = j;
                }
            }
        }
    }
}
void updateEQ( int index, int min){
    EQAry[index] = min;

}
int manageEQAry(int *EQAry, int newLabel){
    int readLabel = 0;
    for (int i = 1; i < newLabel + 1; i++)
    {
        if (i != EQAry[i])
            EQAry[i] = EQAry[EQAry[i]];
        else {
            readLabel++;
            EQAry[i] = readLabel;
            }

    }
    return readLabel;
}
void drawBoxes (){
    int index = 1;

    while (index < trueNumCC + 1)
    {
            int minRow = CCProperty[index].minR;
            int minCol = CCProperty[index].minC;
            int maxRow = CCProperty[index].maxR;
            int maxCol = CCProperty[index].maxC;
            int label =  CCProperty[index].label;


            for (int i = minCol; i < maxCol + 1; i++)
            {
                zeroFramedAry[minRow][i] = label;

            }
            for (int i = minCol; i < maxCol + 1; i++)
                zeroFramedAry[maxRow][i] = label;
            for (int i = minRow; i < maxRow + 1; i++)
                zeroFramedAry[i][minCol] = label;
            for (int i = minRow; i < maxRow + 1; i++)
                zeroFramedAry[i][maxCol] = label;

            index++;
        }
    }

    void imgReformat(int **inAry, ofstream & OutImg){
        OutImg <<" " << this->numRows <<" " << this->numCols << " " <<
this->newMin << " " << this->newMax << endl;
        OutImg << endl;


        for (int r = 1; r < numRows +  1; r++){
            for (int c = 1; c < numCols + 1; c++){
                if  (0 < inAry[r][c] )
                    OutImg << inAry[r][c] << " ";

                else
                    OutImg << ". ";
            }
            OutImg << endl;
        }
        OutImg << endl;
    }
    void printEQAry (int label, ofstream & OutImg){
        for (int i = 1; i < label + 1; i++)
        {
            OutImg << i << ": " << EQAry[i] << endl;
        }
        OutImg << endl << endl;
    }

    void printImg(ofstream & OutImg){
        OutImg <<" " << this->numRows <<" " << this->numCols << " " <<
this->newMin << " " << this->newMax << endl;
        OutImg << endl;

        for (int r = 1; r < numRows +  1; r++){
            for (int c = 1; c < numCols + 1; c++){
                OutImg << setw(2) << zeroFramedAry[r][c] << " ";
            }
            OutImg << endl;
        }
        OutImg << endl;
    }

    void printCCproperty(ofstream & Output){
                Output << endl;
        Output << " " << this->numRows <<" " << this->numCols << " "
<< this->newMin << " " << this->newMax << endl;
        Output << " " << this ->trueNumCC << endl;

        Output <<
"********************************************************" <<endl;
        for (int i = 1; i < trueNumCC + 1; i++){
            Output << endl;
            Output << CCProperty[i].label << endl;
            Output << CCProperty[i].numPixel << endl;
            Output << CCProperty[i].minR << "  " << CCProperty[i].minC
<<endl;
            Output << CCProperty[i].maxR << "  " << CCProperty[i].maxC
<<endl;
            Output <<
"********************************************************" <<endl;
        }

    }


    void free_Heap (){
        for (int i = 0; i < this->numRows + 2; ++i)
            delete[] this->zeroFramedAry[i];
        delete[] this->zeroFramedAry;

        delete[] this->CCProperty;

        delete[] this-> EQAry;
    }
};

int main(int argc, const char * argv[]) {
    int connectness = atoi(argv[2]);
    string inputName = argv[1];
    ifstream input;
    input.open(inputName);

    string output1 = argv[3];
    ofstream RFprettyPrintFile;
    RFprettyPrintFile.open(output1);

    string output2 = argv[4];
    ofstream labelFile;
    labelFile.open(output2);

    string output3 = argv[5];
    ofstream propertyFile;
    propertyFile.open(output3);

    CCLabel* read_img = new CCLabel();
    read_img -> constructor(input);
    read_img -> zero2D ();
    read_img -> loadImage(input);

    if (connectness == 4){
        read_img -> connect4Pass1();
        RFprettyPrintFile << "Connect 4 Pass 1: " << endl;
        read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);
        RFprettyPrintFile << "Equivalence Table after Pass 1: " <<
endl;
        read_img -> printEQAry (read_img -> newLabel,
RFprettyPrintFile);

        read_img -> connect4Pass2();
        RFprettyPrintFile << "Connect 4 Pass 2: " << endl;
        read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);
        RFprettyPrintFile << "Equivalence Table after Pass 2: " <<
endl;
        read_img -> printEQAry (read_img -> newLabel,
RFprettyPrintFile);
    }
    if (connectness == 8){
        read_img -> connect8Pass1();
        RFprettyPrintFile << "Connect 8 Pass 1: " << endl;
        read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);
        RFprettyPrintFile << "Equivalence Table after Pass 1: " <<
endl;
        read_img -> printEQAry (read_img -> newLabel,
RFprettyPrintFile);
```

```
        read_img -> connect8Pass2();
        RFprettyPrintFile << "Connect 8 Pass 2: " << endl;
        read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);
        RFprettyPrintFile << "Equivalence Table after Pass 2: " <<
endl;
        read_img -> printEQAry (read_img -> newLabel,
RFprettyPrintFile);
    }


    read_img -> trueNumCC = read_img -> manageEQAry(read_img-> EQAry,
read_img-> newLabel);
    RFprettyPrintFile << " New Equivalence Table after ManageEQ : " <<
endl;
    read_img -> printEQAry (read_img -> newLabel, RFprettyPrintFile);
    read_img -> connectPass3();
    if (connectness == 4)
        RFprettyPrintFile << "Connect 4 Pass 3: " << endl;
    if (connectness == 8)
        RFprettyPrintFile << "Connect 8 Pass 3: " << endl;
    read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);
    RFprettyPrintFile << "Equivalence Table after Pass 3: " << endl;
    read_img -> printEQAry (read_img -> newLabel, RFprettyPrintFile);

    read_img -> printImg(labelFile);
    read_img -> genProperty();
    read_img -> printCCproperty(propertyFile);
    read_img -> drawBoxes();
    RFprettyPrintFile << "Connect 8 After drawBoxes: " << endl;
    read_img -> imgReformat(read_img -> zeroFramedAry,
RFprettyPrintFile);

    RFprettyPrintFile << "Total Number of Connected Components: " <<
endl;
    RFprettyPrintFile << "Number of CC: " << read_img -> trueNumCC <<
endl;


    read_img -> free_Heap();
    input.close();
    RFprettyPrintFile.close();
    labelFile.close();
    propertyFile.close();

}
```

Connect Component 8 Output:
RFprettyPrintFile

```
Connect 8 Pass 1:
 25 31 0 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 .  .  .  .  .  .  .

Equivalence Table 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
```

```
Connect 8 Pass 2:
 25 31 0 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 .  .  .  .  .  .  .

Equivalence Table 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: 2
13: 13
14: 13
```

```
 New Equivalence Table after ManageEQ :
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


Connect 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 . . . . . . .
```

```
Equivalence Table 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


Connect 8 After drawBoxes:
 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  .  .  .

Total Number of Connected Components:
Number of CC: 8
```

labelFile

```
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
```

propertyFile

```
 25 31 0 8
 8
********************************************************

1
9
1  1
5  6
********************************************************

2
193
1  3
25  28
********************************************************

3
9
1  26
6  30
********************************************************

4
10
6  2
10  6
********************************************************

5
1
10  7
10  7
********************************************************

6
2
18  20
19  21
********************************************************

7
8
21  3
24  5
********************************************************

8
5
23  24
25  26
********************************************************
```

Connect Component 4 Output:
RFprettyPrintFile

```
Connect 4 Pass 1:
 25 31 0 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  .  .  .  .  .  .  .
```

```
Equivalence Table after Pass 1:
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: 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
```

```
Connect 4 Pass 2:
 25 31 0 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  .  .  .  .  .  .  .
```

```
Equivalence Table after Pass 2:
1: 1
2: 2
3: 3
4: 4
5: 2
6: 6
7: 7
8: 2
9: 6
10: 2
11: 9
12: 12
13: 2
14: 2
15: 2
16: 16
17: 2
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: 8
38: 38
39: 39
40: 8
```

```
 New Equivalence Table after ManageEQ :
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
```

```
Connect 4 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  .  .  .  .  .  .  .
```

```
Equivalence Table 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
```

```
Connect 8 After drawBoxes:
 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  .  .  .

Total Number of Connected Components:
Number of CC: 24
```

labelFile

```
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
```

propertyFile

```
 25 31 0 24
 24
********************************************************

1
3
1  1
2  2
********************************************************

2
180
1  3
25  28
********************************************************

3
2
1  30
2  30
********************************************************

4
1
3  3
3  3
********************************************************

5
7
3  26
6  29
********************************************************

6
5
4  4
5  6
********************************************************

7
6
6  3
```

```
8  6
*******************************************************

8
1
8  2
8  2
*******************************************************

9
1
9  3
9  3
*******************************************************

10
2
9  5
10  5
*******************************************************

11
1
10  7
10  7
*******************************************************

12
1
14  5
14  5
*******************************************************

13
1
15  4
15  4
*******************************************************

14
1
15  23
15  23
*******************************************************
```

```
15
1
16  10
16  10
*****************************************************

16
6
16  24
18  26
*****************************************************

17
1
17  9
17  9
*****************************************************

18
1
18  8
18  8
*****************************************************

19
1
18  20
18  20
*****************************************************

20
1
19  7
19  7
*****************************************************

21
1
19  21
19  21
*****************************************************

22
8
21  3
24  5
```

```
**************************************************************

23
4
23    24
25    25
**************************************************************

24
1
23    26
23    26
**************************************************************
```