Student: Pawan Bhatta
Project Due Date: 04/10/2021

The North Algorithm:
Step 1: Scan img Left-Right & Top-Bottom

Step 2: P(i,j) ← get next pixel

Step 3: if P(i,j) > 0 :
Change P(i,j) to 0 under the following conditions:
(1) NORTH Neighbor of P(i,j) is 0
(2) P(i,j) has at least 4 object neighbors (i.e. > 0)
(3) P(i,j) is not a connector, i.e., by flipping P(i,j) to 0 will not create more than one Connected Component in P's 3x3 neighborhood.

Step 4: Repeat steps 2 to 3 until all pixels are processed

Thinning Algorithm Steps:
Step 0: img ← given Binary Image

Step 1: Thin NORTH (1 layer)

Step 2: Thin SOUTH (1 layer)

Step 3: Thin WEST (1 layer)

Step 4 Thin EAST (1 layer)

Step 5: repeat steps 1 to 4 until no more
pixels change from 1 to 0

Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdarg>
using namespace std;

class Thinning
{
public:
    int numRows;
    int numCols;
    int minVal;
    int maxVal;
    int newMin;
    int newMax;
    int rowFrameSize;
    int colFrameSize;
    int extraRows;
    int extraCols;
    int changeFlag;
    int cycleCount;
    int **aryOne;
    int **aryTwo;

    Thinning(ifstream &input)
    {
        loadHeader(input);
        rowFrameSize = 1;
        colFrameSize = 1;
        extraRows = 2 * rowFrameSize;
        extraCols = 2 * colFrameSize;
        //dynamic allocation of zeroframeArray
        aryOne = new int *[numRows + extraRows];
        aryTwo = new int *[numRows + extraRows];

        for (int i = 0; i < numRows + extraRows; i++)
        {
            aryOne[i] = new int[numCols + extraCols];
            aryTwo[i] = new int[numCols + extraCols];
        }
        zero2D(aryOne, numRows + extraRows, numCols + extraCols);
        zero2D(aryTwo, numRows + extraRows, numCols + extraCols);

        changeFlag = 1;
    }

    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 >> aryOne[i][j];
        }
    }
}

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 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, int **ary)
{
    outFile << numRows << " " << numCols << " " << minVal << " " << maxVal << 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 (ary[i][j] == 0)
            {
                outFile << "."
```

```
                        << " ";
            }
            else
            {
                outFile << ary[i][j] << " ";
            }
            str = to_string(ary[i][j]);
            int ww = str.length();
            while (ww < width)
            {
                outFile << " ";
                ww++;
            }
        }
        outFile << endl;
    }
}

void printImg(ofstream &outFile)
{
    outFile << numRows << " " << numCols << " " << minVal << " " << maxVal << 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 << aryOne[i][j] << " ";
            str = to_string(aryOne[i][j]);
            int ww = str.length();
            while (ww < width)
            {
                outFile << " ";
                ww++;
            }
        }
        outFile << endl;
    }
}

void copyArys(int **arr1, int **arr2)
{
    for (int i = 0; i < numRows + extraRows; i++)
    {
        for (int j = 0; j < numCols + extraCols; j++)
        {
            arr1[i][j] = arr2[i][j];
        }
    }
}
```

```
    bool hasXNeighbors(int i, int j, int numOfObjectNeighbors)
    {
        int a = aryOne[i - 1][j - 1];
        int b = aryOne[i - 1][j];
        int c = aryOne[i - 1][j + 1];
        int d = aryOne[i][j - 1];
        int e = aryOne[i][j + 1];
        int f = aryOne[i + 1][j - 1];
        int g = aryOne[i + 1][j];
        int h = aryOne[i + 1][j + 1];
        int sum = a + b + c + d + e + f + g + h;

        if (sum >= numOfObjectNeighbors)
        {
            return true;
        }
        return false;
    }

    bool isConnector(int i, int j)
    {
        int a = aryOne[i - 1][j - 1];
        int b = aryOne[i - 1][j];
        int c = aryOne[i - 1][j + 1];
        int d = aryOne[i][j - 1];
        int e = aryOne[i][j + 1];
        int f = aryOne[i + 1][j - 1];
        int g = aryOne[i + 1][j];
        int h = aryOne[i + 1][j + 1];

        //case 1
        if (d == 0 && e == 0 && (a == 1 || b == 1 || c == 1) && (f == 1 || g == 1 || h ==
1))
        {
            return true;
        }
        //case 2
        if (b == 0 && g == 0 && (a == 1 || d == 1 || f == 1) && (c == 1 || e == 1 || h ==
1))
        {
            return true;
        }
        //case Alpha
        if (b == 0 && d == 0 && a == 1)
        {
            return true;
        }
        //case Beta
        if (d == 0 && g == 0 && f == 1)
```

```
        {
            return true;
        }
        //case Gamma
        if (b == 0 && e == 0 && c == 1)
        {
            return true;
        }

        //case delta
        if (g == 0 && e == 0 && h == 1)
        {
            return true;
        }
        return false;
    }

    void NorthThinning(int **arr1, int **arr2)
    {
        for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)
        {
            for (int j = colFrameSize; j < numCols + colFrameSize; j++)
            {
                if (arr1[i][j] > 0)
                {
                    arr2[i][j] = 1;

                    //Must follow all 3 conditions to flip this pixel
                    //FIRST CONDITION= North neighbour is 0
                    if (arr1[i - 1][j] == 0 && hasXNeighbors(i, j, 4) && !isConnector(i,
j))
                    {
                        arr2[i][j] = 0;
                        changeFlag++;
                    }
                }
            }
        }
    }

    void SouthThinning(int **arr1, int **arr2)
    {
        for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)
        {
            for (int j = colFrameSize; j < numCols + colFrameSize; j++)
            {
                if (arr1[i][j] > 0)
                {
                    arr2[i][j] = 1;
```

```
                    //Must follow all 3 conditions to flip this pixel
                    //FIRST CONDITION= South neighbour is 0
                    if (arr1[i + 1][j] == 0 && hasXNeighbors(i, j, 4) && !isConnector(i,
j))
                    {

                        arr2[i][j] = 0;
                        changeFlag++;
                    }
                }
            }
        }
    }

    void EastThinning(int **arr1, int **arr2)
    {
        for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)
        {
            for (int j = colFrameSize; j < numCols + colFrameSize; j++)
            {
                if (arr1[i][j] > 0)
                {
                    arr2[i][j] = 1;

                    //Must follow all 3 conditions to flip this pixel
                    //FIRST CONDITION= East neighbour is 0
                    if (arr1[i][j + 1] == 0 && hasXNeighbors(i, j, 3) && !isConnector(i,
j))
                    {

                        arr2[i][j] = 0;
                        changeFlag++;
                    }
                }
            }
        }
    }

    void WestThinning(int **arr1, int **arr2)
    {
        for (int i = rowFrameSize; i < numRows + rowFrameSize; i++)
        {
            for (int j = colFrameSize; j < numCols + colFrameSize; j++)
            {
                if (arr1[i][j] > 0)
                {
                    arr2[i][j] = 1;

                    //Must follow all 3 conditions to flip this pixel
                    //FIRST CONDITION= West neighbour is 0
```

```
                    if (arr1[i][j - 1] == 0 && hasXNeighbors(i, j, 3) && !isConnector(i,
j))
                    {

                        arr2[i][j] = 0;
                        changeFlag++;
                    }
                }
            }
        }
    }

    ~Thinning()
    {
        for (int i = 0; i < numRows + extraRows; i++)
        {
            delete[] aryOne[i];
            delete[] aryTwo[i];
        }
    }
};

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

    //WRITES
    string thiningOutputName = argv[2], prettyPrintName = argv[3];
    ofstream thinningOutput, rfPrettyPrint;
    rfPrettyPrint.open(prettyPrintName);
    thinningOutput.open(thiningOutputName);

    //Checking if IO operations succeeds
    if (input.is_open())
    {
        if (rfPrettyPrint.is_open() && thinningOutput.is_open())
        {
            Thinning t(input);
            t.loadImage(input);
            t.cycleCount = 0;
            rfPrettyPrint << "Original Image" << endl;
            t.imgReformat(rfPrettyPrint, t.aryOne);

            while (t.changeFlag > 0)
            {
                t.changeFlag = 0;
                t.NorthThinning(t.aryOne, t.aryTwo);
```

```
                t.copyArys(t.aryOne, t.aryTwo);
                t.SouthThinning(t.aryOne, t.aryTwo);
                t.copyArys(t.aryOne, t.aryTwo);
                t.WestThinning(t.aryOne, t.aryTwo);
                t.copyArys(t.aryOne, t.aryTwo);
                t.EastThinning(t.aryOne, t.aryTwo);
                t.copyArys(t.aryOne, t.aryTwo);
                t.cycleCount++;
                rfPrettyPrint << "\nResult of Thinning (N,S = 4): Cycle - " << t.cycleCount
<< endl;
                t.imgReformat(rfPrettyPrint, t.aryOne);
            }
            t.printImg(thinningOutput);
        }
        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();
    thinningOutput.close();
    return 0;
}
```

Outputs

Data_1:

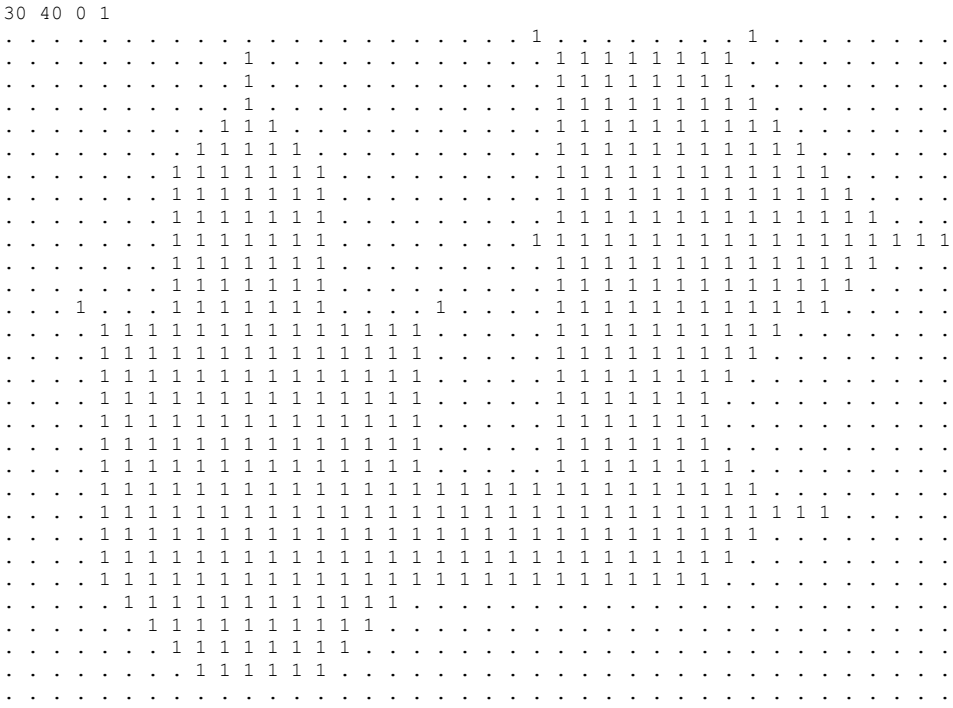
Original Image

```
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 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 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 1 1 1 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 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 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 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 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 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 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 1 0
0 0 0 0 0 0 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 1 1 1
0 0 0 0 0 0 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 1 0
0 0 0 0 0 0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 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 1 1 1 1 1 0 0 0 1 1 1 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 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 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 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 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 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 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 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
```

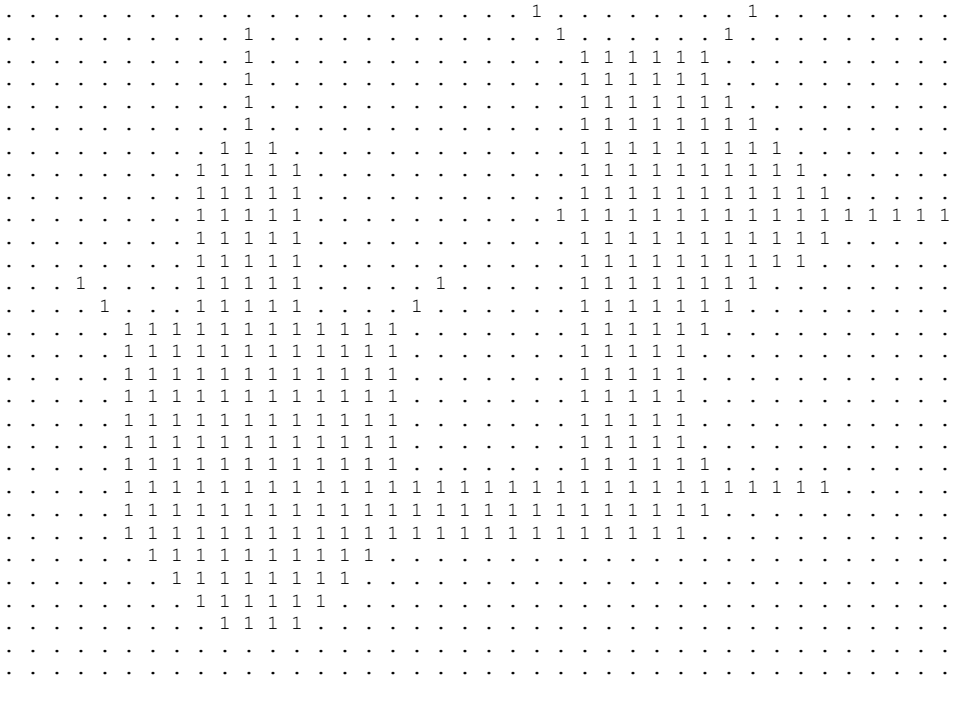
Before Thinning

```
30 40 0 1
. . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . 1 1 1 . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . . .
. . . . . . . . 1 1 1 1 1 . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . . . 1 1 1 1 1 1 1 . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . . . . . 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 . . . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . .
. . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . .
```

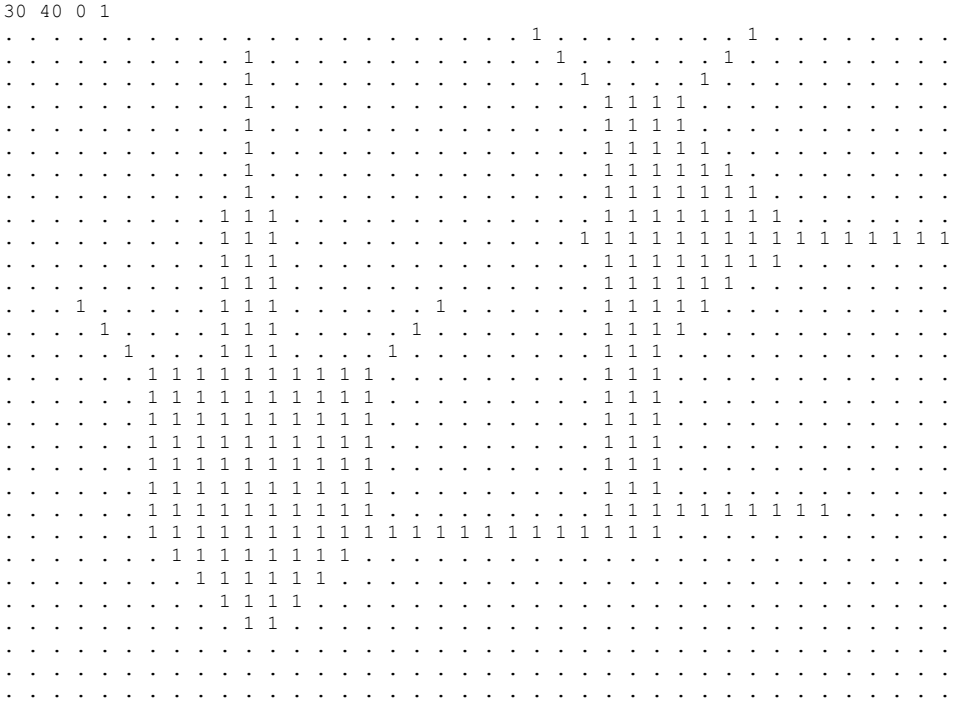
Result of Thinning (N,S = 4): Cycle - 1



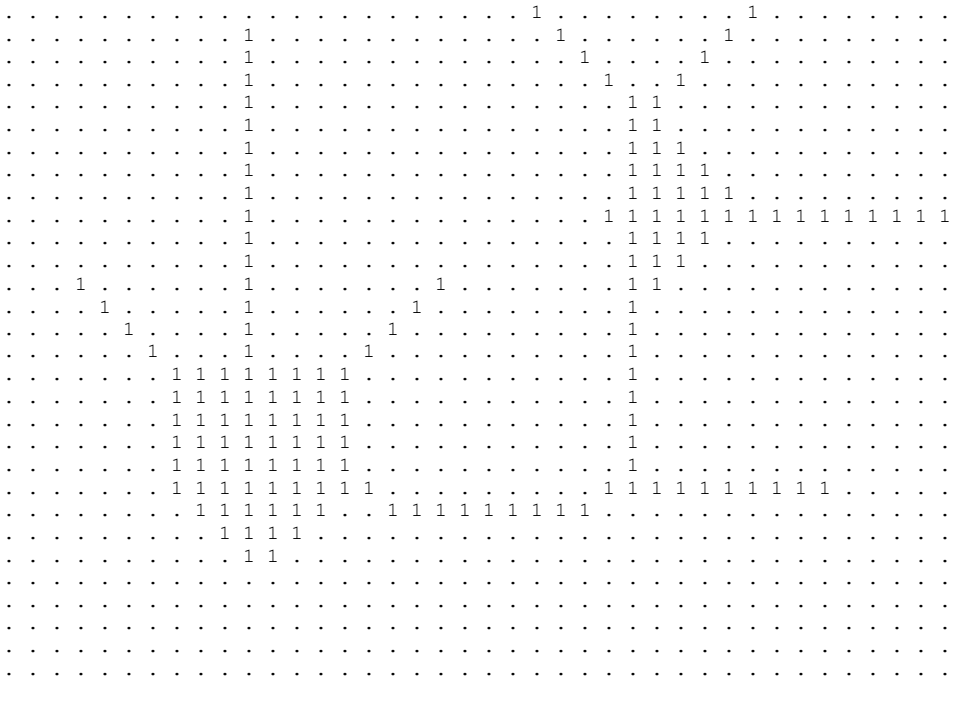
Result of Thinning (N,S = 4): Cycle - 2



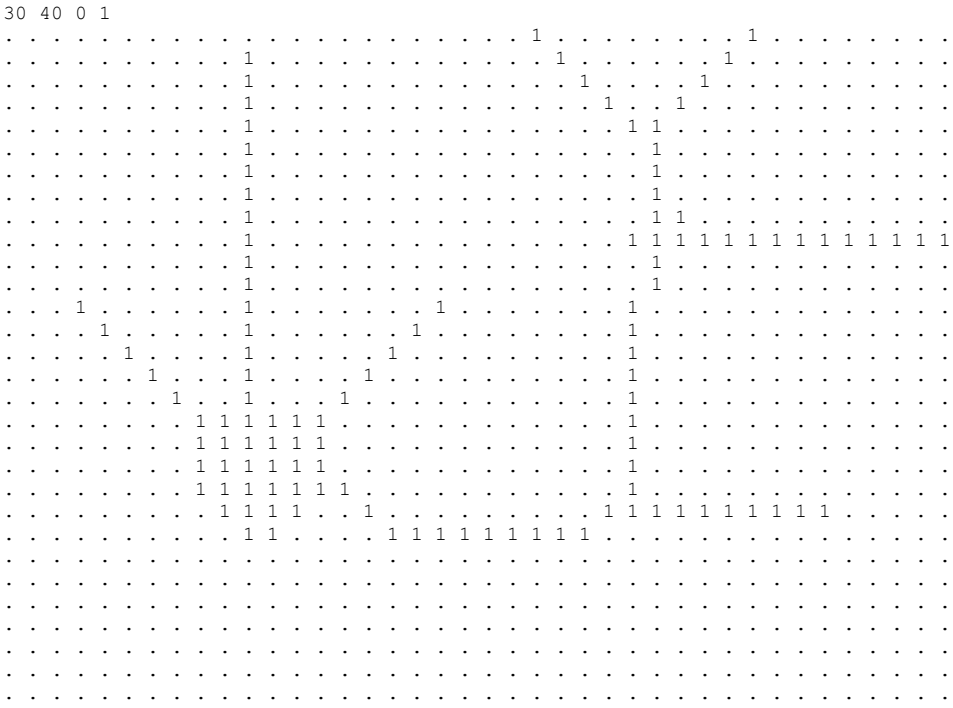
Result of Thinning (N,S = 4): Cycle - 3



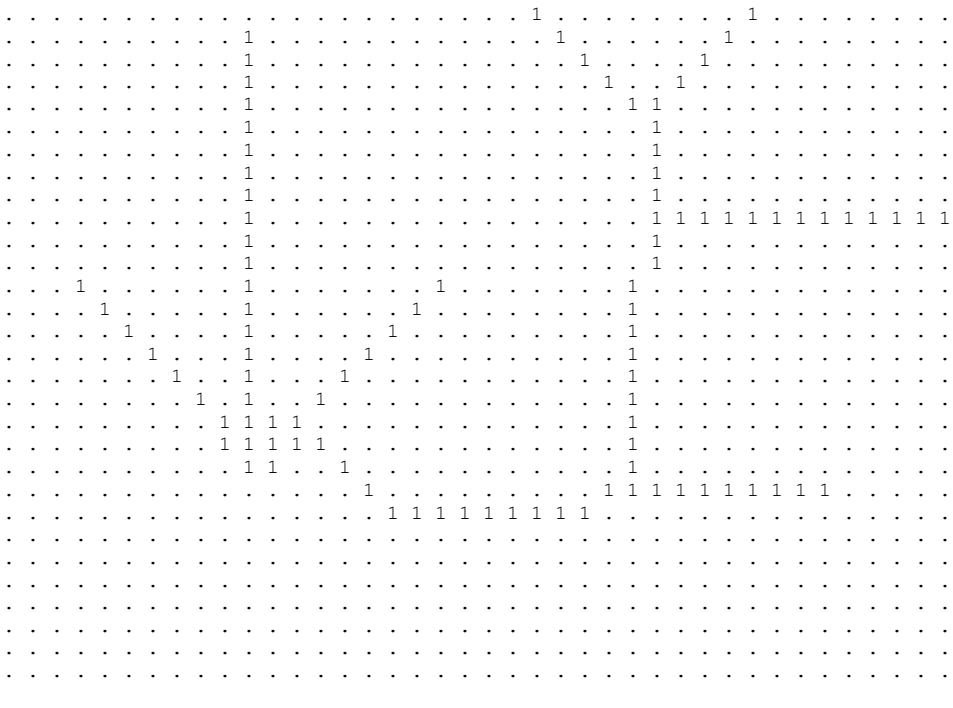
Result of Thinning (N,S = 4): Cycle - 4



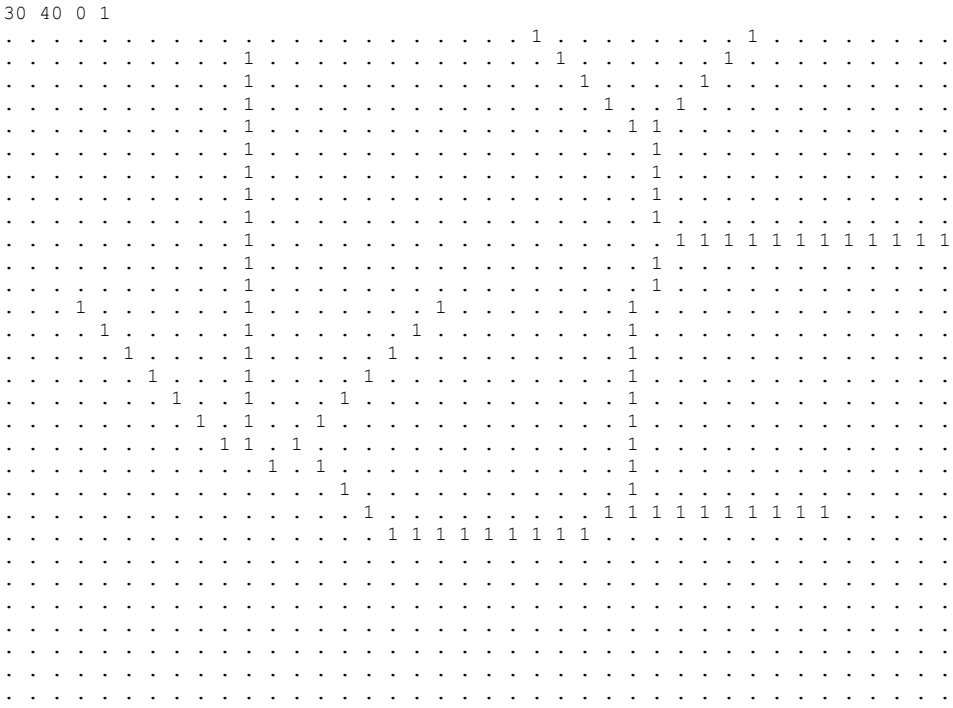
Result of Thinning (N,S = 4): Cycle - 5



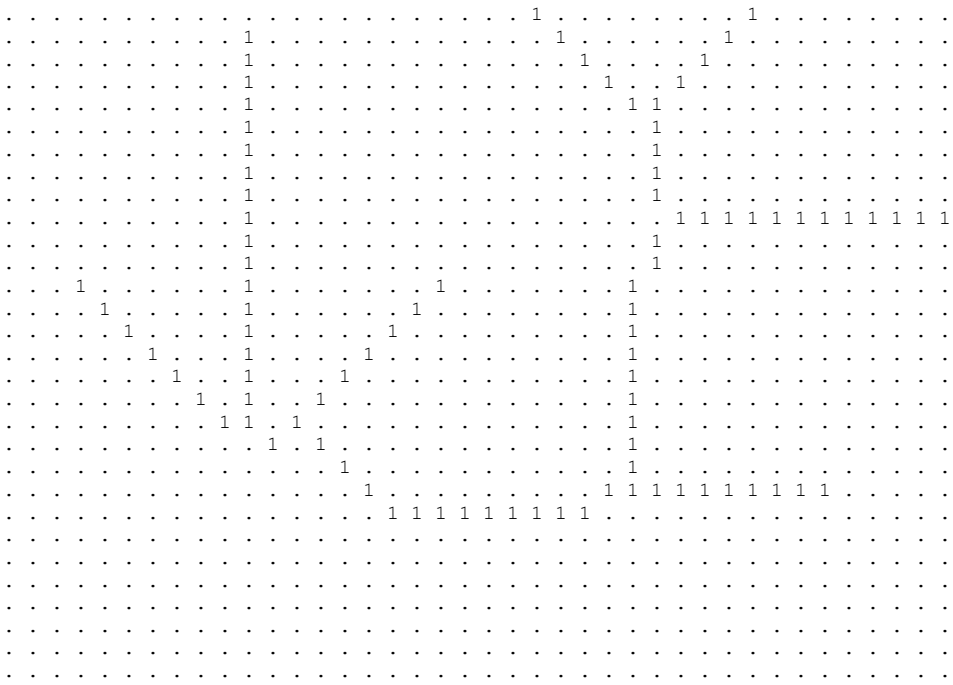
Result of Thinning (N,S = 4): Cycle - 6



Result of Thinning (N,S = 4): Cycle - 7



Result of Thinning (N,S = 4): Cycle - 8

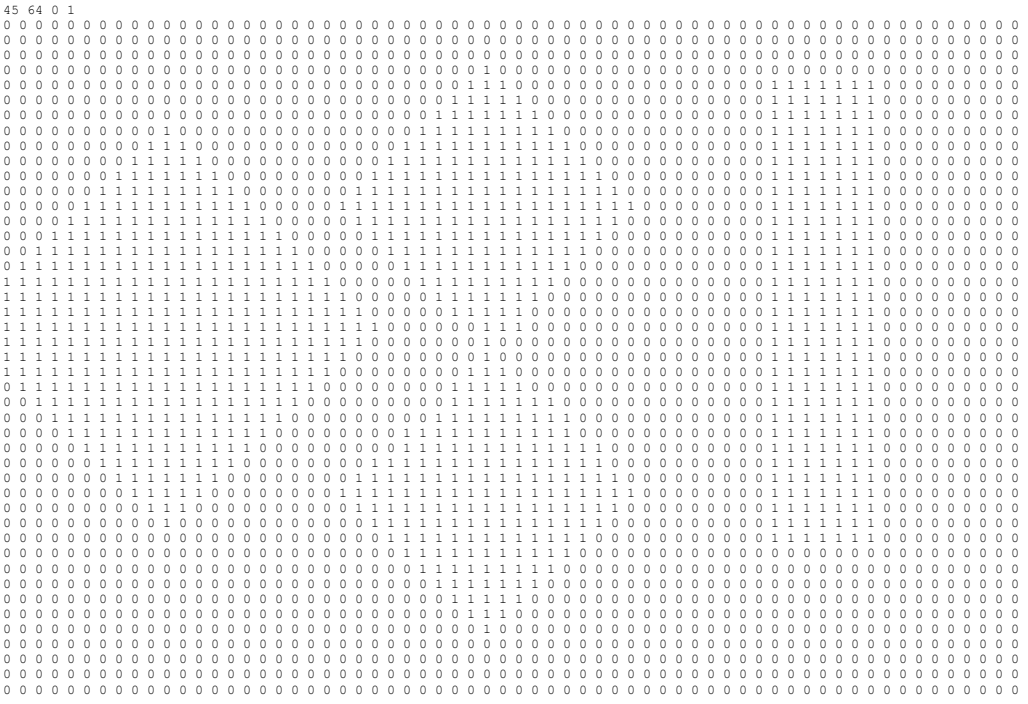


Final Skeleton Image after Thinning

```
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 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 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 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 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 1 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 1 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 1 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 1 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
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 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 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 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 1 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 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 1 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 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 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 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 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 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 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 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
```

Data_2:

Original Image



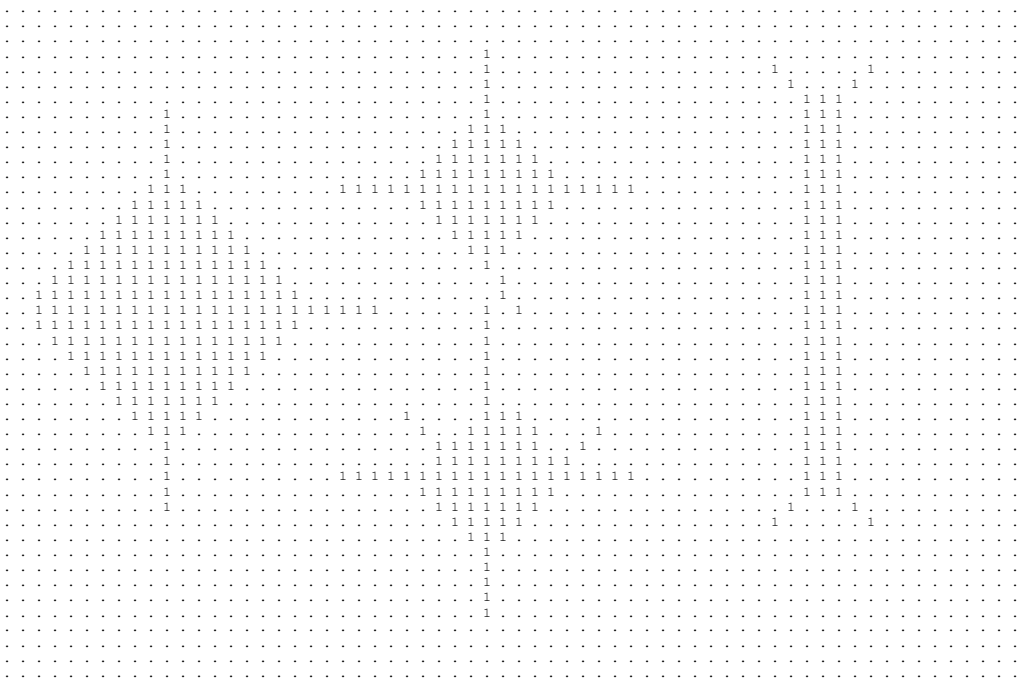
Before Thinning

Result of Thinning (N,S = 4): Cycle - 1

45 64 0 1

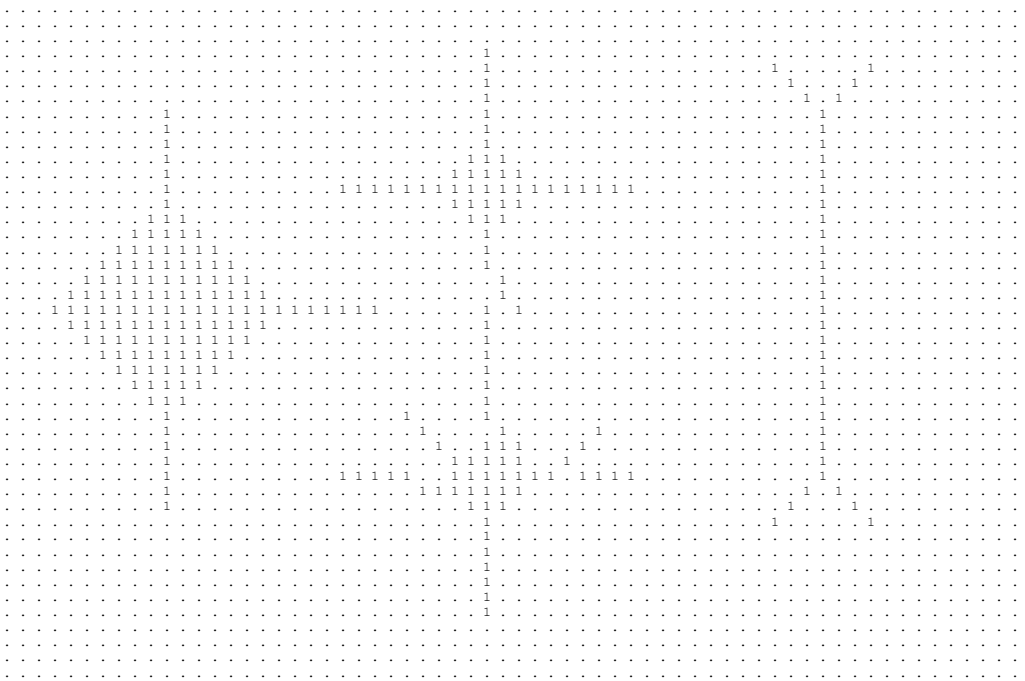
Result of Thinning (N,S = 4): Cycle - 2

45 64 0 1



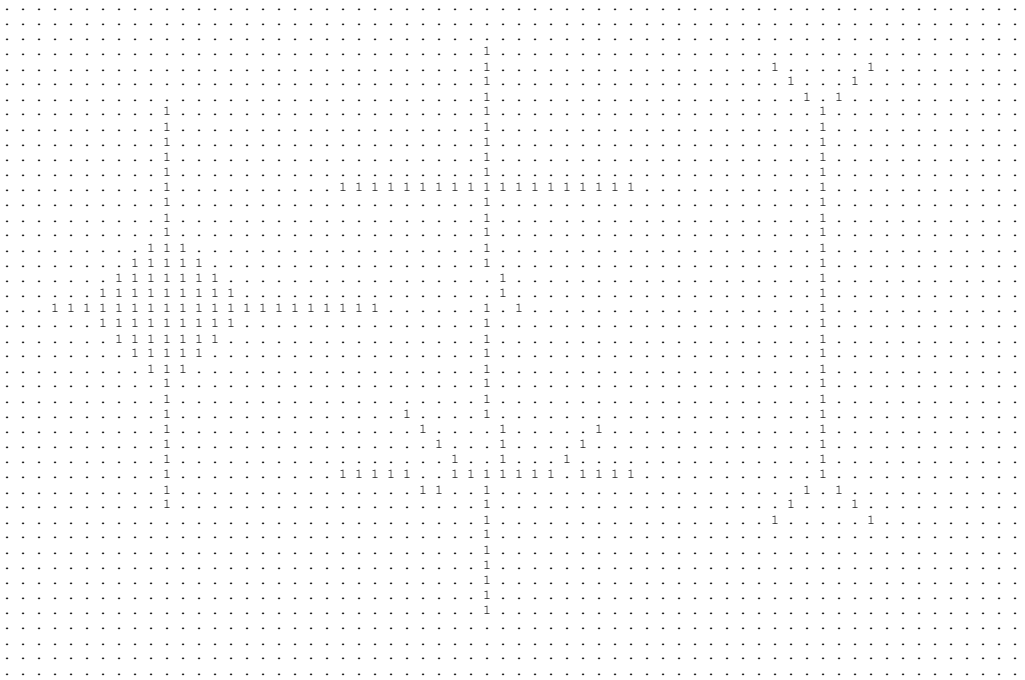
Result of Thinning (N,S = 4): Cycle – 3

45 64 0 1

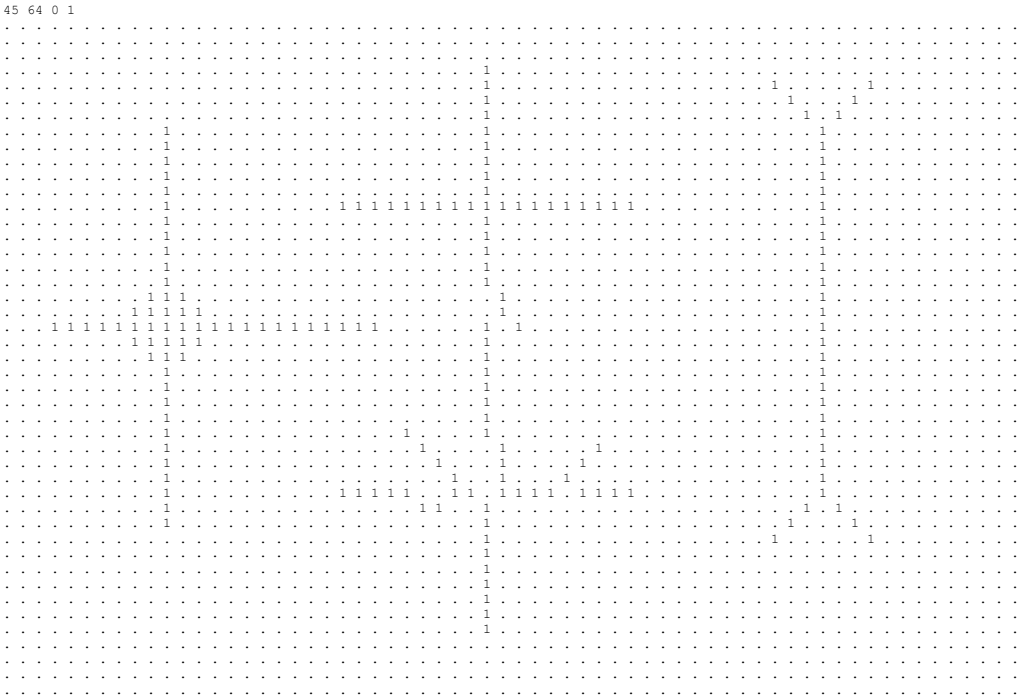


Result of Thinning (N,S = 4): Cycle – 4

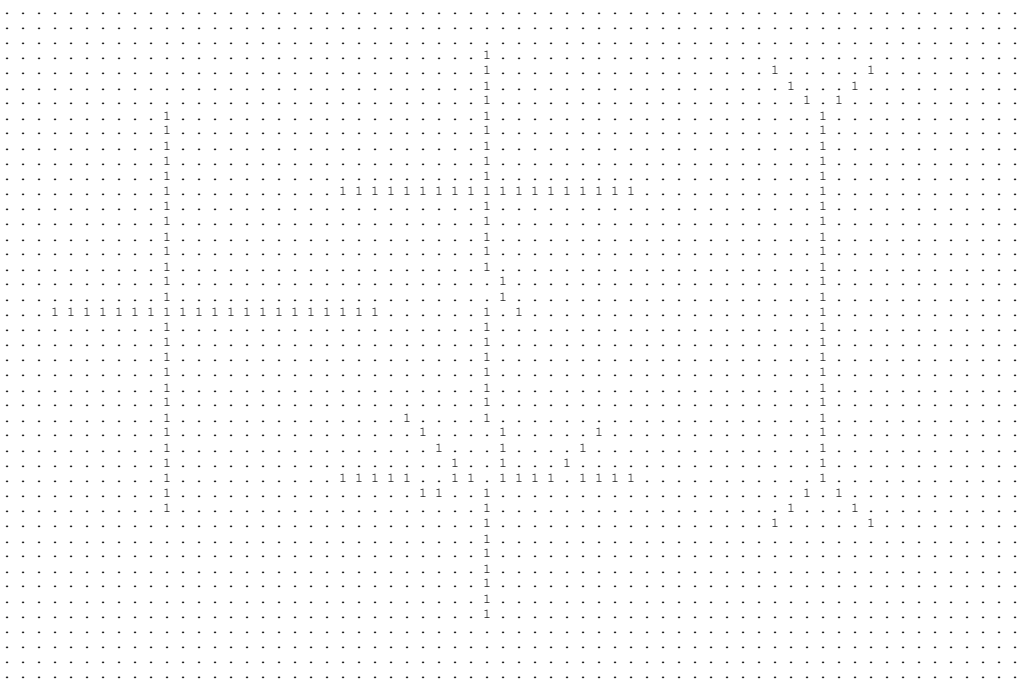
45 64 0 1



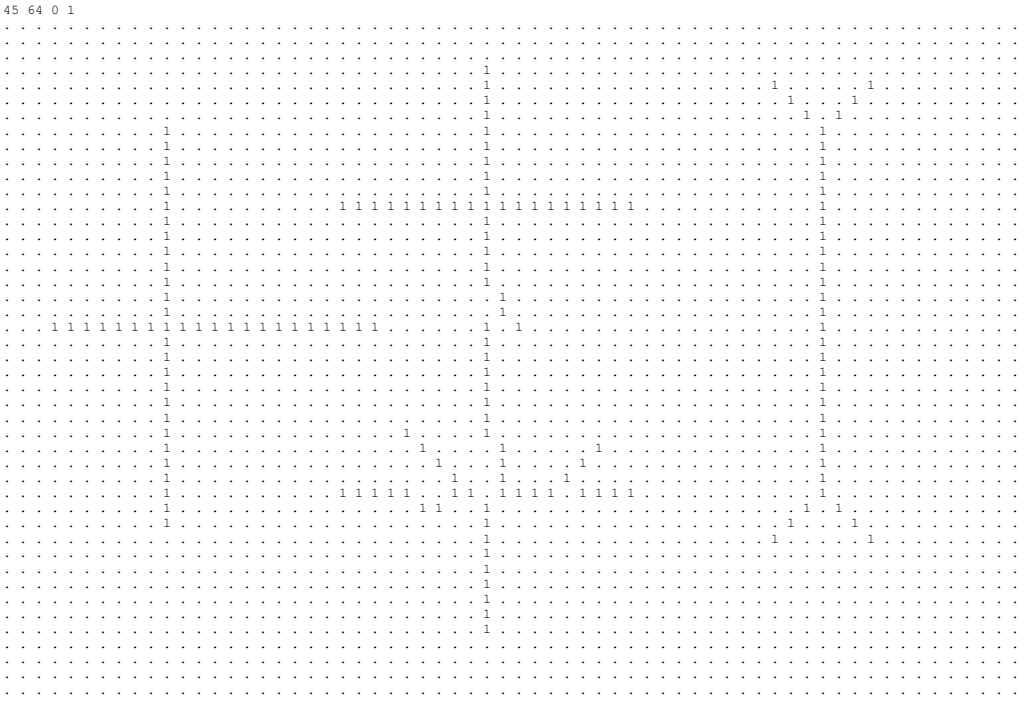
Result of Thinning (N,S = 4): Cycle – 5



Result of Thinning (N,S = 4): Cycle – 6



Result of Thinning (N,S = 4): Cycle - 7



Final Skeleton Image After Thinning