Assignment 1
CS 401 Algorithms

Objective

The objective of this assignment is to give you hands-on experience with using the *union-find* algorithm (specifically the *Weighted Quick Union* implementation) to solve a problem.

Note: you are supposed to implement union-find and you cannot use the union-find implementations provided in alg4.jar that comes with the textbook.

Problem Statement and Instructions

The Ability of Soil to Hold Water: Soil scientists characterize and classify soils into different groups, where each group of soil has differing ability to hold water depending on the particles inside. For the sake of simplicity, you are given the internal structure of soil as *n*-by-*n* grid of *cells*. Each cell is either *1* or *0*.

- 1 means the cell allows water to drain.
- 0 means the cell holds the water.

The input data is a file that looks like the below. Your visual interpretation of the data should be as drawn in Figure 1. That is, the first row is ground surface, and the bottom row is deep in the ground.

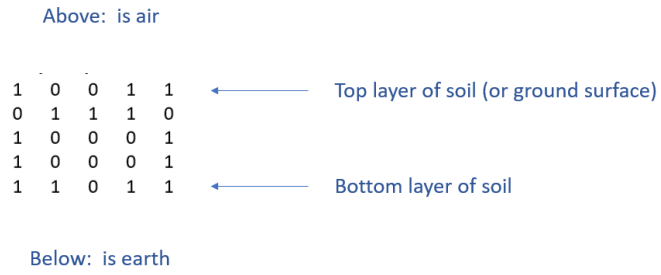

Figure 1: Orientation of the data with respect to the ground.

What your code should do is determine if the sample soil data given to you allows water (when placed on the ground surface) to drain all the way to the bottom layer or not. Note that we will assume that water can move horizontally or vertically (but not diagonally). Your code should use the Weighted Quick Union union-find algorithm as part of solving the problem.

You need to create the required classes and methods to enable the following client application

to read data from a specified file path and determine if water can drain from it or not. Your `doesDrain()` method should simply return `true` or `false`.

Please ensure that your code works correctly with the example code shown below, as it will be tested using this same code (the argument in blue will be the full path to some file I will pass to your `Soil` class constructor).

```
public class App {
    public static void main(String[] args) throws Exception {
        Soil soil = new Soil(<path_to_sample_file>.txt);
        if (soil.doesDrain()) {
            System.out.println("Allows water to drain");
        } else {
            System.out.println("Doesn't allow water to drain");
        }
    }
}
```

To help you verify your implementation, enclosed are 3 sample files to test with (sample1.txt, sample2.txt, and sample3.txt) with their expected results.

Sample Input File-1:

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

Expected Output:

```
Allows water to drain
```

Sample Input File-2:

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

Expected Output:

```
Doesn’t allow water to drain
```

Sample Input File-3:

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

Expected Output:

```
Allows water to drain
```

Grading

Your work will be graded based on these factors:

- Meeting the requirements (10%)
- Running error-free (80%)
- Clear and well-explained code (5%)
- Use of object-oriented programming principles (5%)

What to Submit:

1. Create a folder named lastName_firstName_project1 (example: smith_john_project1)
2. Put your VSCode project in the above folder (If you are using an IDE other than Java, then include all *.java classes with a README if there is anything I need to know to make it work in VSCode).
3. Zip the lastName_firstName_project1 folder to generate file lastName_firstName_project1.zip
4. Upload lastName_firstName_project1.zip to Canvas by the due date.