There is only one coding problem given 100min.

The problem is as below:

A group of farmers has some elevation data, and we’re going to help them understand how rainfall flows over their farmland. We’ll represent the land as a two-dimensional array of altitudes and use the following model, based on the idea that water flows downhill: If a cell’s four neighboring cells all have higher altitudes, we call this cell a sink; water collects in sinks. Otherwise, water will flow to the neighboring cell with the lowest altitude. If a cell is not a sink, you may assume it has a unique lowest neighbor and that this neighbor will be lower than the cell. Cells that drain into the same sink – directly or indirectly – are said to be part of the same basin. Your challenge is to partition the map into basins. In particular, given a map of elevations, your code should partition the map into basins and output the sizes of the basins, in descending order. **Assume the elevation maps are square.**

Input will begin with a line with one integer, S, the height (and width) of the map.

The next S lines will each contain a row of the map, each with S integers – the elevations of the S cells in the row. Some farmers have small land plots such as the examples below, while some have larger plots. However, in no case will a farmer have a plot of land larger than S = 1000. Note: The input uses unix line endings (\n).

While correctness and performance are the most important parts of this problem, a human will be reading your solution, so please make an effort to submit clean, readable code. In particular, do not write code as if you were solving a problem for a competition.

A few examples are below.

Input: 3 1 5 2 2 4 7 3 6 9

Output: 7 2

The basins, labeled with A’s and B’s, are: A A B A A B A A A

Input: 1 10

Output: 1

There is only one basin in this case.

Input: 5 1 0 2 5 8 2 3 4 7 9 3 5 7 8 9 1 2 5 4 3 3 3 5 2 1

Output: 11 7 7

The basins, labeled with A’s, B’s, and C’s, are: A A A A A A A A A A B B A C C B B B C C B B C C C

Input: 4 0 2 1 3 2 1 0 4 3 3 3 3 5 5 2 1

Output: 7 5 4

The basins, labeled with A’s, B’s, and C’s, are: A A B B A B B B A B B C A C C C

Key Points:

* Think about one elevation with 2 sinks. (ie. 12321 has height of 3 with 2 sinks)
* One basin may have 2 adjacent sinks (ie. 541134 would be a basin with 2 sinks)

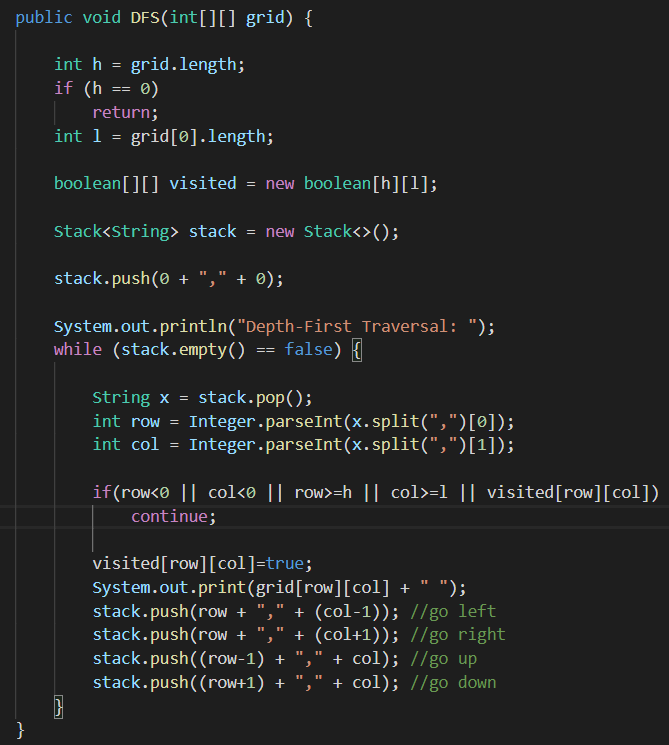
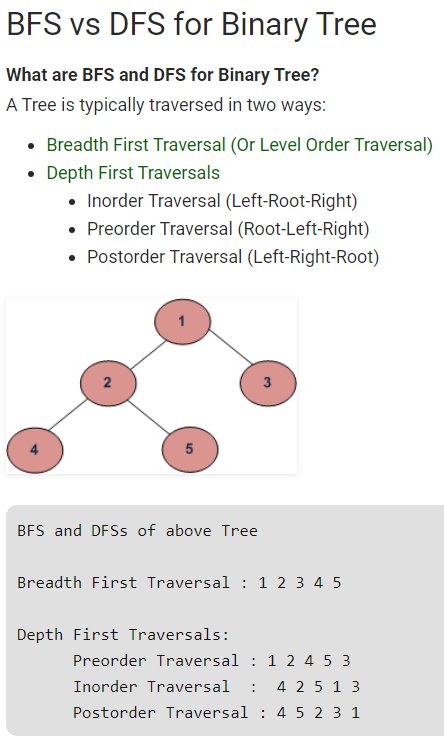
A sink is defined by having 4 neighboring cells with all higher altitudes (ie. 54145).

But (541145 is not a sink, as there is an additional 1 making it a basin.

Potentially use a Union-Find use Quick-Union algorithm,

Three questions.  
Two easy question + One medium question.

#1. Brute force to find the points where it is the only biggest value in 8 neighbors  
#2. DFS from the matrix you get from #1, which matrix[i][j] = 1is the water source. DFS to find how many layers of water from different sources.  
#3. Union Find to find plateau. pleateau definition: same level, no neighbors(8 neighbors) are higher than any points of this plateau. return a matrix of the plateau.



**733. Flood Fill**

An image is represented by a 2-D array of integers, each integer representing the pixel value of the image (from 0 to 65535). Given a coordinate (sr, sc) representing the starting pixel (row and column) of the flood fill, and a pixel value newColor, "flood fill" the image.

To perform a "flood fill", consider the starting pixel, plus any pixels connected 4-directionally to the starting pixel of the same color as the starting pixel, plus any pixels connected 4-directionally to those pixels (also with the same color as the starting pixel), and so on. Replace the color of all of the aforementioned pixels with the newColor. At the end, return the modified image.

**Example 1:**

**Input:**

image = [[1,1,1],[1,1,0],[1,0,1]]

sr = 1, sc = 1, newColor = 2

**Output:** [[2,2,2],[2,2,0],[2,0,1]]

**Explanation:**

From the center of the image (with position (sr, sc) = (1, 1)), all pixels connected

by a path of the same color as the starting pixel are colored with the new color.

Note the bottom corner is not colored 2, because it is not 4-directionally connected

to the starting pixel.

**Note:**

 The length of image and image[0] will be in the range [1, 50].

 The given starting pixel will satisfy 0 <= sr < image.length and 0 <= sc < image[0].length.

 The value of each color in image[i][j] and newColor will be an integer in [0, 65535].

