PA8: Drainage

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

You decide to give up computer science, and instead go into environmental engineering. Luckily, your computer science skills will come in handy! Your first job is to deal with modeling the water run-off – or drainage – in a basin area. Given a representation of the area to model, your task is to determine how far the water will flow.

Problem Description

The land will be represented by topographical map, which is a two-dimensional grid of elevations. Each grid will have r rows and c columns. Each grid location – or grid cell – will have an integer height elevation. Because you will be dealing with multiple grids, each one will have a title as well.

Your task is to figure out the *longest* sequence of grid locations that water can flow between. Water will flow from a higher elevation to a lower elevation. For the purposes of this problem, water will *never* flow

from a given elevation to the *same* elevation, nor will it flow uphill. Furthermore, water can only flow from one grid cell to an *adjacent* cell (adjacent cells are above, below, left, and right; not diagonal!).

As an example, consider the following 5x5 grid. Note that the input in this example is justified to help illustrate the grid; there will only be one space between heights in the actual input.

```
66 78 41  3 77
 4 90 41  8 68
12 11 29 24 53
 0 51 58  9 28
97 99 96 58 92
```

There are many such valid drainage paths in this grid. One starts in the cell (1,1), and flows through heights 90-78-41-3. Note that the sequennce of heights 90-41-41-3 is not a valid drainage flow, as water is not always flowing downhill (41-41 is *not* downhill). The longest drainage path in this example is of length 7, and flows from the 99 in cell (4,1); the full sequence of heights is 99-96-58-29-24-8-3. The equivalent path of locations is (4,1)-(4,2)-(3,2)-(2,2)-(2,3)-(1,3)-(0,3).

You may solve this problem using top-down Dynamic Programming or using bottom-up Dynamic Programming, but a brute-force solution's running time will be too long.

Input: A 2-d array of doubles/floats indicating the altitude of each location.

Output: Print the sequence of locations of the longest drainage run

Input Format

Input will be given in the console (same as with the previous PAs). To provide an input grid of size $r \times c$, you will have $r + 1$ lines in your input as follows:

– The first line contains two space-separated ints representing the values of r and c, i.e. or rows and columns in the grid, respectively.

– The remaining r lines contain each contain c space-separated floats/doubles representing the altitudes of each location in the grid.

Output Format

To give output your program should print the sequence of grid indices that represent the longest drainage run. Indices should be printed such that the highest elevation location in the path is printed first, and the lowest elevation location is printed last. The locations should be printed one per line, with the indices separated by single space. For example, for the grid above you should print:

```
4 1
4 2
3 2
2 2
2 3
1 3
0 3
```


Test Cases Summary

There are 10 test cases provided to you. These represent all of the test cases that the autograder will use to check your code. There are no hidden tests. Some tests were designed to produce particular patterns through the grid, but most are just random.

The path lengths for all 10 tests are listed below:

- `examplegrid.txt` is length 7
- `snake.txt` is length 25
- `flat.txt` is length 1
- `8x8.txt` is length 6
- `10x10.txt` is length 9
- `15x15.txt` is length 10
- `20x20.txt` is length 400
- `25x25.txt` is length 9
- `30x30.txt` is length 117
- `100x100.txt` is length 13

Starter Code Summary

The starter code provided consists of two files:

- **`ReadGrid`**`.java/read_grid.py`: This is the program you will use to run your code. You will not submit this program, instead the autograder will be using its own version to interact with your code.
- `Drainage.java/drainage.py`: This is the program that you will edit and submit. There is just one function in the program. This function, called `max_run` takes in the grid. It then returns the maximum length drainage run in the grid. As always, you're welcome to add any "helper functions" as you see fit.

