Damavis Challenge

Hello, future Damavis teammate! first of all thanks for accepting our challenge. You are free to use any programming language you like (Preferably use Scala, Python, Java or R)

There are two different challenges, one to evaluate how you code, and another one to evaluate how you work with data. For the first challenge try not to use any framework, use only the base language capabilities.

We value readable code, structure, tests, good code principles and best practices. You are free to choose the programming code paradigm and architecture that you think that fits best.
If you cannot finish the challenge, please send us your partial solution anyway.

Please send us your own code, thought and written by yourself without any help from anyone.

It's preferably you to create a Version Control System (VCS) repository. Git could be a good choice to track your commits, and easily share your results with us. Although, if you are not familiar with VCS, simply send us a zip file containing your code.

BEST OF LUCK!

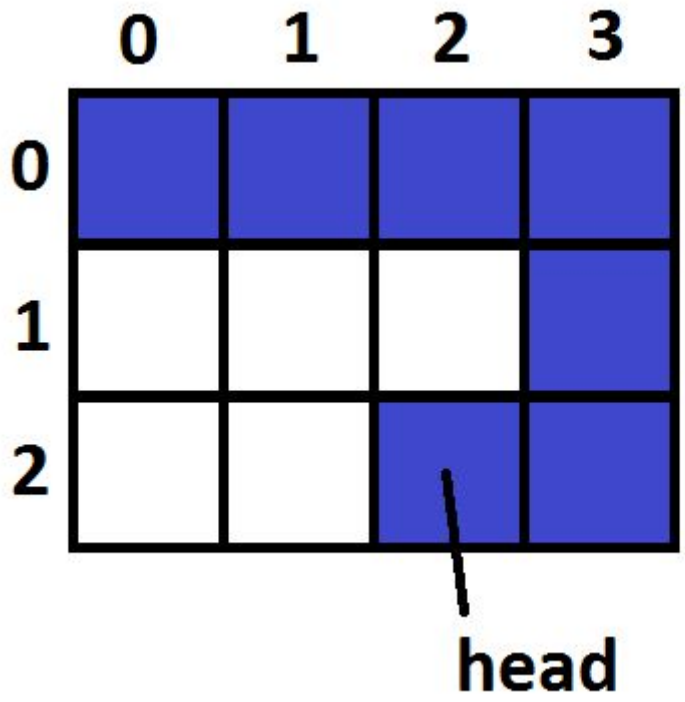
1. Algorithm and data structures

Consider a rectangular board consisting of $n \times m$ cells. There is a snake lying on some of the cells, and all of the other cells are empty. The snake consists of one or more cells such that cells with consecutive numbers are either horizontally or vertically adjacent. The first cell is the snake's head, and the last cell is the snake's tail.

On each turn, the snake's head moves to one of the horizontally or vertically adjacent cells, the second cell of the snake moves to the cell where the head was situated, the third cell takes the former place of the second cell, etc. All these movements happen simultaneously, so the head could potentially take the place of the tail. There are two configurations of the snake's cells that are prohibited: self-intersection (meaning that after each step any pair of snake cells should have pairwise different coordinates), and crossing the board's border. The path is a sequence of characters L, R, D, and U(corresponding to left, right, down, and up, respectively) describing the movements of snake's head on each turn. How many distinct valid paths of length p can the snake make on the board?

Example:

For board = *[4, 3]*, snake = *[[2, 2], [3, 2], [3, 1], [3, 0], [2, 0], [1, 0], [0, 0]]*, and depth = *3*, the output should be “numberOfAvailableDifferentPaths(board, snake, depth) = 7”.



Here are all of the valid paths with a length of 3 that the snake can make:

- LLU
- LUR
- LUL
- ULL
- ULD
- LUU
- ULU

Contract Input Output.

There are 3 input argument in this challenge:

Input

- **board** as array.integer

  The *board* is an array describing the dimensions of the board. *board[0]* stands for the number of rows, and *board[1]* corresponds to the number of columns.
  Guaranteed constraints:
  board.length = 2,
  1 ≤ board[i] ≤ 10.

- **snake** as array.array.integer

  The *snake* is an array that describes the configuration of the snake's body on the board. *snake[0]* corresponds to the initial coordinates of the snake's head, *snake[1]* corresponds to coordinates of the second cell, etc.
  It is guaranteed that *snake[i]* and *snake[i + 1]* are horizontally or vertically adjacent, and that its initial configuration is valid (i.e. there are no self-intersections and the snake's entire body lies inside the board).
  Guaranteed constraints:
  3 ≤ *snake.length* ≤ 7,
  snake[i].length = 2,
  0 ≤ *snake[i][j]* < board[j].

- **depth** as integer

  The *depth* is the paths depth. You have to discard all paths with a different path length.
  Guaranteed constraints:
  1 ≤ n ≤ 20.

Output as integer

The number of distinct valid paths of length *p* that the snake can make, modulo $10^9 + 7$.

Acceptance tests

- Test 1:
  - board: [4, 3]
  - snake: [[2,2], [3,2], [3,1], [3,0], [2,0], [1,0], [0,0]]
  - depth: 3

  Result: 7

- Test 2:
  - board: [2, 3]
  - snake: [[0,2], [0,1], [0,0], [1,0], [1,1], [1,2]]
  - depth: 10

  Result: 1

- Test 3:
  - board: [10, 10]
  - snake: [[5,5], [5,4], [4,4], [4,5]]
  - depth: 4

  Result: 81

2. Data analytics

The second challenge is a very well known use case. As in the previous challenge, you are free to use the language you are most familiar with. What we want from you is that you develop a tool that generates a few reports, which will be specified in the following lines, about an unstructured dataset which consists of an Apache2 web server log file. You can download the dataset from the following URL:

https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs

Restrictions

You cannot import more than 3 third party libraries (without of course counting dependencies). It is not allowed to use a Database (DBMS), such as MySQL or MongoDB. Please, write a README file explaining how to use your tool, what dependencies it has and any assumptions you have made.

Mandatory

* Top 10 requested pages and number of made requests for each one
* Percentage of successful requests (anything in the 200s and 300s range)
* Percentage of unsuccessful requests (anything that is not in the 200s or 300s range)
* Top 10 unsuccessful page requests
* The top 10 IPs making the largest number of requests. Please, display the IP address and the number of made requests.
* Your solution must take a parameter which specifies which of the previous reports must be built. A single report must be built per execution of your program (e.g. only the top 10 urls, only the percentage of successful requests and so on). You can call the possible values of the parameter as you like, such as TOP_10, TOP_OK, TOP_KO...

Extra

* Unit tests for your code.
* Total number of made requests every minute in the entire time period covered by the provided file.
* For each of the top 10 IPs, show the top 5 pages requested and the number of requests for each one.