introduction to web development and its scope

HTML5 (fundamentals, tags, forms, Media and Website Structure)

CSS3 fundamentals (box model, typography, media queries, transitions, keyframes etc.)

Working on Real-world project using HTML, CSS

Getting started with bootstrap (Layout, Components, forms, cards, Navbar, drop down etc.)

Introduction to JavaScript (datatypes, conditionals, loops, Document Object Model, Event Handling, AJAX etc.)

Future roadmap for tainees

Codewriting

Minesweeper is a popular single-player computer game. The goal is to locate mines within a rectangular grid of cells. At the start of the game, all of the cells are concealed. On each turn, the player clicks on a blank cell to reveal its contents, leading to the following result:

If there's a mine on this cell, the player loses and the game is over;

Otherwise, a number appears on the cell, representing how many mines there are within the 8 neighbouring cells (up, down, left, right, and the 4 diagonal directions);

If the revealed number is 0, each of the 8 neighbouring cells are automatically revealed in the same way.

demonstration

You are given a boolean matrix field representing the distribution of bombs in the rectangular field. You are also given integers x and y, representing the coordinates of the player's first clicked cell - x represents the row index, and y represents the column index, both of which are 0-based.

Your task is to return an integer matrix of the same dimensions as field, representing the resulting field after applying this click. If a cell remains concealed, the corresponding element should have a value of -1.

It is guaranteed that the clicked cell does not contain a mine.

Example

For

field = [[false, true, true],

[true, false, true],

[false, false, true]]

x = 1, and y = 1, the output should be

solution(field, x, y) = [[-1, -1, -1],

[-1, 5, -1],

[-1, -1, -1]]

example

There are 5 neighbors of the cell (1, 1) which contain a mine, so the value in (1, 1) should become 5, and the other elements of the resulting matrix should be -1 since no other cell would be expanded.

For

field = [[true, false, true, true, false],

[true, false, false, false, false],

[false, false, false, false, false],

[true, false, false, false, false]]

x = 3, and y = 2, the output should be

solution(field, x, y) = [[-1, -1, -1, -1, -1],

[-1, 3, 2, 2, 1],

[-1, 2, 0, 0, 0],

[-1, 1, 0, 0, 0]]

example

Since the value in the cell (3, 2) is 0, all of its neighboring cells ((2, 1), (2, 2), (2, 3), (3, 1), and (3, 3)) are also revealed. Since the value in the cell (2, 2) is also 0, its neighbouring cells (1, 1), (1, 2) and (1, 3) are revealed, and since the value in cell (2, 3) is 0, its neighbours (1, 4), (2, 4), and (3, 4) are also revealed. The cells (3, 3), (2, 4), and (3, 4) also contain the value 0, but since all of their neighbours have already been revealed, no further action is required.

Input/Output

[execution time limit] 4 seconds (js)

[input] array.array.boolean field

A rectangular matrix representing the locations of the mines in the game field.

Guaranteed constraints:

2 ≤ field.length ≤ 100,

2 ≤ field[i].length ≤ 100.

[input] integer x

The row number of the cell which is clicked (0-based).

Guaranteed constraints:

0 ≤ x < field.length.

[input] integer y

The column number of the cell which is clicked (0-based).

Guaranteed constraints:

0 ≤ y < field[0].length.

[output] array.array.integer

The expanded matrix after the click.

[JavaScript] Syntax Tips

// Prints help message to the console

// Returns a string

function helloWorld(name) {

console.log("This prints to the console when you Run Tests");

return "Hello, " + name;

}

Given an array of positive integers a, your task is to calculate the sum of every possible a[i] ∘ a[j], where a[i] ∘ a[j] is the concatenation of the string representations of a[i] and a[j] respectively.

Example

For a = [10, 2], the output should be solution(a) = 1344.

a[0] ∘ a[0] = 10 ∘ 10 = 1010,

a[0] ∘ a[1] = 10 ∘ 2 = 102,

a[1] ∘ a[0] = 2 ∘ 10 = 210,

a[1] ∘ a[1] = 2 ∘ 2 = 22.

So the sum is equal to 1010 + 102 + 210 + 22 = 1344.

For a = [8], the output should be solution(a) = 88.

There is only one number in a, and a[0] ∘ a[0] = 8 ∘ 8 = 88, so the answer is 88.

For a = [1, 2, 3], the output should be solution(a) = 198.

a[0] ∘ a[0] = 1 ∘ 1 = 11,

a[0] ∘ a[1] = 1 ∘ 2 = 12,

a[0] ∘ a[2] = 1 ∘ 3 = 13,

a[1] ∘ a[0] = 2 ∘ 1 = 21,

a[1] ∘ a[1] = 2 ∘ 2 = 22,

a[1] ∘ a[2] = 2 ∘ 3 = 23,

a[2] ∘ a[0] = 3 ∘ 1 = 31,

a[2] ∘ a[1] = 3 ∘ 2 = 32,

a[2] ∘ a[2] = 3 ∘ 3 = 33.

The total result is 11 + 12 + 13 + 21 + 22 + 23 + 31 + 32 + 33 = 198.

Input/Output

[execution time limit] 4 seconds (js)

[input] array.integer a

A non-empty array of positive integers.

Guaranteed constraints:

1 ≤ a.length ≤ 105,

1 ≤ a[i] ≤ 106.

[output] integer64

The sum of all a[i] ∘ a[j]s. It's guaranteed that the answer is less than 253.

[JavaScript] Syntax Tips

Given an array of integers a, your task is to find how many of its contiguous subarrays of length m contain a pair of integers with a sum equal to k.

More formally, given the array a, your task is to count the number of indices 0 ≤ i ≤ a.length - m such that a subarray [a[i], a[i + 1], ..., a[i + m - 1]] contains at least one pair (a[s], a[t]), where:

s ≠ t

a[s] + a[t] = k

Example

For a = [2, 4, 7, 5, 3, 5, 8, 5, 1, 7], m = 4, and k = 10, the output should be solution(a, m, k) = 5.

Let's consider all subarrays of length m = 4 and see which fit the description conditions:

Subarray a[0..3] = [2, 4, 7, 5] doesn't contain any pair of integers with a sum of k = 10. Note that although the pair (a[3], a[3]) has the sum 5 + 5 = 10, it doesn't fit the requirement s ≠ t.

Subarray a[1..4] = [4, 7, 5, 3] contains the pair (a[2], a[4]), where a[2] + a[4] = 7 + 3 = 10.

Subarray a[2..5] = [7, 5, 3, 5] contains two pairs (a[2], a[4]) and (a[3], a[5]), both with a sum of k = 10.

Subarray a[3..6] = [5, 3, 5, 8] contains the pair (a[3], a[5]), where a[3] + a[5] = 5 + 5 = 10.

Subarray a[4..7] = [3, 5, 8, 5] contains the pair (a[5], a[7]), where a[5] + a[7] = 5 + 5 = 10.

Subarray a[5..8] = [5, 8, 5, 1] contains the pair (a[5], a[7]), where a[5] + a[7] = 5 + 5 = 10.

Subarray a[6..9] = [8, 5, 1, 7] doesn't contain any pair with a sum of k = 10.

So the answer is 5, because there are 5 contiguous subarrays that contain a pair with a sum of k = 10.

For a = [15, 8, 8, 2, 6, 4, 1, 7], m = 2, and k = 8, the output should be solution(a, m, k) = 2.

There are 2 subarrays satisfying the description conditions:

a[3..4] = [2, 6], where 2 + 6 = 8

a[6..7] = [1, 7], where 1 + 7 = 8

Input/Output

[execution time limit] 4 seconds (js)

[input] array.integer a

The given array of integers.

Guaranteed constraints:

2 ≤ a.length ≤ 105,

0 ≤ a[i] ≤ 109.

[input] integer m

An integer representing the length of the contiguous subarrays being considered.

Guaranteed constraints:

2 ≤ m ≤ a.length.

[input] integer k

An non-negative integer value representing the sum of the pairs we're trying to find within each subarray.

Guaranteed constraints:

0 ≤ k ≤ 109.

[output] integer

An integer representing the number of subarrays that contain a pair of integers with a sum of k.

You are given an array of arrays a. Your task is to group the arrays a[i] by their mean values, so that arrays with equal mean values are in the same group, and arrays with different mean values are in different groups.

Each group should contain a set of indices (i, j, etc), such that the corresponding arrays (a[i], a[j], etc) all have the same mean. Return the set of groups as an array of arrays, where the indices within each group are sorted in ascending order, and the groups are sorted in ascending order of their minimum element.

Example

For

a = [[3, 3, 4, 2],

[4, 4],

[4, 0, 3, 3],

[2, 3],

[3, 3, 3]]

the output should be

solution(a) = [[0, 4],

[1],

[2, 3]]

mean(a[0]) = (3 + 3 + 4 + 2) / 4 = 3;

mean(a[1]) = (4 + 4) / 2 = 4;

mean(a[2]) = (4 + 0 + 3 + 3) / 4 = 2.5;

mean(a[3]) = (2 + 3) / 2 = 2.5;

mean(a[4]) = (3 + 3 + 3) / 3 = 3.

There are three groups of means: those with mean 2.5, 3, and 4. And they form the following groups:

Arrays with indices 0 and 4 form a group with mean 3;

Array with index 1 forms a group with mean 4;

Arrays with indices 2 and 3 form a group with mean 2.5.

Note that neither

solution(a) = [[0, 4],

[2, 3],

[1]]

nor

solution(a) = [[0, 4],

[1],

[3, 2]]

will be considered as a correct answer:

In the first case, the minimal element in the array at index 2 is 1, and it is less then the minimal element in the array at index 1, which is 2.

In the second case, the array at index 2 is not sorted in ascending order.

For

a = [[-5, 2, 3],

[0, 0],

[0],

[-100, 100]]

the output should be

solution(a) = [[0, 1, 2, 3]]

The mean values of all of the arrays are 0, so all of them are in the same group.

Input/Output

[execution time limit] 4 seconds (js)

[input] array.array.integer a

An array of arrays of integers.

Guaranteed constraints:

1 ≤ a.length ≤ 100,

1 ≤ a[i].length ≤ 100,

-100 ≤ a[i][j] ≤ 100.

[output] array.array.integer

An array of arrays, representing the groups of indices.

[JavaScript] Syntax Tips