**300**

**0**

**0**

300 : 300

ROUND 2

**300**

**0**

**0**

DESCRIPTION

SOLUTIONS

366

RULES

README

**CODEWRITING**

**300 PTS**

Backing up files is a pretty straightforward process most of the time, but it can get much more complicated when you are backing up to the cloud. Let's assume that Datto backs up files in the cloud in batches. This works well until new files are added to the queue while a batch is already in the process of being backed up. We call these files *trouble files*, and identifying them is important when it comes to optimizing backup efficiency.

You are given an array, files, representing all files that need a backup, where:

* files[i][0] is the time a file was added to the backup queue;
* files[i][1] is the size of the file.

You are also given a sorted array, backups, representing multiple completed backups.

backups[i] is the time the ith backup was started. Each backup only backs up files that were not backed up or marked as *trouble* by the previous backups. If a backup started or finished at the same moment a file was added, assume that appending the new file to the queue occurred first. You can also assume that the time it takes to back up a list of files is equal to the sum of the sizes of those files.

Your goal is to calculate the number of *trouble files* for each completed backup, to estimate how severely they impacted the efficiency of the backup.

**Example**

For

files = [[461618501, 3],

[461618502, 1],

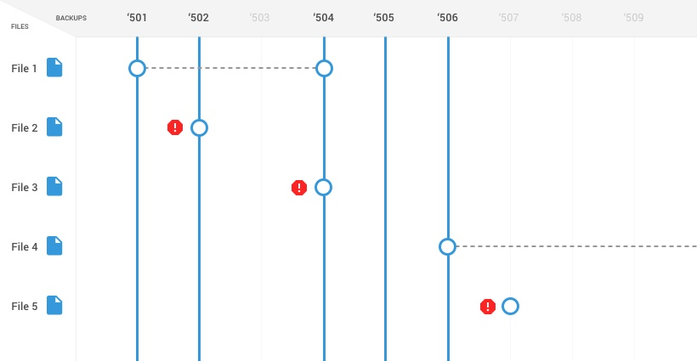
[461618504, 2],

[461618506, 5],

[461618507, 6]]

and backups = [461618501, 461618502, 461618504, 461618505, 461618506],  
the output should be  
troubleFiles(files, backups) = [2, 0, 0, 0, 1].

* The first backup started at 461618501, backing up the first file from the files list.
* The second file was added to the queue at 461618502, and labeled as *trouble*. After that the second backup started and immediately finished since there was nothing to back up.
* At 461618504, the third file was added and marked as *trouble* for the first backup, after which the first backup finished and the third one started. Yet there was nothing to back up for it, so it finished immediately.
* At 461618505 the fourth backup started and immediately finished since there was still nothing to back up.
* At 461618506 the fourth file was added to the queue, and the fifth backup process started.
* Finally, at 461618507 the last file was added and marked as *trouble* for the fifth backup.

Here is the backups timeline:  


**Input/Output**

* **[time limit] 3000ms (java)**
* **[input] array.array.integer files**

A non-empty array of files. Each file is given as an array of two positive integers, the first one being the time it was added, and the second one being its size. The arrays are sorted by their first elements. If for some i files[i][0] = files[i + 1][0], assume that the ith file was added first.

1 ≤ files.length ≤ 25,  
108 ≤ files[i][0] ≤ 109,  
1 ≤ files[i][1] ≤ 100.

* **[input] array.integer backups**

A non-empty sorted array of unique positive integers, where the ith element is the time the ith backup started.

1 ≤ backups.length ≤ 65,  
108 ≤ backups[i] ≤ 109.

* **[output] array.integer**

Array of the same length as backups, where the ith element is the number of *trouble* files occurred during the ith backup.

**GIVE UP**

**START CODING**

1:10