Problem 3433: Count Mentions Per User

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given an integer

numberOfUsers

representing the total number of users and an array

events

of size

n x 3

.

Each

events[i]

can be either of the following two types:

Message Event:

["MESSAGE", "timestamp

i

", "mentions_string

i

"]

This event indicates that a set of users was mentioned in a message at

timestamp

i

.

The

mentions_string

i

string can contain one of the following tokens:

id<number>

: where

<number>

is an integer in range

[0,numberOfUsers - 1]

. There can be

multiple

ids separated by a single whitespace and may contain duplicates. This can mention even the offline users.

ALL

: mentions

all

users.

HERE

: mentions all

online

users.

Offline Event:

["OFFLINE", "timestamp

i

", "id

i

"]

This event indicates that the user

id

i

had become offline at

timestamp

i

for

60 time units

. The user will automatically be online again at time

timestamp

i

+ 60

.

Return an array

mentions

where

mentions[i]

represents the number of mentions the user with id

i

has across all

MESSAGE

events.

All users are initially online, and if a user goes offline or comes back online, their status change is processed

before

handling any message event that occurs at the same timestamp.

Note

that a user can be mentioned

multiple

times in a

single

message event, and each mention should be counted

separately

.

Example 1:

Input:

numberOfUsers = 2, events = [["MESSAGE","10","id1 id0"],["OFFLINE","11","0"],["MESSAGE","71","HERE"]]

Output:

[2,2]

Explanation:

Initially, all users are online.

At timestamp 10,

id1

and

id0

are mentioned.

mentions = [1,1]

At timestamp 11,

id0

goes

offline.

At timestamp 71,

id0

comes back

online

and

"HERE"

is mentioned.

mentions = [2,2]

Example 2:

Input:

numberOfUsers = 2, events = [["MESSAGE","10","id1
id0"],["OFFLINE","11","0"],["MESSAGE","12","ALL"]]

Output:

[2,2]

Explanation:

Initially, all users are online.

At timestamp 10,

id1

and

id0

are mentioned.

mentions = [1,1]

At timestamp 11,

id0

goes

offline.

At timestamp 12,

"ALL"

is mentioned. This includes offline users, so both

id0

and

id1

are mentioned.

mentions = [2,2]

Example 3:

Input:

numberOfUsers = 2, events = [["OFFLINE","10","0"],["MESSAGE","12","HERE"]]

Output:

[0,1]

Explanation:

Initially, all users are online.

At timestamp 10,

id0

goes

offline.

At timestamp 12,

"HERE"

is mentioned. Because

id0

is still offline, they will not be mentioned.

mentions = [0,1]

Constraints:

1 <= numberOfUsers <= 100

1 <= events.length <= 100

events[i].length == 3

events[i][0]

will be one of

MESSAGE

or

OFFLINE

.

1 <= int(events[i][1]) <= 10

5

The number of

id<number>

mentions in any

"MESSAGE"

event is between

1

and

100

.

0 <= <number> <= numberOfUsers - 1

It is

guaranteed

that the user id referenced in the

OFFLINE

event is

online

at the time the event occurs.

Code Snippets

C++:

```
class Solution {
public:
vector<int> countMentions(int numberOfUsers, vector<vector<string>>& events)
{

}
};
```

Java:

```
class Solution {
public int[] countMentions(int numberOfUsers, List<List<String>> events) {

}
}
```

Python3:

```
class Solution:
def countMentions(self, numberOfUsers: int, events: List[List[str]]) ->
```

```
List[int]:
```

Python:

```
class Solution(object):
def countMentions(self, numberOfUsers, events):
"""
:type numberOfUsers: int
:type events: List[List[str]]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number} numberOfUsers
* @param {string[][]} events
* @return {number[]}
*/
var countMentions = function(numberOfUsers, events) {

};
```

TypeScript:

```
function countMentions(numberOfUsers: number, events: string[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] CountMentions(int numberOfUsers, IList<IList<string>> events) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countMentions(int numberOfUsers, char*** events, int eventsSize, int*
```

```
eventsColSize, int* returnSize) {

}
```

Go:

```
func countMentions(numberOfUsers int, events [][]string) []int {

}
```

Kotlin:

```
class Solution {
fun countMentions(numberOfUsers: Int, events: List<List<String>>): IntArray {

}
}
```

Swift:

```
class Solution {
func countMentions(_ numberOfUsers: Int, _ events: [[String]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn count_mentions(number_of_users: i32, events: Vec<Vec<String>>) ->
Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer} number_of_users
# @param {String[][]} events
# @return {Integer[]}
def count_mentions(number_of_users, events)

end
```

PHP:

```
class Solution {

/**
* @param Integer $numberOfUsers
* @param String[][] $events
* @return Integer[]
*/
function countMentions($numberOfUsers, $events) {

}
}
```

Dart:

```
class Solution {
List<int> countMentions(int numberOfUsers, List<List<String>> events) {

}
}
```

Scala:

```
object Solution {
def countMentions(numberOfUsers: Int, events: List[List[String]]): Array[Int]
= {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_mentions(number_of_users :: integer, events :: [[String.t]]) ::
[integer]
def count_mentions(number_of_users, events) do

end
end
```

Erlang:

```
-spec count_mentions(NumberOfUsers :: integer(), Events ::
[[unicode:unicode_binary()]]) -> [integer()].
count_mentions(NumberOfUsers, Events) ->
.
```

Racket:

```
(define/contract (count-mentions numberOfUsers events)
(-> exact-integer? (listof (listof string?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<int> countMentions(int numberOfUsers, vector<vector<string>>& events)
{

}
};
```

Java Solution:

```
/**
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
* Approach: Use two pointers or sliding window technique
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int[] countMentions(int numberOfUsers, List<List<String>> events) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Mentions Per User
Difficulty: Medium
Tags: array, string, math, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countMentions(self, numberOfUsers: int, events: List[List[str]]) ->
List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countMentions(self, numberOfUsers, events):
"""
:type numberOfUsers: int
:type events: List[List[str]]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} numberOfUsers
* @param {string[][]} events
* @return {number[]}
*/
var countMentions = function(numberOfUsers, events) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countMentions(numberOfUsers: number, events: string[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int[] CountMentions(int numberOfUsers, IList<IList<string>> events) {

}
}
```

C Solution:

```
/*
* Problem: Count Mentions Per User
* Difficulty: Medium
* Tags: array, string, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countMentions(int numberOfUsers, char*** events, int eventsSize, int*
eventsColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Count Mentions Per User
// Difficulty: Medium
// Tags: array, string, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach
```

```
func countMentions(numberOfUsers int, events [][]string) []int {

}
```

Kotlin Solution:

```
class Solution {
fun countMentions(numberOfUsers: Int, events: List<List<String>>): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func countMentions(_ numberOfUsers: Int, _ events: [[String]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Count Mentions Per User
// Difficulty: Medium
// Tags: array, string, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_mentions(number_of_users: i32, events: Vec<Vec<String>>) ->
Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer} number_of_users
# @param {String[][]} events
```

```
# @return {Integer[]}
def count_mentions(number_of_users, events)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $numberOfUsers
* @param String[][] $events
* @return Integer[]
*/
function countMentions($numberOfUsers, $events) {

}
}
```

Dart Solution:

```
class Solution {
List<int> countMentions(int numberOfUsers, List<List<String>> events) {

}
}
```

Scala Solution:

```
object Solution {
def countMentions(numberOfUsers: Int, events: List[List[String]]): Array[Int]
= {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_mentions(number_of_users :: integer, events :: [[String.t]]) ::
[integer]
```

```
def count_mentions(number_of_users, events) do

end
end
```

Erlang Solution:

```
-spec count_mentions(NumberOfUsers :: integer(), Events ::
[[unicode:unicode_binary()]]) -> [integer()].
count_mentions(NumberOfUsers, Events) ->
.
```

Racket Solution:

```
(define/contract (count-mentions numberOfUsers events)
(-> exact-integer? (listof (listof string?)) (listof exact-integer?))
)
```