Problem 1817: Finding the Users Active Minutes

Problem Information

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

Problem Description

You are given the logs for users' actions on LeetCode, and an integer

k

. The logs are represented by a 2D integer array

logs

where each

logs[i] = [ID

i

, time

i

]

indicates that the user with

ID

i

performed an action at the minute

time

i

.

Multiple users

can perform actions simultaneously, and a single user can perform

multiple actions

in the same minute.

The

user active minutes (UAM)

for a given user is defined as the

number of unique minutes

in which the user performed an action on LeetCode. A minute can only be counted once, even if multiple actions occur during it.

You are to calculate a

1-indexed

array

answer

of size

k

such that, for each

j

(

1 <= j <= k

),

answer[j]

is the

number of users

whose

UAM

equals

j

.

Return

the array

answer

as described above

.

Example 1:

Input:

logs = [[0,5],[1,2],[0,2],[0,5],[1,3]], k = 5

Output:

[0,2,0,0,0]

Explanation:

The user with ID=0 performed actions at minutes 5, 2, and 5 again. Hence, they have a UAM of 2 (minute 5 is only counted once). The user with ID=1 performed actions at minutes 2 and 3. Hence, they have a UAM of 2. Since both users have a UAM of 2, answer[2] is 2, and the remaining answer[j] values are 0.

Example 2:

Input:

logs = [[1,1],[2,2],[2,3]], k = 4

Output:

[1,1,0,0]

Explanation:

The user with ID=1 performed a single action at minute 1. Hence, they have a UAM of 1. The user with ID=2 performed actions at minutes 2 and 3. Hence, they have a UAM of 2. There is one user with a UAM of 1 and one with a UAM of 2. Hence, answer[1] = 1, answer[2] = 1, and the remaining values are 0.

Constraints:

1 <= logs.length <= 10

4

0 <= ID

i

<= 10

9

1 <= time

i

<= 10

5

k

is in the range

[The maximum

UAM

for a user, 10

5

]

.

Code Snippets

C++:

```
class Solution {
public:
vector<int> findingUsersActiveMinutes(vector<vector<int>>& logs, int k) {
```

```
}
};
```

Java:

```
class Solution {
public int[] findingUsersActiveMinutes(int[][] logs, int k) {

}
}
```

Python3:

```
class Solution:
def findingUsersActiveMinutes(self, logs: List[List[int]], k: int) ->
List[int]:
```

Python:

```
class Solution(object):
def findingUsersActiveMinutes(self, logs, k):
"""
:type logs: List[List[int]]
:type k: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[][]} logs
* @param {number} k
* @return {number[]}
*/
var findingUsersActiveMinutes = function(logs, k) {

};
```

TypeScript:

```
function findingUsersActiveMinutes(logs: number[][], k: number): number[] {
```

```
};
```

C#:

```
public class Solution {
public int[] FindingUsersActiveMinutes(int[][] logs, int k) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findingUsersActiveMinutes(int** logs, int logsSize, int* logsColSize,
int k, int* returnSize) {

}
```

Go:

```
func findingUsersActiveMinutes(logs [][]int, k int) []int {

}
```

Kotlin:

```
class Solution {
fun findingUsersActiveMinutes(logs: Array<IntArray>, k: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func findingUsersActiveMinutes(_ logs: [[Int]], _ k: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn finding_users_active_minutes(logs: Vec<Vec<i32>>, k: i32) -> Vec<i32>
{

}
}
```

Ruby:

```
# @param {Integer[][]} logs
# @param {Integer} k
# @return {Integer[]}
def finding_users_active_minutes(logs, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $logs
* @param Integer $k
* @return Integer[]
*/
function findingUsersActiveMinutes($logs, $k) {

}
}
```

Dart:

```
class Solution {
List<int> findingUsersActiveMinutes(List<List<int>> logs, int k) {

}
}
```

Scala:

```
object Solution {
def findingUsersActiveMinutes(logs: Array[Array[Int]], k: Int): Array[Int] =
{

}
}
```

Elixir:

```
defmodule Solution do
@spec finding_users_active_minutes(logs :: [[integer]], k :: integer) ::
[integer]
def finding_users_active_minutes(logs, k) do

end
end
```

Erlang:

```
-spec finding_users_active_minutes(Logs :: [[integer()]], K :: integer()) ->
[integer()].
finding_users_active_minutes(Logs, K) ->
.
```

Racket:

```
(define/contract (finding-users-active-minutes logs k)
(-> (listof (listof exact-integer?)) exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
vector<int> findingUsersActiveMinutes(vector<vector<int>>& logs, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int[] findingUsersActiveMinutes(int[][] logs, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Finding the Users Active Minutes
Difficulty: Medium
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
```

```
def findingUsersActiveMinutes(self, logs: List[List[int]], k: int) ->
List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findingUsersActiveMinutes(self, logs, k):
"""
:type logs: List[List[int]]
:type k: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[][]} logs
* @param {number} k
* @return {number[]}
*/
var findingUsersActiveMinutes = function(logs, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
```

```
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findingUsersActiveMinutes(logs: number[][], k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int[] FindingUsersActiveMinutes(int[][] logs, int k) {

}
}
```

C Solution:

```
/*
* Problem: Finding the Users Active Minutes
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findingUsersActiveMinutes(int** logs, int logsSize, int* logsColSize,
int k, int* returnSize) {

}
```

Go Solution:

```
// Problem: Finding the Users Active Minutes
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findingUsersActiveMinutes(logs [][]int, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun findingUsersActiveMinutes(logs: Array<IntArray>, k: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func findingUsersActiveMinutes(_ logs: [[Int]], _ k: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Finding the Users Active Minutes
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn finding_users_active_minutes(logs: Vec<Vec<i32>>, k: i32) -> Vec<i32>
{

}
}
```

Ruby Solution:

```
# @param {Integer[][]} logs
# @param {Integer} k
# @return {Integer[]}
def finding_users_active_minutes(logs, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $logs
* @param Integer $k
* @return Integer[]
*/
function findingUsersActiveMinutes($logs, $k) {

}
}
```

Dart Solution:

```
class Solution {
List<int> findingUsersActiveMinutes(List<List<int>> logs, int k) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def findingUsersActiveMinutes(logs: Array[Array[Int]], k: Int): Array[Int] =
{

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec finding_users_active_minutes(logs :: [[integer]], k :: integer) ::
[integer]
def finding_users_active_minutes(logs, k) do

end
end
```

Erlang Solution:

```
-spec finding_users_active_minutes(Logs :: [[integer()]], K :: integer()) ->
[integer()].
finding_users_active_minutes(Logs, K) ->
.
```

Racket Solution:

```
(define/contract (finding-users-active-minutes logs k)
(-> (listof (listof exact-integer?)) exact-integer? (listof exact-integer?))
)
```