3433. Count Mentions Per User

Solved

Medium | Topics | Companies | Hint

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:

1. **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.
2. **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.

Python:

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

        mentions = [0] * numberOfmentioned
```

```
        online   = [1] * numberOfmentioned
        users = range(numberOfmentioned)

        events.sort(key = lambda x: (int(x[1]), x[0] == "MESSAGE"))

        for action, stamp, mentioned in events:

            if action == "MESSAGE":

                if mentioned == "ALL":
                    for user in users:
                        mentions[user] += 1

                elif mentioned == "HERE":
                    for user in users:
                        if online[user] <= int(stamp):
                            mentions[user]+= 1

                else: # MESSAGE with id string
                    for id in mentioned.replace('id','').split(" "):
                        mentions[int(id)]+= 1

            else:   # OFFLINE
                online[int(mentioned)] = int(stamp) + 60

        return mentions
```

JavaScript:

```
const countMentions = (numberOfUsers, events) => {
    const mentions = new Array(numberOfUsers).fill(0);
    const offline = mentions.slice();

    events.sort(([m1, t1], [m2, t2]) => (t1 - t2) || (m1 > m2 ? -1 : 1));
    for (let [event, time, users] of events) {
        if (event === 'MESSAGE') {
            if (users === 'HERE') {
                mentions.map((_, i, a) => offline[i] <= +time && a[i]++);
            } else if (users === 'ALL') {
                mentions.map((_, i, a) => a[i]++);
            } else {
                users.split(' ').map(idN => mentions[+idN.slice(2)]++);
            }
        } else {
            offline[+users] = +time + 60;
```

```
        }
    }

    return mentions ;
};
```

Java:

```
class Solution {
    public int[] countMentions(int numberOfUsers, List<List<String>> events) {
        int[] mentions=new int[numberOfUsers];
        int[] offTime=new int[numberOfUsers];

        Collections.sort(events, (a,b)->Integer.parseInt(a.get(1))==Integer.parseInt(b.get(1))?
b.get(0).compareTo(a.get(0)): Integer.parseInt(a.get(1))-Integer.parseInt(b.get(1))
                        );

        for(int i=0; i<events.size(); i++){
            if(events.get(i).get(0).equals("MESSAGE")){
                messageFunc(events.get(i), mentions, offTime);
            }
            else if(events.get(i).get(0).equals("OFFLINE")){
                offlineFunc(events.get(i), mentions, offTime);
            }
        }

        return mentions;
    }

    void messageFunc(List<String> event, int[] mentions, int[] offTime){
        int time=Integer.parseInt(event.get(1));
        String[] str=event.get(2).split(" ");

        for(String s:str){
            if(s.equals("ALL")){
                for(int i=0; i<mentions.length; i++){
                    mentions[i]+=1;
                }
            }
            else if(s.equals("HERE")){
                for(int i=0; i<mentions.length; i++){
                    if(offTime[i]==0){ //Only online
                        mentions[i]+=1;
                    }
                    else if(offTime[i]+60<=time){
```

```
                    mentions[i]+=1;
                    offTime[i]=0;
                }
            }
        }
        else{
            int idx=Integer.parseInt( s.substring("id".length()) );
            mentions[idx]+=1;
        }
    }
  }

  void offlineFunc(List<String> event, int[] mentions, int[] offTime){
    int time=Integer.parseInt(event.get(1));
    String[] str=event.get(2).split(" ");

    for(String s:str){
        int idx=Integer.parseInt(s);
        offTime[idx]=time;
    }
  }
}
```