Gmail

William Ting <williamdjting@gmail.com>

FW: Call Counter Solution

1 message

William Ting <william_ting_2@sfu.ca> Mon, May 22, 2023 at 5:10 AM
To: "williamdjting@gmail.com" <williamdjting@gmail.com>

From: Daily Byte
Sent: Monday, May 22, 2023 5:10:07 AM (UTC-08:00) Pacific Time (US & Canada)
To: William Ting
Subject: Call Counter Solution

The Daily Byte

Hey,

We hope you have been enjoying practicing with these daily problems as much as we have had preparing them for you! We just wanted to let you know a few small pieces of information regarding what is coming up in the next few days. We know you are on the free plan so you have not been getting the topic introduction emails but you are about to receive your last question on stacks and queues and will be moving onto trees next.

We have not really advertised this aspect but we wanted to let you know that when you sign up for a premium account you will receive an email with the solutions to the last 30 problems you have seen, right now you are on day 28 so in a few days you will no longer be able to get all of the solutions. We also wanted to take this time to go over the solution for a problem we particularly enjoyed that you received recently named Call Counter. To recap the problem:

This question is asked by Google. Create a class `CallCounter` that tracks the number of calls a client has made within the last 3 seconds. Your class should contain one method, `ping(int t)` that receives the current timestamp (in milliseconds) of a new call being made and returns the number of calls made within the

last 3 seconds.
Note: you may assume that the time associated with each subsequent call to
`ping` is strictly increasing.

Ex: Given the following calls to
`ping` ...

```
ping(1), return 1 (1 call within the last 3 seconds)
ping(300), return 2 (2 calls within the last 3 seconds)
ping(3000), return 3 (3 calls within the last 3 seconds)
ping(3002), return 3 (3 calls within the last 3 seconds)
ping(7000), return 1 (1 call within the last 3 seconds)
```

Solution

To solve this problem, we can use a queue to keep track of our calls to `ping` . Each call made to
`ping` contains a unique, increasing, timestamp in milliseconds which allows for easy determination of what calls have occurred within our 3-second timespan. For each call made, we can add it to our queue and then remove any calls that are currently stored within our queue that have occurred more than 3 seconds ago (i.e. 3000+ milliseconds ago). After we've removed such calls, we simply need to return the size of our queue as the calls remaining in our queue represent the calls that have been made within the last 3 seconds.

```
class RecentCounter {
 Queue<Integer> queue;

  public RecentCounter() {
      queue = new LinkedList<>();
  }

  public int ping(int t) {
      queue.add(t);
      while (queue.peek() < t - 3000) {
          queue.remove();
      }

      return queue.size();
  }


}
```

Big-O Analysis

Runtime: O(N) where
`N` is the number of queries made to `ping`.
Space complexity: generally speaking O(W) where
`W` is our window size but this can be considered O(1) in our case since our window size is fixed at 3 seconds.

Thanks,
The Daily Byte