359. Logger Rate Limiter

Easy 1016 151 Add to List Share

Design a logger system that receives a stream of messages along with their timestamps. Each **unique** message should only be printed **at most every 10 seconds** (i.e. a message printed at timestamp `t` will prevent other identical messages from being printed until timestamp `t + 10`).

All messages will come in chronological order. Several messages may arrive at the same timestamp.

Implement the `Logger` class:

- `Logger()` Initializes the `logger` object.
- `bool shouldPrintMessage(int timestamp, string message)` Returns `true` if the `message` should be printed in the given `timestamp`, otherwise returns `false`.

Example 1:

```
Input
["Logger", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage",
"shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage"]
[[], [1, "foo"], [2, "bar"], [3, "foo"], [8, "bar"], [10, "foo"], [11, "foo"]]
Output
[null, true, true, false, false, false, true]

Explanation
Logger logger = new Logger();
logger.shouldPrintMessage(1, "foo");  // return true, next allowed timestamp for
"foo" is 1 + 10 = 11
logger.shouldPrintMessage(2, "bar");  // return true, next allowed timestamp for
"bar" is 2 + 10 = 12
logger.shouldPrintMessage(3, "foo");  // 3 < 11, return false
logger.shouldPrintMessage(8, "bar");  // 8 < 12, return false
logger.shouldPrintMessage(10, "foo"); // 10 < 11, return false
logger.shouldPrintMessage(11, "foo"); // 11 >= 11, return true, next allowed
timestamp for "foo" is 11 + 10 = 21
```

Constraints:

- `0 <= timestamp <= 10^9`
- Every `timestamp` will be passed in non-decreasing order (chronological order).
- `1 <= message.length <= 30`
- At most `10^4` calls will be made to `shouldPrintMessage`.

Accepted 187,145 | Submissions 252,658

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google | 28 Atlassian | 7 Microsoft | 4 Apple | 2

Related Topics

Hash Table Design

Similar Questions

Design Hit Counter Medium

Java Autocomplete

```java
class Logger {
  private HashMap<String, Integer> msgDict;

  /** Initialize your data structure here. */
  public Logger() {
    msgDict = new HashMap<String, Integer>();
  }

  /**
   * Returns true if the message should be printed in the given timestamp, otherwise returns false.
   */
  public boolean shouldPrintMessage(int timestamp, String message) {

    if (!this.msgDict.containsKey(message)) {
      this.msgDict.put(message, timestamp);
      return true;
    }

    Integer oldTimestamp = this.msgDict.get(message);
    if (timestamp - oldTimestamp >= 10) {
      this.msgDict.put(message, timestamp);
      return true;
    } else
      return false;
  }
}
```