LeetCode
Explore
Problems
Interview New
Contest
Discuss
Store
🎁 LeetCoding Challenge + GIVEAWAY! 🎁
Description
Solution
Discuss (193)
Submissions
Java
Autocomplete

635. Design Log Storage System

Medium 382 159 Add to List Share

You are given several logs, where each log contains a unique ID and timestamp. Timestamp is a string that has the following format: `Year:Month:Day:Hour:Minute:Second`, for example, `2017:01:01:23:59:59`. All domains are zero-padded decimal numbers.

Implement the `LogSystem` class:

- `LogSystem()` Initializes the `LogSystem` object.
- `void put(int id, string timestamp)` Stores the given log `(id, timestamp)` in your storage system.
- `int[] retrieve(string start, string end, string granularity)` Returns the IDs of the logs whose timestamps are within the range from `start` to `end` inclusive. `start` and `end` all have the same format as `timestamp`, and `granularity` means how precise the range should be (i.e. to the exact `Day`, `Minute`, etc.). For example, `start = "2017:01:01:23:59:59"`, `end = "2017:01:02:23:59:59"`, and `granularity = "Day"` means that we need to find the logs within the inclusive range from **Jan. 1st 2017** to **Jan. 2nd 2017**, and the `Hour`, `Minute`, and `Second` for each log entry can be ignored.

Example 1:

```
Input
["LogSystem", "put", "put", "put", "retrieve", "retrieve"]
[[], [1, "2017:01:01:23:59:59"], [2, "2017:01:01:22:59:59"], [3,
"2016:01:01:00:00:00"], ["2016:01:01:01:01:01", "2017:01:01:23:00:00", "Year"],
["2016:01:01:01:01:01", "2017:01:01:23:00:00", "Hour"]]
Output
[null, null, null, null, [3, 2, 1], [2, 1]]

Explanation
LogSystem logSystem = new LogSystem();
logSystem.put(1, "2017:01:01:23:59:59");
logSystem.put(2, "2017:01:01:22:59:59");
logSystem.put(3, "2016:01:01:00:00:00");

// return [3,2,1], because you need to return all logs between 2016 and 2017.
logSystem.retrieve("2016:01:01:01:01:01", "2017:01:01:23:00:00", "Year");

// return [2,1], because you need to return all logs between Jan. 1, 2016 01:XX:XX
and Jan. 1, 2017 23:XX:XX.
// Log 3 is not returned because Jan. 1, 2016 00:00:00 comes before the start of
the range.
logSystem.retrieve("2016:01:01:01:01:01", "2017:01:01:23:00:00", "Hour");
```

Constraints:

- `1 <= id <= 500`
- `2000 <= Year <= 2017`
- `1 <= Month <= 12`
- `1 <= Day <= 31`
- `0 <= Hour <= 23`
- `0 <= Minute, Second <= 59`
- `granularity` is one of the values `["Year", "Month", "Day", "Hour", "Minute", "Second"]`.
- At most `500` calls will be made to `put` and `retrieve`.

Accepted 27,119 Submissions 43,977

Seen this question in a real interview before? Yes No

Companies i

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

Twitter 7

Related Topics

Hash Table String Design Ordered Set

Similar Questions

Design In-Memory File System Hard

```java
public class LogSystem {
    TreeMap < Long, Integer > map;
    public LogSystem() {
        map = new TreeMap < Long, Integer > ();
    }

    public void put(int id, String timestamp) {
        int[] st = Arrays.stream(timestamp.split(":")).mapToInt(Integer::parseInt).toArray();
        map.put(convert(st), id);
    }
    public long convert(int[] st) {
        st[1] = st[1] - (st[1] == 0 ? 0 : 1);
        st[2] = st[2] - (st[2] == 0 ? 0 : 1);
        return (st[0] - 1999L) * (31 * 12) * 24 * 60 * 60 + st[1] * 31 * 24 * 60 * 60 + st[2] * 24 * 60 * 60 + st[3] * 60 * 60 + st[4] * 60 + st[5];
    }
    public List < Integer > retrieve(String s, String e, String gra) {
        ArrayList < Integer > res = new ArrayList();
        long start = granularity(s, gra, false);
        long end = granularity(e, gra, true);
        for (long key: map.tailMap(start).keySet()) {
            if (key >= start && key < end)
                res.add(map.get(key));
        }
        return res;
    }

    public long granularity(String s, String gra, boolean end) {
        HashMap < String, Integer > h = new HashMap();
        h.put("Year", 0);
        h.put("Month", 1);
        h.put("Day", 2);
        h.put("Hour", 3);
        h.put("Minute", 4);
        h.put("Second", 5);
        String[] res = new String[] {"1999", "00", "00", "00", "00", "00"};
        String[] st = s.split(":");
        for (int i = 0; i <= h.get(gra); i++) {
            res[i] = st[i];
        }
        int[] t = Arrays.stream(res).mapToInt(Integer::parseInt).toArray();
        if (end)
            t[h.get(gra)]++;
        return convert(t);
    }
}
```