981. Time Based Key-Value Store

Created	@October 15, 2022 4:43 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/time-based-key-value-store/
Importance	
Tag	Binary search NEET
Video	

Design a time-based key-value data structure that can store multiple values for the same key at different time stamps and retrieve the key's value at a certain timestamp.

Implement the `TimeMap` class:

- `TimeMap()` Initializes the object of the data structure.
- `void set(String key, String value, int timestamp)` Stores the key `key` with the value `value` at the given time `timestamp`.
- `String get(String key, int timestamp)` Returns a value such that `set` was called previously, with `timestamp_prev <= timestamp`. If there are multiple such values, it returns the value associated with the largest `timestamp_prev`. If there are no values, it returns `""`.

Example 1:

```
Input
["TimeMap", "set", "get", "get", "set", "get", "get"]
[[], ["foo", "bar", 1], ["foo", 1], ["foo", 3], ["foo", "bar2", 4], ["foo", 4], ["foo",
 5]]
Output
[null, null, "bar", "bar", null, "bar2", "bar2"]

Explanation
TimeMap timeMap = new TimeMap();
timeMap.set("foo", "bar", 1);  // store the key "foo" and value "bar" along with timestamp
= 1.
timeMap.get("foo", 1);         // return "bar"
timeMap.get("foo", 3);         // return "bar", since there is no value corresponding to f
```

```
oo at timestamp 3 and timestamp 2, then the only value is at timestamp 1 is "bar".
timeMap.set("foo", "bar2", 4); // store the key "foo" and value "bar2" along with timestam
p = 4.
timeMap.get("foo", 4);         // return "bar2"
timeMap.get("foo", 5);         // return "bar2"
```

Constraints:

- `1 <= key.length, value.length <= 100`
- `key` and `value` consist of lowercase English letters and digits.
- `1 <= timestamp <= 10 7`
- All the timestamps `timestamp` of `set` are strictly increasing.
- At most `2 * 105` calls will be made to `set` and `get`

Solution

```
class TimeMap:

    def __init__(self):
        self.keyStore = {} # key: list of [val, timestamp]

    def set(self, key: str, value: str, timestamp: int) -> None:
        if key not in self.keyStore:
            self.keyStore[key] = []
        self.keyStore[key].append([value, timestamp])

    def get(self, key: str, timestamp: int) -> str:
        values = self.keyStore.get(key, [])
        if not values:
            return ''

        left, right = 0, len(values) - 1

        while left + 1 < right:
            mid = (left + right) // 2
            if values[mid][1] < timestamp:
                left = mid
            elif values[mid][1] > timestamp:
                right = mid
            else:
                return values[mid][1]

        if values[right][1] <= timestamp:
            return values[right][0]
        if values[left][1] <= timestamp:
            return values[left][0]

        return ''


# Your TimeMap object will be instantiated and called as such:
# obj = TimeMap()
# obj.set(key,value,timestamp)
# param_2 = obj.get(key,timestamp)
```

```
class TimeMap:
    def __init__(self):
        """
        Initialize your data structure here.
        """
        self.keyStore = {}  # key : list of [val, timestamp]

    def set(self, key: str, value: str, timestamp: int) -> None:
        if key not in self.keyStore:
            self.keyStore[key] = []
        self.keyStore[key].append([value, timestamp])

    def get(self, key: str, timestamp: int) -> str:
        res, values = "", self.keyStore.get(key, [])
        l, r = 0, len(values) - 1
        while l <= r:
            m = (l + r) // 2
            if values[m][1] <= timestamp:
                res = values[m][0]
                l = m + 1
            else:
                r = m - 1
        return res
```