1244. Design A Leaderboard

Medium 343 63 Add to List Share

Design a Leaderboard class, which has 3 functions:

1. `addScore(playerId, score)` : Update the leaderboard by adding `score` to the given player's score. If there is no player with such id in the leaderboard, add him to the leaderboard with the given `score` .
2. `top(K)` : Return the score sum of the top `K` players.
3. `reset(playerId)` : Reset the score of the player with the given id to 0 (in other words erase it from the leaderboard). It is guaranteed that the player was added to the leaderboard before calling this function.

Initially, the leaderboard is empty.

Example 1:

```
Input:
["Leaderboard","addScore","addScore","addScore","addScore","addScore","top","reset",

[[],[1,73],[2,56],[3,39],[4,51],[5,4],[1],[1],[2],[2,51],[3]]
Output:
[null,null,null,null,null,null,73,null,null,null,141]

Explanation:
Leaderboard leaderboard = new Leaderboard ();
leaderboard.addScore(1,73);   // leaderboard = [[1,73]];
leaderboard.addScore(2,56);   // leaderboard = [[1,73],[2,56]];
leaderboard.addScore(3,39);   // leaderboard = [[1,73],[2,56],[3,39]];
leaderboard.addScore(4,51);   // leaderboard = [[1,73],[2,56],[3,39],[4,51]];
leaderboard.addScore(5,4);    // leaderboard = [[1,73],[2,56],[3,39],[4,51],
[5,4]];
leaderboard.top(1);           // returns 73;
leaderboard.reset(1);         // leaderboard = [[2,56],[3,39],[4,51],[5,4]];
leaderboard.reset(2);         // leaderboard = [[3,39],[4,51],[5,4]];
leaderboard.addScore(2,51);   // leaderboard = [[2,51],[3,39],[4,51],[5,4]];
leaderboard.top(3);           // returns 141 = 51 + 51 + 39;
```

Constraints:

- `1 <= playerId, K <= 10000`
- It's guaranteed that `K` is less than or equal to the current number of players.
- `1 <= score <= 100`
- There will be at most `1000` function calls.

Accepted 29,597 | Submissions 43,966

Seen this question in a real interview before? Yes No

Companies

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

Bloomberg | 7 Twitter | 2

Related Topics

Hash Table Design Sorting

Hide Hint 1

What data structure can we use to keep the players' data?

Hide Hint 2

Keep a map (dictionary) of player scores.

Hide Hint 3

For each top(K) function call, find the maximum K scores and add them.

Java Autocomplete

```java
class Leaderboard {
    private HashMap<Integer, Integer> scores;
    public Leaderboard() {
        this.scores = new HashMap<Integer, Integer>();
    }
    public void addScore(int playerId, int score) {

        if (!this.scores.containsKey(playerId)) {
            this.scores.put(playerId, 0);
        }

        this.scores.put(playerId, this.scores.get(playerId) + score);
    }
    public int top(int K) {

        List<Integer> values = new ArrayList<Integer>(this.scores.values());
        Collections.sort(values, Collections.reverseOrder());

        int total = 0;
        for (int i = 0; i < K; i++) {
            total += values.get(i);
        }

        return total;
    }
    public void reset(int playerId) {
        this.scores.put(playerId, 0);
    }
}
```