Doc link:
https://docs.python.org/2/library/collections.html#collections.Counter

Complexity

`__init__` is `O(1)`
`addScore` is `O(1)`
`top` is `O(NlogK)`, can be improve to `O(N)`
`reset` is `O(1)`
Space `O(N)`

Python:

```
class Leaderboard(object):

    def __init__(self):
        self.A = collections.Counter()

    def addScore(self, playerId, score):
        self.A[playerId] += score

    def top(self, K):
        return sum(v for i,v in self.A.most_common(K))

    def reset(self, playerId):
        self.A[playerId] = 0
```


Type comment here... (Markdown is supported)

Post

orangezeit ★ 517 Last Edit: November 2, 2019 9:46 PM

It is amazing that I can always learn from your codes even for simple problems. (I think the optimal complexity for "top" should be O(K))

5 Show 8 replies Reply

MichaelZ ★ 633 November 4, 2019 8:03 AM

A C++ solution:

```
unordered_map<int, int> players;
map<int, int> scores;
Leaderboard() {
    players.clear();
    scores.clear();
}

void addScore(int playerId, int score) {
    if(players.find(playerId)!=players.end()) scores[players[playerId]]--;
```

Read More

2 Reply

dmyma ★ 15 November 3, 2019 12:13 PM

How would you improve top to O(n)?

1 Show 1 reply Reply

calvinchankf ★ 2550 Last Edit: February 21, 2020 10:14 AM

Thanks for telling us the easiest and intuitive way @lee215

I didnt know there is a method, most_common, in collections.Counter, so I did a very complicated one in my first attempt LOoooL

However, my approach works faster if top() is being called for very large number of times becos i dont use sort().
The drawback is it takes time to addScore() and reset(), not O(1) anymore.

```
"""
    1st: binary search + hashtable

    Time of addScore()      O(K), it can be O(logN) depends on language becos of array.insert()
    Time of reset()         O(K)
```

Read More

0 Reply