6417. Frequency Tracker

My Submissions (/contest/weekly-contest-344/problems/frequency-tracker/submissions/) Back to Contest (/contest/weekly-contest-344/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Design a data structure that keeps track of the values in it and answers some queries regarding their frequencies.

Implement the `FrequencyTracker` class.

- `FrequencyTracker()` : Initializes the `FrequencyTracker` object with an empty array initially.
- `void add(int number)` : Adds `number` to the data structure.
- `void deleteOne(int number)` : Deletes **one** occurence of `number` from the data structure. The data structure **may not contain** `number` , and in this case nothing is deleted.
- `bool hasFrequency(int frequency)` : Returns `true` if there is a number in the data structure that occurs `frequency` number of times, otherwise, it returns `false` .

Example 1:

```
Input
["FrequencyTracker", "add", "add", "hasFrequency"]
[[], [3], [3], [2]]
Output
[null, null, null, true]

Explanation
FrequencyTracker frequencyTracker = new FrequencyTracker();
frequencyTracker.add(3); // The data structure now contains [3]
frequencyTracker.add(3); // The data structure now contains [3, 3]
frequencyTracker.hasFrequency(2); // Returns true, because 3 occurs twice
```

Example 2:

```
Input
["FrequencyTracker", "add", "deleteOne", "hasFrequency"]
[[], [1], [1], [1]]
Output
[null, null, null, false]

Explanation
FrequencyTracker frequencyTracker = new FrequencyTracker();
frequencyTracker.add(1); // The data structure now contains [1]
frequencyTracker.deleteOne(1); // The data structure becomes empty []
frequencyTracker.hasFrequency(1); // Returns false, because the data structure is empty
```

Example 3:

```
Input
["FrequencyTracker", "hasFrequency", "add", "hasFrequency"]
[[], [2], [3], [1]]
Output
[null, false, null, true]

Explanation
FrequencyTracker frequencyTracker = new FrequencyTracker();
frequencyTracker.hasFrequency(2); // Returns false, because the data structure is empty
frequencyTracker.add(3); // The data structure now contains [3]
frequencyTracker.hasFrequency(1); // Returns true, because 3 occurs once
```

Constraints:

- $1 <= \text{number} <= 10^5$
- $1 <= \text{frequency} <= 10^5$

- At most, $2 * 10^5$ calls will be made to `add`, `deleteOne`, and `hasFrequency` in **total**.

JavaScript

```
const addOneOrManyMap = (m, x, cnt = 1) => m.set(x, m.get(x) + cnt || cnt);
const removeOneOrManyMap = (m, x, cnt = 1) => { let occ = m.get(x); occ > cnt ? m.set(x, occ - cnt) : m.delete(x); };

function FrequencyTracker() {
    let m = new Map(), fm = new Map();
    return { add, deleteOne, hasFrequency }
    function add(x) {
        let beforeOcc = m.get(x) || 0;
        addOneOrManyMap(m, x);
        let afterOcc = m.get(x) || 0;
        removeOneOrManyMap(fm, beforeOcc);
        addOneOrManyMap(fm, afterOcc);
    }
    function deleteOne(x) {
        let beforeOcc = m.get(x) || 0;
        removeOneOrManyMap(m, x);
        let afterOcc = m.get(x) || 0;
        removeOneOrManyMap(fm, beforeOcc);
        addOneOrManyMap(fm, afterOcc);
    }
    function hasFrequency(freq) {
        return fm.has(freq);
    }
}
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/945806231/) More Details (/submissions/detail/945806231/)

Share your acceptance!