5860. Find Original Array From Doubled Array

My Submissions (/contest/biweekly-contest-61/problems/find-original-array-from-doubled-array/submissions/)

Back to Contest (/contest/biweekly-contest-61/)

An integer array `original` is transformed into a **doubled** array `changed` by appending **twice the value** of every element in `original`, and then randomly **shuffling** the resulting array.

Given an array `changed`, return `original` *if* `changed` *is a* ***doubled*** *array. If* `changed` *is not a* ***doubled*** *array, return an empty array. The elements in* `original` *may be returned in* ***any*** *order.*

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

Example 1:

```
Input: changed = [1,3,4,2,6,8]
Output: [1,3,4]
Explanation: One possible original array could be [1,3,4]:
- Twice the value of 1 is 1 * 2 = 2.
- Twice the value of 3 is 3 * 2 = 6.
- Twice the value of 4 is 4 * 2 = 8.
Other original arrays could be [4,3,1] or [3,1,4].
```

Example 2:

```
Input: changed = [6,3,0,1]
Output: []
Explanation: changed is not a doubled array.
```

Example 3:

```
Input: changed = [1]
Output: []
Explanation: changed is not a doubled array.
```

Constraints:

- `1 <= changed.length <= 10^5`
- `0 <= changed[i] <= 10^5`

JavaScript

```
const counter = (a_or_s) => { let map = new Map(); for (const i of a_or_s) map.set(i, map.get(i) + 1 || 1); return map; };

const findOriginalArray = (a) => {
    a.sort((x, y) => y - x);
    // pr(a);
    let res = [];
    let m = counter(a);
    // pr(m);
    for (const [x, occ] of m) {
        if (x & 1) {
            let t = x * 2;
            if (m.has(t)) {
                let tocc = m.get(t);
                if (tocc < occ) {
```

```
                    return [];
                } else if (tocc == occ) {
                    m.delete(x)
                    m.delete(t)
                    let repeat = occ;
                    while (repeat--) res.push(x);
                } else {
                    m.delete(x)
                    m.set(t, tocc - occ);
                    let repeat = occ;
                    while (repeat--) res.push(x);
                }
            } else {
                return [];
            }
        }
    }
    // pr(m);
    for (const [x, occ] of m) {
        if (x == 0) {
            if (occ & 1) return [];
            let repeat = occ / 2;
            while (repeat--) res.push(x);
            m.delete(x);
            continue;
        }
        let t = x / 2;
        if (m.has(t)) {
            let tocc = m.get(t);
            if (tocc < occ) {
                return [];
            } else if (tocc == occ) {
                m.delete(x)
                m.delete(t)
                let repeat = occ;
                while (repeat--) res.push(t);
            } else {
                m.delete(x)
                m.set(t, tocc - occ);
                let repeat = occ;
                while (repeat--) res.push(t);
            }
        } else {
            return [];
        }
    }
    // pr(m);
    return m.size > 0 ? [] : res;
};
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/557013610/) More Details (/submissions/detail/557013610/)

Share your acceptance!

Copyright © 2021 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)