5495. Most Visited Sector in a Circular Track

My Submissions (/contest/weekly-contest-203/problems/most-visited-sector-in-a-circular-track/submissions/)

Back to Contest (/contest/weekly-contest-203/)

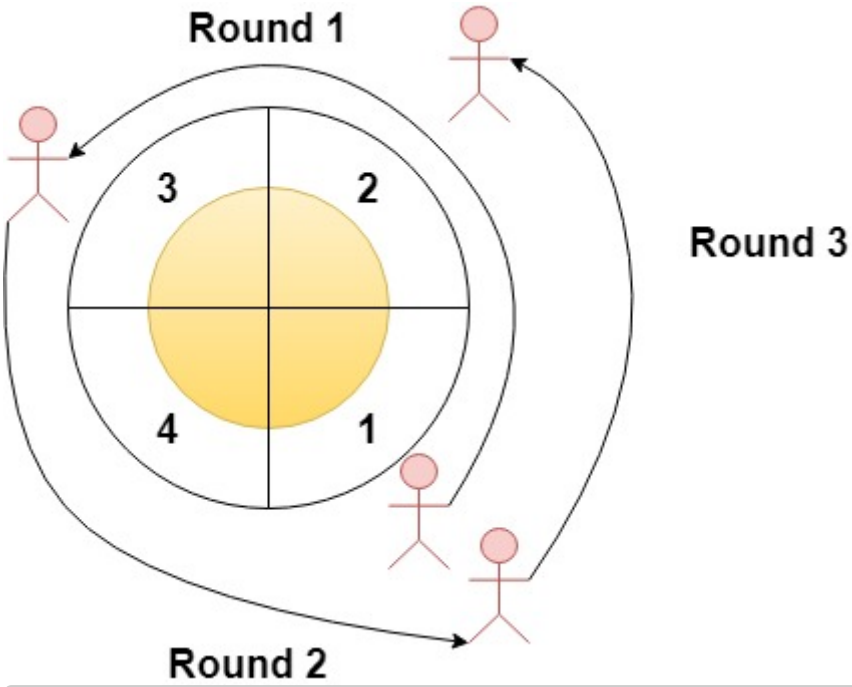
Given an integer `n` and an integer array `rounds`. We have a circular track which consists of `n` sectors labeled from `1` to `n`. A marathon will be held on this track, the marathon consists of `m` rounds. The `i^th` round starts at sector `rounds[i - 1]` and ends at sector `rounds[i]`. For example, round 1 starts at sector `rounds[0]` and ends at sector `rounds[1]`

Return *an array of the most visited sectors* sorted in **ascending** order.

Notice that you circulate the track in ascending order of sector numbers in the counter-clockwise direction (See the first example).

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

Example 1:




```
Input: n = 4, rounds = [1,3,1,2]
Output: [1,2]
Explanation: The marathon starts at sector 1. The order of the visited sectors is as follo
1 --> 2 --> 3 (end of round 1) --> 4 --> 1 (end of round 2) --> 2 (end of round 3 and the
We can see that both sectors 1 and 2 are visited twice and they are the most visited secto
```

Example 2:

```
Input: n = 2, rounds = [2,1,2,1,2,1,2,1,2]
Output: [2]
```

Example 3:

```
Input: n = 7, rounds = [1,3,5,7]
Output: [1,2,3,4,5,6,7]
```

Constraints:

- `2 <= n <= 100`
- `1 <= m <= 100`
- `rounds.length == m + 1`
- `1 <= rounds[i] <= n`
- `rounds[i] != rounds[i + 1]` for `0 <= i < m`

JavaScript

```
/**
 * @param {number} n
 * @param {number[]} rounds
 * @return {number[]}
 */
const mostVisited = (n, rounds) => {
    let data = [];
    for (let i = 1; i < rounds.length; i++) {
        let start = rounds[i - 1];
        let end = rounds[i];
        let tmp = [];
        if (start <= end) {
            for (let j = start; j <= end; j++) {
                tmp.push(j);
            }
        } else {
            for (let j = start; j <= n; j++) {
                tmp.push(j);
            }
            for (let j = 1; j <= end; j++) {
                tmp.push(j);
            }
        }
        data.push(tmp);
    }
    let res = [data[0]];
    for (let i = 1; i < data.length; i++) {
        res.push(data[i].slice(1));
    }
    let newRes = [];
```

```
    for (const r of res) {
        newRes = newRes.concat(r);
    }
    let map = new Map();
    let element = [...new Set(newRes)];
    for (const e of element) {
        map.set(e, getFrequency(newRes, e));
    }
    let arr = [...newRes].sort((a, b) => map.get(b) - map.get(a));
    let max = map.get(arr[0]);
    let ret = [];
    for (const k of map.keys()) {
        if (map.get(k) == max) {
            ret.push(k);
        }
    }
    ret.sort((a, b) => a - b);
    return ret;
};

const getFrequency = (arr, item) => {
    return arr.filter(x => x === item).length;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/384956306/)

More Details ❯ (/submissions/detail/384956306/)

Share your acceptance!