6269. Shortest Distance to Target String in a Circular Array

My Submissions (/contest/weekly-contest-325/problems/shortest-distance-to-target-string-in-a-circular-array/submissions/)

Back to Contest (/contest/weekly-contest-325/)

You are given a **0-indexed circular** string array `words` and a string `target`. A **circular array** means that the array's end connects to the array's beginning.

- Formally, the next element of `words[i]` is `words[(i + 1) % n]` and the previous element of `words[i]` is `words[(i - 1 + n) % n]`, where `n` is the length of `words`.

Starting from `startIndex`, you can move to either the next word or the previous word with `1` step at a time.

Return *the **shortest** distance needed to reach the string* `target`. If the string `target` does not exist in `words`, return `-1`.

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

Example 1:

```
Input: words = ["hello","i","am","leetcode","hello"], target = "hello", startIndex = 1
Output: 1
Explanation: We start from index 1 and can reach "hello" by
- moving 3 units to the right to reach index 4.
- moving 2 units to the left to reach index 4.
- moving 4 units to the right to reach index 0.
- moving 1 unit to the left to reach index 0.
The shortest distance to reach "hello" is 1.
```

Example 2:

```
Input: words = ["a","b","leetcode"], target = "leetcode", startIndex = 0
Output: 1
Explanation: We start from index 0 and can reach "leetcode" by
- moving 2 units to the right to reach index 3.
- moving 1 unit to the left to reach index 3.
The shortest distance to reach "leetcode" is 1.
```

Example 3:

```
Input: words = ["i","eat","leetcode"], target = "ate", startIndex = 0
Output: -1
Explanation: Since "ate" does not exist in words, we return -1.
```

Constraints:

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 100`
- `words[i]` and `target` consist of only lowercase English letters.
- `0 <= startIndex < words.length`

JavaScript

```
const counter_value_in_indexA_in = (a_or_s) => { let m = new Map(); let n = a_or_s.length; for (let i = 0; i < n; i++) {
if (!m.has(a_or_s[i])) m.set(a_or_s[i], []); m.get(a_or_s[i]).push(i); } return m; };

const closetTarget = (words, target, startIndex) => {
    let m = counter_value_in_indexA_in(words), res = Number.MAX_SAFE_INTEGER;
    if (!m.has(target)) return -1;
    let a = m.get(target);
    for (const idx of a) {
        let small = Math.min(idx, startIndex), large = Math.max(idx, startIndex);
        let dis = indexMove(words.length, small, large);
        res = Math.min(res, dis);
    }
    return res;
};

```

```
15 ▾ const indexMove = (n, small, large) => {
16       let moveRight = large - small, moveLeft = small + n - large;
17       return Math.min(moveLeft, moveRight);
18   };
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/865033186/) | More Details > (/submissions/detail/865033186/)

Share your acceptance!