2584. Split the Array to Make Coprime Products

My Submissions (/contest/weekly-contest-335/problems/split-the-array-to-make-coprime-products/submissions/)

Back to Contest (/contest/weekly-contest-335/)

User Accepted:	985
User Tried:	6602
Total Accepted:	1065
Total Submissions:	17645
Difficulty:	Hard

You are given a **0-indexed** integer array `nums` of length `n`.

A **split** at an index `i` where `0 <= i <= n - 2` is called **valid** if the product of the first `i + 1` elements and the product of the remaining elements are coprime.

- For example, if `nums = [2, 3, 3]`, then a split at the index `i = 0` is valid because `2` and `9` are coprime, while a split at the index `i = 1` is not valid because `6` and `3` are not coprime. A split at the index `i = 2` is not valid because `i == n - 1`.

Return *the smallest index* `i` *at which the array can be split validly or* `-1` *if there is no such split*.

Two values `val1` and `val2` are coprime if `gcd(val1, val2) == 1` where `gcd(val1, val2)` is the greatest common divisor of `val1` and `val2`.

Example 1:

index	prefixproduct	suffixproduct	gcd
0	4	12600	4
1	28	1800	4
2	224	225	1
3	3360	15	15
4	10080	5	5

```
Input: nums = [4,7,8,15,3,5]
Output: 2
Explanation: The table above shows the values of the product of the first i + 1 elements, the remaining elements, and their gcd
The only valid split is at index 2.
```

Example 2:

index	prefixproduct	suffixproduct	gcd
0	4	12600	4
1	28	1800	4
2	420	120	60
3	3360	15	15
4	10080	5	5

```
Input: nums = [4,7,15,8,3,5]
Output: -1
Explanation: The table above shows the values of the product of the first i + 1 elements, the remaining elements, and their gcd
There is no valid split.
```

Constraints:

- `n == nums.length`
- `1 <= n <= 10^4`
- `1 <= nums[i] <= 10^6`

Discuss (https://leetcode.com/problems/split-the-array-to-make-coprime-products/discuss)

JavaScript

```javascript
const findValidSplit = (a) => {
    let N = Math.max(...a), n = a.length, first = Array(N + 1).fill(-1), last = Array(N + 1).fill(0);
    let lpf = enumLowestPrimeFactors(N), imos = Array(n + 1).fill(0);
    for (let i = 0; i < n; i++) {
        let f = factorFast(a[i], lpf);
        for (const e of f) {
            if (first[e[0]] == -1) {
                first[e[0]] = i;
            }
            last[e[0]] = i;
        }
    }
    for (let i = 1; i <= N; i++) {
        if (first[i] != -1 && first[i] != last[i]) {
            imos[first[i]]++;
            imos[last[i]]--;
        }
    }
    for (let i = 0; i < n; i++) imos[i + 1] += imos[i];
    for (let i = 0; i < n - 1; i++) {
        if (imos[i] == 0) return i;
    }
    return -1;
};

// reference: https://www.geeksforgeeks.org/least-prime-factor-of-numbers-till-n/
function enumLowestPrimeFactors(n) {
    let lpf = Array(n + 1).fill(0);
    for (let i = 2; i <= n; i++) {
        if (lpf[i] == 0) {
            lpf[i] = i;
            for (let j = i * i; j <= n; j += i) {
                if (lpf[j] == 0) lpf[j] = i;
            }
        }
    }
    return lpf;
}

const factorFast = (n, lpf) => {
    let f = Array(9), q = 0;
    while (lpf[n] > 0) {
        let p = lpf[n];
        if (q == 0 || p != f[q - 1][0]) {
            f[q++] = [p, 1];
        } else {
            f[q - 1][1]++;
        }
        n /= p;
    }
    return f.slice(0, q); // f[0] ^ f[1] = n
};
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/913173401/) More Details (/submissions/detail/913173401/)

Share your acceptance!