6364. Count the Number of Square-Free Subsets

 $My\ Submissions\ (/contest/weekly-contest-333/problems/count-the-number-of-square-free-subsets/submissions/)$

Back to Contest (/contest/weekly-contest-333/)

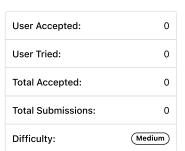
You are given a positive integer **0-indexed** array nums.

A subset of the array nums is **square-free** if the product of its elements is a **square-free integer**.

A **square-free integer** is an integer that is divisible by no square number other than 1.

Return the number of square-free non-empty subsets of the array **nums**. Since the answer may be too large, return it **modulo** $10^9 + 7$.

A **non-empty subset** of nums is an array that can be obtained by deleting some (possibly none but not all) elements from nums. Two subsets are different if and only if the chosen indices to delete are different.



Example 1:

```
Input: nums = [3,4,4,5]
Output: 3
Explanation: There are 3 square-free subsets in this example:
- The subset consisting of the 0<sup>th</sup> element [3]. The product of its elements is 3, which is a square-free integer.
- The subset consisting of the 3<sup>rd</sup> element [5]. The product of its elements is 5, which is a square-free integer.
- The subset consisting of 0<sup>th</sup> and 3<sup>rd</sup> elements [3,5]. The product of its elements is 15, which is a square-free integer.
It can be proven that there are no more than 3 square-free subsets in the given array.
```

Example 2:

```
Input: nums = [1]
Output: 1
Explanation: There is 1 square-free subset in this example:
    The subset consisting of the 0<sup>th</sup> element [1]. The product of its elements is 1, which is a square-free integer.
It can be proven that there is no more than 1 square-free subset in the given array.
```

Constraints:

- 1 <= nums.length <= 1000
- 1 <= nums[i] <= 30

```
C
JavaScript
             const ll = BigInt;
             const powmod = (a, b, mod) \Rightarrow \{ let r = 1n; while (b > 0n) \{ if (b % 2n == 1) r = r * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b >>= 1n; a = a * a % mod; b == a * a % mo
              mod; } return r; };
             const gcd = (a, b) \Rightarrow b == 0 ? a : gcd(b, a % b);
             const minus_mod = (x, y, mod) \Rightarrow ((x - y) \% mod + mod) \% mod;
   5
   6
             const mod = 1e9 + 7, bmod = ll(mod);
   7
             const squareFreeSubsets = (a) => {
                           let d = [2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30];
   8
   9
                           let f = Array(31).fill(0), dp = new Map(), sum = 0;
10
                            for (const x of a) f[x]++;
11
                           dp.set(1, Number(powmod(2n, ll(f[1]), bmod)));
12 ▼
                           for (const v of d) {
13 ▼
                                         for (const [x,] of dp) {
14 ▼
                                                       if (\gcd(v, x) == 1) {
                                                                     let occ = dp.qet(x * v) | | 0, occ2 = dp.qet(x) | | 0;
15
16
                                                                    dp.set(x * v, (occ + occ2 * f[v]) % mod);
17
                                                       }
18
                                         }
19
20
                           for (const [, v] of dp) sum += v;
21
                            return minus_mod(sum, 1, mod);
22
             };
```

Custom Testcase

Use Example Testcases

Run

Accepted (/submissions/detail/902713109/)

More Details ➤ (/submissions/detail/902713109/)

Share your acceptance!

Copyright © 2023 LeetCode

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

United States (/region)