2501. Longest Square Streak in an Array

You are given an integer array `nums`. A subsequence of `nums` is called a **square streak** if:

- The length of the subsequence is at least `2`, and
- **after** sorting the subsequence, each element (except the first element) is the **square** of the previous number.

Return *the length of the* ***longest square streak*** *in* `nums`*, or return* `-1` *if there is no* ***square streak***.

A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

```
Input: nums = [4,3,6,16,8,2]
Output: 3
Explanation: Choose the subsequence [4,16,2]. After sorting it, it becomes
[2,4,16].
- 4 = 2 * 2.
- 16 = 4 * 4.
Therefore, [4,16,2] is a square streak.
It can be shown that every subsequence of length 4 is not a square streak.
```

Example 2:

```
Input: nums = [2,3,5,6,7]
Output: -1
Explanation: There is no square streak in nums so return -1.
```

Constraints:

- `2 <= nums.length <= 10^5`
- `2 <= nums[i] <= 10^5`

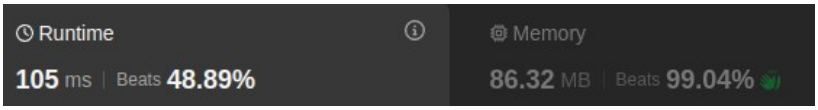
2501. Longest Square Streak in an Array

/*

Binary search without knowledge of processed elements

Time complexity: O(nlogn)

Space complexity: O(1)



*/

```
class Solution {
public:
    int longestSquareStreak(std::vector<int>& nums) {
        int n=nums.size();
        std::sort(nums.begin(),nums.end());
        int ans=0;
        for(auto& e: nums){
            int i=std::lower_bound(nums.begin(),nums.end(),e)-nums.begin();
            int cnt=0;
            long long x=e;
            while(i<n && x==nums[i]){
                x*=x;
                i=std::lower_bound(nums.begin(),nums.end(),x)-nums.begin();
                cnt++;
            }
            ans=std::max(ans,cnt);
        }
        return ans>1?ans:-1;
    }
};
```

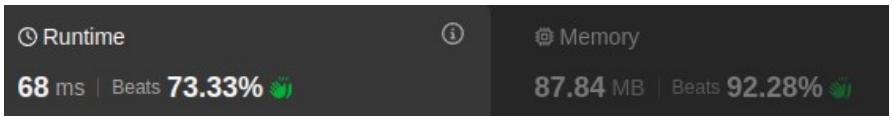
2501. Longest Square Streak in an Array

/*

Binary search with knowledge of processed elements

Time complexity: O(nlogn)

Space complexity: O(n)



*/

```
class Solution {
public:
    int longestSquareStreak(std::vector<int>& nums) {
        int n=nums.size();
        std::sort(nums.begin(),nums.end());
        int ans=0;
        std::vector<bool> processed(100001,false);
        for(auto& e: nums){
            if(processed[e]) continue;
            int i=std::lower_bound(nums.begin(),nums.end(),e)-nums.begin();
            int cnt=0;
            long long x=e;
            while(i<n && !processed[x] && x==nums[i]){
                processed[x]=true;
                x*=x;
                i=std::lower_bound(nums.begin(),nums.end(),x)-nums.begin();
                cnt++;
            }
            ans=std::max(ans,cnt);
        }
        return ans>1?ans:-1;
    }
};
```

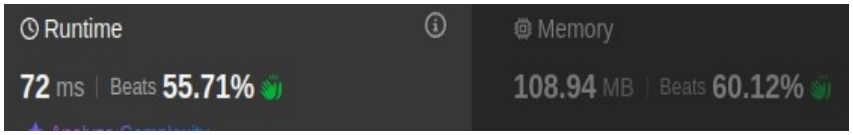
2501. Longest Square Streak in an Array

/*

Unordered Set

Time complexity: $\Theta(n), O(n^2)$

Space complexity: O(n)



*/

```
class Solution {
public:
    int longestSquareStreak(std::vector<int>& nums) {
        int n=nums.size();
        int mx=*std::max_element(nums.begin(),nums.end());
        std::unordered_set<int> numbers(nums.begin(),nums.end());
        int ans=0;
        for(auto& e: numbers){
            int cnt=0;
            long long x=e;
            while(x<=mx && numbers.find(x)!=numbers.end()){
                x*=x;
                cnt++;
            }
            ans=std::max(ans,cnt);
        }
        return ans>1?ans:-1;
    }
};
```