

2154. Keep Multiplying Found Values by Two

Solved

[Easy](#) [Topics](#) [Companies](#) [Hint](#)

You are given an array of integers `nums`. You are also given an integer `original` which is the first number that needs to be searched for in `nums`.

You then do the following steps:

1. If `original` is found in `nums`, **multiply** it by two (i.e., set `original = 2 * original`).
2. Otherwise, **stop** the process.
3. **Repeat** this process with the new number as long as you keep finding the number.

Return *the final value of* `original`.

Example 1:

Input: `nums = [5,3,6,1,12]`, `original = 3`

Output: 24

Explanation:

- 3 is found in `nums`. 3 is multiplied by 2 to obtain 6.
- 6 is found in `nums`. 6 is multiplied by 2 to obtain 12.
- 12 is found in `nums`. 12 is multiplied by 2 to obtain 24.
- 24 is not found in `nums`. Thus, 24 is returned.

Example 2:

Input: `nums = [2,7,9]`, `original = 4`

Output: 4

Explanation:

- 4 is not found in `nums`. Thus, 4 is returned.

Constraints:

- `1 <= nums.length <= 1000`
- `1 <= nums[i], original <= 1000`

Python:

```
class Solution:
    def findFinalValue(self, nums: list[int], k: int) -> int:
        bits = 0
        for num in nums:
            if num % k != 0:
                continue
            n = num // k
            if n & (n - 1) == 0:
                bits |= n
        d = bits + 1
        return k * (d & -d)
```

JavaScript:

```
const findFinalValue = (nums, k) => {
    let bits = 0;
    for (let num of nums) {
        if (num % k !== 0) continue;
        const n = num / k;
        if ((n & (n - 1)) === 0)
            bits |= n;
    }

    const d = bits + 1;

    return k * (d & -d);
};
```

Java:

```
class Solution {
    public int findFinalValue(int[] nums, int k) {
        int bits = 0;
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] % k != 0) continue;
            nums[i] = nums[i] / k;
            if ((nums[i] & (nums[i] - 1)) == 0)
                bits |= nums[i];
        }
        int d = bits + 1;
        return k * (d & -d);
    }
}
```