Problem 3141: Maximum Hamming Distances

Problem Information

Difficulty: Hard
Acceptance Rate: 48.25%
Paid Only: Yes
Tags: Array, Bit Manipulation, Breadth-First Search

Problem Description

Given an array `nums` and an integer `m`, with each element `nums[i]` satisfying `0 <= nums[i] < 2m`, return an array `answer`. The `answer` array should be of the same length as `nums`, where each element `answer[i]` represents the _maximum_ **Hamming distance** between `nums[i]` and any other element `nums[j]` in the array.

The **Hamming distance** between two binary integers is defined as the number of positions at which the corresponding bits differ (add leading zeroes if needed).

Example 1:

Input: nums = [9,12,9,11], m = 4

Output: [2,3,2,3]

Explanation:

The binary representation of `nums = [1001,1100,1001,1011]`.

The maximum hamming distances for each index are:

* `nums[0]`: 1001 and 1100 have a distance of 2. * `nums[1]`: 1100 and 1011 have a distance of 3. * `nums[2]`: 1001 and 1100 have a distance of 2. * `nums[3]`: 1011 and 1100 have a distance of 3.

Example 2:

Input: nums = [3,4,6,10], m = 4

Output: [3,3,2,3]

Explanation:

The binary representation of `nums = [0011,0100,0110,1010]`.

The maximum hamming distances for each index are:

* `nums[0]`: 0011 and 0100 have a distance of 3. * `nums[1]`: 0100 and 0011 have a distance of 3. * `nums[2]`: 0110 and 1010 have a distance of 2. * `nums[3]`: 1010 and 0100 have a distance of 3.

Constraints:

* `1 <= m <= 17` * `2 <= nums.length <= 2m` * `0 <= nums[i] < 2m`

Code Snippets

C++:

```
class Solution {
public:
vector<int> maxHammingDistances(vector<int>& nums, int m) {

}
};
```

Java:

```
class Solution {
public int[] maxHammingDistances(int[] nums, int m) {

}
}
```

Python3:

```
class Solution:
def maxHammingDistances(self, nums: List[int], m: int) -> List[int]:
```