https://leetcode.com/problems/maximum-product-of-word-lengths/

Given a string array `words`, find the maximum value of `length(word[i]) * length(word[j])` where the two words do not share common letters. You may assume that each word will contain only lower case letters. If no such two words exist, return 0.

Example 1:

```
Input: ["abcw","baz","foo","bar","xtfn","abcdef"]

Output: 16

Explanation: The two words can be "abcw", "xtfn".
```

Example 2:

```
Input: ["a","ab","abc","d","cd","bcd","abcd"]

Output: 4

Explanation: The two words can be "ab", "cd".
```

Example 3:

```
Input: ["a","aa","aaa","aaaa"]

Output: 0

Explanation: No such pair of words.
```

https://leetcode.com/problems/single-number-iii/submissions/

Given an integer array `nums`, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once. You can return the answer in **any order**.

Follow up: Your algorithm should run in linear runtime complexity. Could you implement it using only constant space complexity?

Example 1:

```
Input: nums = [1,2,1,3,2,5]

Output: [3,5]

Explanation:  [5, 3] is also a valid answer.
```

Example 2:

```
Input: nums = [-1,0]

Output: [-1,0]
```

Example 3:

```
Input: nums = [0,1]

Output: [1,0]
```

Constraints:

- `1 <= nums.length <= 30000`
- Each integer in `nums` will appear twice, only two integers will appear once.

https://leetcode.com/problems/counting-bits/

Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.

Example 1:

```
Input: 2

Output: [0,1,1]
```

Example 2:

```
Input: 5

Output: [0,1,1,2,1,2]
```

Follow up:

- It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
- Space complexity should be **O(n)**.
- Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.