**338. Counting Bits: -**

Easy Accepted: 865.1K Submissions: 1.1M Acceptance Rate: 76.7%

Given an integer n, return *an array*ans*of length*n + 1*such that for each*i(0 <= i <= n)*,*ans[i]*is the****number of***1***'s****in the binary representation of*i.

**Example 1:**

**Input:** n = 2

**Output:** [0,1,1]

**Explanation:**

0 --> 0

1 --> 1

2 --> 10

**Example 2:**

**Input:** n = 5

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

**Explanation:**

0 --> 0

1 --> 1

2 --> 10

3 --> 11

4 --> 100

5 --> 101

**Constraints:**

* 0 <= n <= 105

**Follow up:**

* It is very easy to come up with a solution with a runtime of O(n log n). Can you do it in linear time O(n) and possibly in a single pass?
* Can you do it without using any built-in function (i.e., like \_\_builtin\_popcount in C++)?

**Code: -**

class Solution {

public:

    vector<int> countBits(int n) {

        vector<int> ans(n+1);

        if(n==0)

            return ans;

        for(int i=0; i<=n; ++i){

            int num=i, count=0;

            while(num > 0){

                num = num & (num - 1);

                ++count;

            }

            ans[i] = count;

        }

        return ans;

    }

};

**T.C: - O(N \* no. of set bits per index)**

**S.C: - O(1)**