338 Counting Bits (link)

Description

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 <= 10^5`

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++)?

(scroll down for solution)

Solution

Language: cpp

Status: Accepted

```cpp
#include <vector>

class Solution {
public:
    std::vector<int> countBits(int n) {
        std::vector<int> ans(n + 1, 0);

        for (int i = 0; i <= n; ++i) {
            ans[i] = countOnes(i);
        }

        return ans;
    }

    int countOnes(int num) {
        int count = 0;
        while (num > 0) {
            count += num & 1;
            num >>= 1;
        }
        return count;
    }
};
```