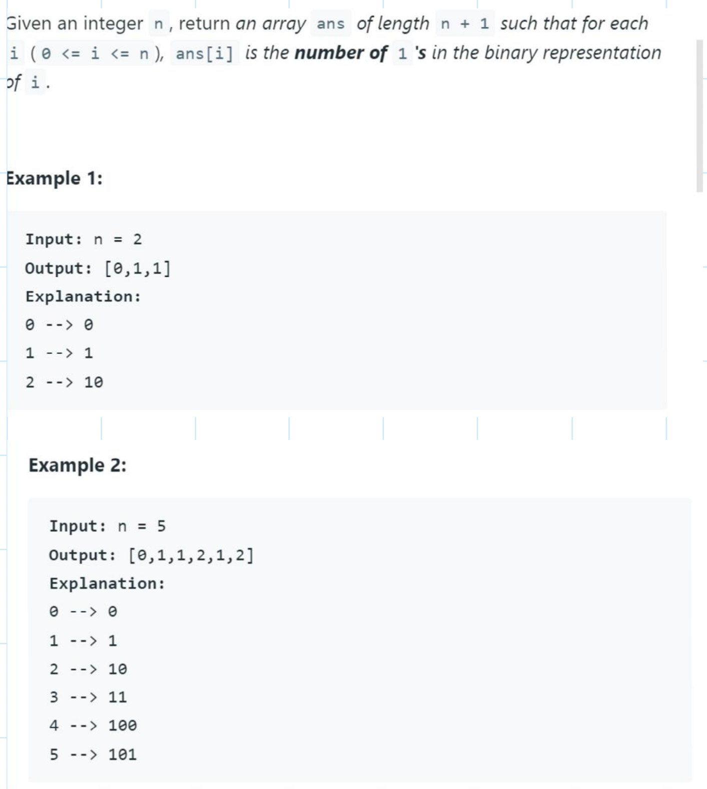
# 338. Counting Bits

01 March 2022 05:04 PM



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
```

$n = 5$

$0 \rightarrow 0 \qquad 3 \rightarrow 11$

$1 \rightarrow 1 \qquad 4 \rightarrow 100$

$2 \rightarrow 10 \qquad 5 \rightarrow 101$

$[0, 1, 1, 2, 1, 2]$

So, we have to count no of 1, how do we do that ?

To form 0 we want only 0

To form 1 we want only 1

By seeing that I can say if you have even nos. let say 4, So you want the answer of 4.

Whatever the ans of 4 is the ans of 2 as well. So, I can say that even = n/2

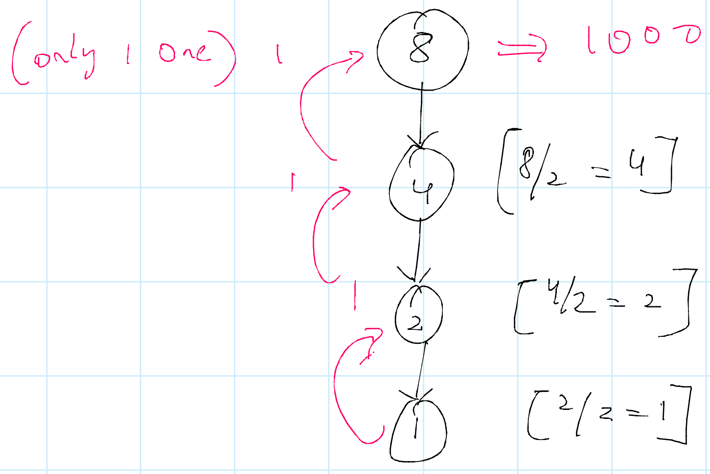
for e.g.

Even Case

| | | nos of 1 |
|---|---|---|
| 2 → | 0010 | 1 |
| 3 → | 0011 | |
| 4 → | 0100 | 1 |

So if you divide by 2 you get that

let take e.g. as 8 now



we know for 1 its '1'

Odd case

for odd = 1 + n/2

let say for 5, what ever the ans of 5 is the ans 3.

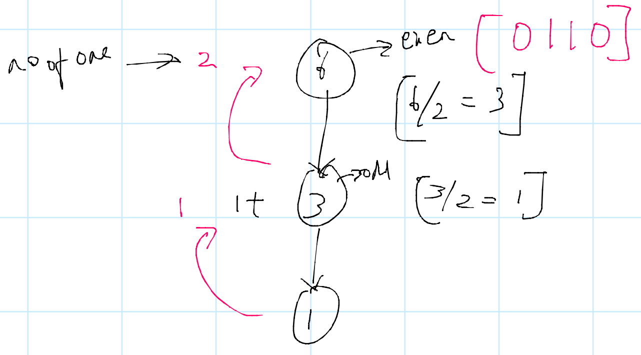
3 0011 --> 2

4 0100

5 0100 --> 2

no of one is 2.

Now you guys may ask for 6. let solve that



T.C ⇒ O(N log N)

S.C ⇒ O(N)

The logic for adding '+1' for the odd nos

(LSB)

→ The odd nos have their rightmost bit . the reason that no bit

The logic [illegible]

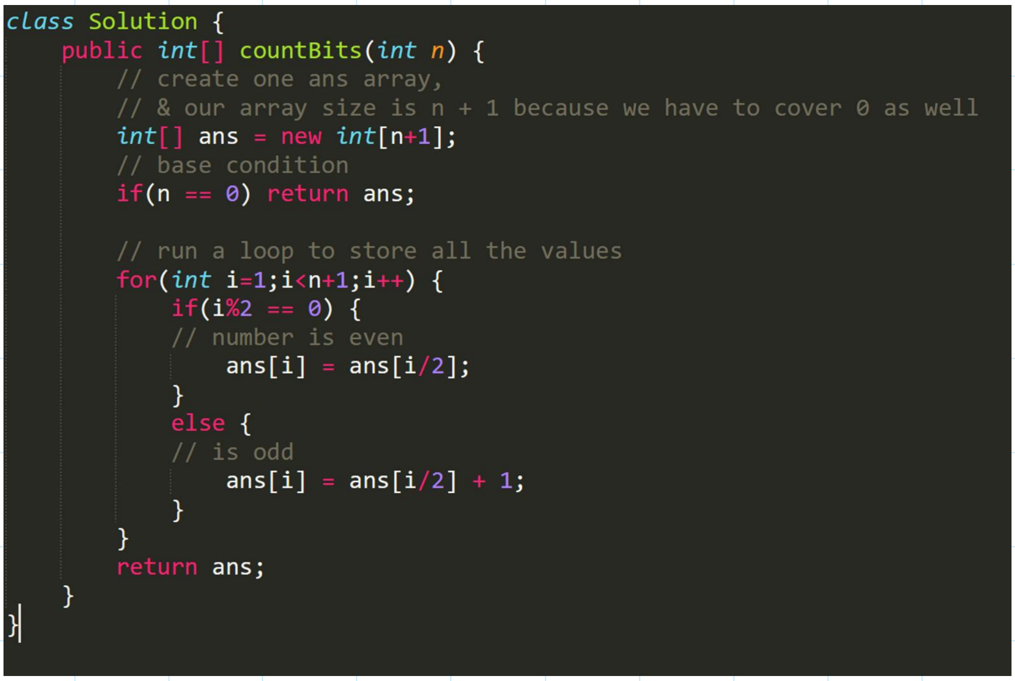
→ The odd nos have their rightmost bit (LSB), the reason that no bit other than '0th' bit will contribute an addition of odd nos, all the other power of '2' will add to even

Eg

2 ⇒ 0 0 1 0
3 ⇒ 0 0 1 1
4 ⇒ 0 1 0 0
5 ⇒ 0 1 0 1 → LSB is set to 1 for all odd.
6 ⇒ 0 1 1 0
7 ⇒ 0 1 1 1

→ dividing by 2 is equal to right shift by 1

in case of odd number, when you do right shift by 1, we will lose the right most bit. So inorder to compensate that lose bit we add +1.



```java
class Solution {
    public int[] countBits(int n) {
        // create one ans array,
        // & our array size is n + 1 because we have to cover 0 as well
        int[] ans = new int[n+1];
        // base condition
        if(n == 0) return ans;

        // run a loop to store all the values
        for(int i=1;i<n+1;i++) {
            if(i%2 == 0) {
            // number is even
                ans[i] = ans[i/2];
            }
            else {
            // is odd
                ans[i] = ans[i/2] + 1;
            }
        }
        return ans;
    }
}
```