**Title: Counting Bits**

**Scenario**

Imagine you are a software developer working on a low-level system that requires frequent manipulation and analysis of binary representations of integers. As part of your tasks, you need to frequently count the number of 1's in the binary representation of various integers. To optimize this task, you decide to write a function that, given an integer n, precomputes the number of 1's for all integers from 0 to n and stores these counts in an array for quick lookup.

**Problem Statement**

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.

**Input Format**

* An integer n.

**Output Format**

* An array of integers representing the number of 1's in the binary representation of each number from 0 to n.

**Constraints**

* 0 <= n <= 105

**Example 1**

**Input:**

2

**Output:**

[0, 1, 1]

**Explanation:**

* 0 --> 0
* 1 --> 1
* 2 --> 10

**Example 2**

**Input:**

5

**Output:**

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

**Explanation:**

* 0 --> 0
* 1 --> 1
* 2 --> 10
* 3 --> 11
* 4 --> 100
* 5 --> 101

**Additional Test Cases**

**Test Case 1**

**Input:**

0

**Output:**

[0]

**Explanation:**

* 0 --> 0

**Test Case 2**

**Input:**

1

**Output:**

[0, 1]

**Explanation:**

* 0 --> 0
* 1 --> 1

**Test Case 3**

**Input:**

10

**Output:**

[0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2]

**Explanation:**

* 0 --> 0
* 1 --> 1
* 2 --> 10
* 3 --> 11
* 4 --> 100
* 5 --> 101
* 6 --> 110
* 7 --> 111
* 8 --> 1000
* 9 --> 1001
* 10 --> 1010

**Test Case 4**

**Input:**

15

**Output:**

[0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4]

**Explanation:**

* 0 --> 0
* 1 --> 1
* 2 --> 10
* 3 --> 11
* 4 --> 100
* 5 --> 101
* 6 --> 110
* 7 --> 111
* 8 --> 1000
* 9 --> 1001
* 10 --> 1010
* 11 --> 1011
* 12 --> 1100
* 13 --> 1101
* 14 --> 1110
* 15 --> 1111

**Test Case 5**

**Input:**

50

**Output:**

[0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3]

**Test Case 6**

**Input:**

255

**Output:**

[0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8]

These test cases should help verify the correctness and efficiency of the solution across a range of input sizes and edge cases