Documentation for Gray Code Solution

Problem Description

An n-bit Gray code sequence is a sequence of (2^n) integers that satisfy the following properties:

- Every integer in the sequence is within the inclusive range ([0, 2^n - 1]).
- The first integer in the sequence is 0.
- Each integer appears no more than once in the sequence.
- The binary representation of every pair of adjacent integers differs by exactly one bit.
- The binary representation of the first and last integers in the sequence differs by exactly one bit.

Given an integer (n), the task is to return any valid n-bit Gray code sequence.

Example 1

Input [n = 2]

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

Explanation

- The binary representation of ([0, 1, 3, 2]) is ([00, 01, 11, 10]).
- 00 and 01 differ by one bit.
- 01 and 11 differ by one bit.
- 11 and 10 differ by one bit.
- 10 and 00 differ by one bit.
- ([0, 2, 3, 1]) is also a valid Gray code sequence, with binary representation ([00, 10, 11, 01]).

Example 2

Input [n = 1]

Output [[0, 1]]

Constraints

- (1 leq n leq 16)

Approach

The solution leverages the mathematical properties of Gray codes. An n-bit Gray code can be generated using the formula:

[text{Gray}(i) = i oplus (i >> 1)]

where (i) ranges from 0 to (2^n - 1).

Detailed Steps

1. Initialization: Create an empty list `result` to store the Gray code sequence.
2. Iterate over range: For each integer (i) from 0 to (2^n - 1), calculate its Gray code using the formula (i oplus (i >> 1)).
3. Append to result: Append the calculated Gray code to the `result` list.
4. Return result: After the loop, return the `result` list containing the Gray code sequence.

Example Usage

1. **For (n = 2):**

- *Input:* (2)
- *Output:* ([0, 1, 3, 2])

2. **For (n = 1):**

- *Input:* (1)
- *Output:* ([0, 1])

Conclusion

The provided solution efficiently generates an n-bit Gray code sequence using bitwise operations. This approach ensures that each pair of consecutive integers in the sequence differs by exactly one bit, and all constraints are satisfied.