

Power of two

To determine if an integer (n) is a power of two, you can use a simple mathematical property: An integer (n) is a power of two if it is greater than zero and the bitwise AND of (n) and ($n-1$) is zero. Here's how that works in different steps:

Steps:

1. Check if (n) is greater than 0.
2. Compute (n) AND ($n-1$).
3. If the result is 0, then (n) is a power of two; otherwise, it is not.

Python Code Implementation:

Here's a simple implementation of the above logic in Python:

```
def is_power_of_two(n):  
    if n <= 0:  
        return False  
    return (n & (n - 1)) == 0
```

Example Usage:

Test the function with the provided examples:

```
print(is_power_of_two(1)) # Output: True  
print(is_power_of_two(16)) # Output: True  
print(is_power_of_two(3)) # Output: False
```

Explanation:

- Example 1: For ($n = 1$), ($1 \& (1-1) = 1 \& 0 = 0$) $\rightarrow$ true.
- Example 2: For ($n = 16$), ($16 \& (16-1) = 16 \& 15 = 0$) $\rightarrow$ true.
- Example 3: For ($n = 3$), ($3 \& (3-1) = 3 \& 2 = 2$) $\rightarrow$ false.

Constraints:

This approach adheres to the constraints specified: ($-2^{31} \leq n \leq 2^{31} - 1$). The check for ($n > 0$) ensures that negative numbers and zero are properly rejected for being powers of two.