

■ ■ Power of Two

Bit Manipulation • Easy

■ The Hook

Powers of 2 are everywhere in computing: memory sizes, hash table capacities, FFT algorithms, network subnets. Checking if a number is a power of 2 with $n \% 2 == 0$ loops won't scale — but there is a one-liner that exploits a beautiful binary pattern. Powers of 2 in binary always have exactly one bit set.

■ Core Intuition

1 = 0001, 2 = 0010, 4 = 0100, 8 = 1000. Notice: exactly one bit is 1. And $n-1$ for a power of 2 looks like: 3 = 0011, 7 = 0111, 15 = 1111. It's the exact inverse bit pattern below the single set bit. So $n \& (n-1)$ is always 0 for a power of 2. This check fails for $n=0$ (no bits set), so we add $n > 0$.

■ 8 = 1000, 7 = 0111, $8 \& 7 = 0000 \rightarrow$ power of 2 ✓

■ 6 = 0110, 5 = 0101, $6 \& 5 = 0100 \neq 0 \rightarrow$ not a power of 2 ✓

■ Dry Run

$n = 16$:

■ 16 = 10000, 15 = 01111

■ $16 \& 15 = 00000 = 0$, and $n > 0 \rightarrow$ TRUE ✓

$n = 12$:

■ 12 = 1100, 11 = 1011

■ $12 \& 11 = 1000 \neq 0 \rightarrow$ FALSE ✓

■ Java Implementation

```
public class PowerOfTwo {  
    // Clean O(1) solution  
    public static boolean isPowerOfTwo(int n) {  
        return n > 0 && (n & (n - 1)) == 0;  
    }  
    // Alternative: count set bits – should be exactly 1
```

```
public static boolean isPowerOfTwoAlt(int n) {
    return n > 0 && Integer.bitCount(n) == 1;
}
}
```

■ Complexity

Case	Time	Space	Why
All cases	O(1)	O(1)	Two operations: compare and AND

■ Common Mistakes

- Forgetting $n > 0$: $n=0$ gives $0 \& -1 = 0$, which would incorrectly return true.
- Negative numbers: In Java, $-1 \& -2 = -2 \neq 0$, so negatives are handled, but the $n > 0$ guard is still cleaner.

■ Practice Problems

Problem	Difficulty
Power of Two (LeetCode 231)	Easy
Power of Three (LeetCode 326)	Easy
Power of Four (LeetCode 342)	Easy
Find the Duplicate Number	Medium

■ Quick Revision

- Power of 2 has exactly one set bit
- Check: $n > 0 \&\& (n \& (n-1)) == 0$
- $n \& (n-1)$ clears the lowest set bit — if result is 0, only one bit was set

■ Mental Model

A power of 2 is a lone wolf in binary — exactly one bit standing alone. Subtracting 1 turns that wolf into a pack of smaller bits below it. AND-ing them together produces zero. No lone wolf survives.