191. Number of 1 Bits

Easy | Topics | Companies

Given a positive integer `n`, write a function that returns the number of set bits in its binary representation (also known as the Hamming weight).

Example 1:

> **Input:** `n = 11`
>
> **Output:** `3`
>
> **Explanation:**
>
> The input binary string **1011** has a total of three set bits.

Example 2:

> **Input:** `n = 128`
>
> **Output:** `1`
>
> **Explanation:**
>
> The input binary string **10000000** has a total of one set bit.

Example 3:

> **Input:** n = 2147483645
>
> **Output:** 30
>
> **Explanation:**
>
> The input binary string **11111111111111111111111111111101** has a total of thirty set bits.

Constraints:

- $1 <= n <= 2^{31} - 1$

Follow up: If this function is called many times, how would you optimize it?

Python:

```
class Solution:
    def hammingWeight(self, n: int) -> int:
        # Approach 1: Built-in (simple, clean)
        # return bin(n).count("1")

        # Approach 2: Bit manipulation (efficient)
        count = 0
        while n:
            n &= (n - 1)  # Remove the lowest set bit
            count += 1
        return count
```

JavaScript:

```
/**
 * @param {number} n
 * @return {number}
 */
var hammingWeight = function(n) {
    let count = 0;
    while (n !== 0) {
        count += n & 1;   // add 1 if the last bit is set
        n >>>= 1;         // unsigned right shift to check next bit
```

```
    }
    return count;
};
```

Java:

```
class Solution {
    public int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            n = n & (n - 1);  // remove the lowest set bit
            count++;
        }
        return count;
    }
}
```