

■■ Count Set Bits

Bit Manipulation • Easy (Popcount / Hamming Weight)

■ The Hook

How many 1s are in the binary representation of a number? This is called the Hamming weight or popcount. It matters in cryptography (Hamming distance), error correction codes, population count in bitmask DP, and even hardware has dedicated instructions for it (POPCNT). Three algorithms exist: naive, Brian Kernighan's trick, and built-in. Understanding all three makes you dangerous in interviews.

■ Core Intuition — Brian Kernighan's Trick

Observation: $n \& (n-1)$ clears the lowest set bit of n . Why? Subtracting 1 from n flips the lowest set bit to 0 and all lower bits to 1. ANDing with n cancels those flipped bits. So every iteration of the loop removes exactly one '1'. Count iterations = count set bits.

■ $n = 12 = 1100$, $n-1 = 11 = 1011$, $n \& (n-1) = 1000$ (removed one '1')

■ $n = 8 = 1000$, $n-1 = 7 = 0111$, $n \& (n-1) = 0000$ (removed last '1')

■ Dry Run — $n = 13$ (1101)

Step	n (binary)	n-1	n & (n-1)	count
1	1101 (13)	1100 (12)	1100 (12)	1
2	1100 (12)	1011 (11)	1000 (8)	2
3	1000 (8)	0111 (7)	0000 (0)	3
4	0000 (0) — STOP	—	—	3 ✓

■ Java Implementation — Three Approaches

```
public class CountSetBits {  
  
    // Approach 1: Naive — check every bit  
    public static int countNaive(int n) {  
        int count = 0;  
        while (n != 0) {  
            count += (n & 1); // check last bit  
            n = n >> 1;  
        }  
    }  
}
```

```

n >>= 1; // shift right
}
return count;
}

// Approach 2: Brian Kernighan — O(set bits)
public static int countKernighan(int n) {
    int count = 0;
    while (n != 0) {
        n = n & (n - 1); // clears lowest set bit
        count++;
    }
    return count;
}

// Approach 3: Built-in Java (fastest in practice)
public static int countBuiltin(int n) {
    return Integer.bitCount(n);
}
}

```

■ Complexity

Case	Time	Space	Why
Naive	$O(\log n)$	$O(1)$	Iterates through all bits (32 for int)
Kernighan	$O(k)$	$O(1)$	k = number of set bits — always $\leq \log n$
Built-in	$O(1)$	$O(1)$	Single POPCNT instruction on modern CPUs

■ Common Mistakes

- Using `n >> 1` on a signed int shifts in the sign bit. Use `n >>> 1` (unsigned right shift) in Java to be safe.
- `n & (n-1)` on `n = 0` is safe (returns 0), but the loop must check `n != 0`.

■ Practice Problems

Problem	Difficulty
Number of 1 Bits (LeetCode 191)	Easy

Counting Bits (LeetCode 338)	Easy
Hamming Distance (LeetCode 461)	Easy
Total Hamming Distance (LeetCode 477)	Medium

■ Quick Revision

- $n \& (n-1)$ removes the lowest set bit
- Loop count = number of set bits (Brian Kernighan)
- Java built-in: `Integer.bitCount(n)`
- Used in Hamming distance, subset enumeration, bitmask DP

■ Mental Model

Every time you do $n \& (n-1)$, you're erasing the 'last candle' in a candelabra. Count how many times you blow out candles before the room goes dark. That count is your answer.