Day 25 Bitwise

ITSRUNTYM

What Are Bitwise Operations?

Bitwise operations directly operate on **binary representations** of integers. They're very fast and powerful, especially in low-level tasks (like bit masks, compression, graphics, cryptography, etc).

List of Bitwise Operators

Operator	Symbol	Description	Example
AND	&	1 if both bits are 1	5 & 3 = 1
OR	`	`	1 if at least one bit is 1
XOR	^	1 if bits are different	5 ^ 3 = 6
NOT	~	Inverts all bits	~5 = -6
Left Shift	<<	Shifts bits left (multiply by 2^n)	5 << 1 = 10
Right Shift	>>	Shifts bits right (divide by 2^n)	5 >> 1 = 2

Binary Example (Let's use 5 and 3)

5 in binary = 0101
3 in binary = 0011
5 & 3 = 0001 = 1
5 | 3 = 0111 = 7
5 ^ 3 = 0110 = 6
5 << 1 = 1010 = 10
5 >> 1 = 0010 = 2

Operator Usage with Examples

1. AND (&)

- Use: Clear a bit / mask check

```
int a = 5; // 0101
int b = 3; // 0011
System.out.println(a & b); // Output: 1
```

2. OR (|)
- Use: Set a bit

```
int a = 5; // 0101
int b = 3; // 0011
System.out.println(a | b); // Output: 7
```

3. XOR (^)
- Use: Toggle bits / swap without temp variable

```
int a = 5; // 0101
int b = 3; // 0011
System.out.println(a ^ b); // Output: 6
```

4. Left Shift (<<)
- Use: Multiply by 2^n

```
int a = 5; // 0101
System.out.println(a << 1); // Output: 10
```

5. Right Shift (>>)
- Use: Divide by 2^n

```
int a = 5; // 0101
System.out.println(a >> 1); // Output: 2
```

Use-Cases in Real Problems

Use Case	Bitwise Trick
Check even/odd	n & 1 == 0 (even)
Multiply by 2	n << 1
Divide by 2	n >> 1
Swap numbers	a ^= b; b ^= a; a ^= b;
Check bit is set	(n & (1 << i)) != 0
Count set bits	n & (n - 1) (Brian Kernighan’s algo)

Example Problems

1. Check if a number is power of 2

```
boolean isPowerOfTwo(int n) {
    return n > 0 && (n & (n - 1)) == 0;
}
```

2. Count number of set bits

```
int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
        n = n & (n - 1); // Removes last set bit
        count++;
    }
    return count;
}
```

3. Find the only non-repeating element

```
int singleNumber(int[] nums) {
    int res = 0;
    for (int num : nums) {
        res ^= num;
    }
    return res;
}
```

4. Get the ith bit

```
int getIthBit(int n, int i) {
    return (n & (1 << i)) != 0 ? 1 : 0;
}
```

5. Set the ith bit

```
int setIthBit(int n, int i) {
    return n | (1 << i);
}
```