Bit Manipulation in Java

Bit manipulation uses bitwise operators to solve problems efficiently. Let's explore finding a single number and checking if a number is a power of 2.

1. Single Number: XOR Magic

Find the only non-repeating number in an array where all others appear twice using XOR.

Java Implementation

```java
public class SingleNumber {
    public static int singleNumber(int[] nums) {
        int result = 0;
        for (int num : nums) result ^= num;
        return result;
    }

    public static void main(String[] args) {
        int[] nums = {2, 2, 1, 1, 4};
        System.out.println("Single number: " + singleNumber(nums));
    }
}
```

Output: `Single number: 4`

2. Power of 2: Bit Check

A number is a power of 2 if it has exactly one 1-bit in its binary representation.

Java Implementation

```java
public class PowerOfTwo {
    public static boolean isPowerOfTwo(int n) {
        return n > 0 && (n & (n - 1)) == 0;
    }

    public static void main(String[] args) {
        int n = 16;
        System.out.println(n + " is power of 2: " + isPowerOfTwo(n));
```

```
        n = 18;
        System.out.println(n + " is power of 2: " + isPowerOfTwo(n));
    }
}
```

Output:

```
16 is power of 2: true
18 is power of 2: false
```

Conclusion Across Blogs

- **Sorting:** Quicksort, merge sort, and bubble sort cover speed, stability, and simplicity.
- **Strings:** KMP, tries, and Rabin-Karp handle pattern matching and prefix searches.
- **Searching:** Binary and linear search offer logarithmic and linear solutions.
- **Graphs:** DFS, BFS, Dijkstra's, and Kruskal's tackle traversal, paths, and MSTs.
- **Bit Manipulation:** XOR and bit checks solve tricky problems with minimal operations.

Each category has its strengths—pick the right tool for your problem and experiment with these implementations! Let me know if you'd like more depth or additional algorithms in any category.