

# Manipulación de Bits en Java

La manipulación de bits utiliza operadores bit a bit para resolver problemas de manera eficiente. exploremos cómo encontrar un solo número y verificar si un número es una potencia de 2.

## 1. Número Único: Magia XOR

Encuentra el único número que no se repite en un arreglo donde todos los demás aparecen dos veces utilizando XOR.

### Implementación en 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("Número único: " + singleNumber(nums));  
    }  
}
```

**Salida:** Número único: 4

## 2. Potencia de 2: Verificación de Bit

Un número es una potencia de 2 si tiene exactamente un 1-bit en su representación binaria.

### Implementación en 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 + " es potencia de 2: " + isPowerOfTwo(n));
        n = 18;
        System.out.println(n + " es potencia de 2: " + isPowerOfTwo(n));
    }
}

```

### Salida:

```

16 es potencia de 2: true
18 es potencia de 2: false

```

---

## Conclusión a lo largo de los Blogs

- **Ordenamiento:** Quicksort, merge sort y bubble sort cubren velocidad, estabilidad y simplicidad.
- **Cadenas:** KMP, tries y Rabin-Karp manejan la coincidencia de patrones y las búsquedas de prefijos.
- **Búsqueda:** Las búsquedas binaria y lineal ofrecen soluciones logarítmicas y lineales.
- **Grafos:** DFS, BFS, Dijkstra's y Kruskal's abordan el recorrido, los caminos y los MSTs.
- **Manipulación de Bits:** XOR y verificaciones de bits resuelven problemas difíciles con operaciones mínimas.

Cada categoría tiene sus fortalezas—elige la herramienta adecuada para tu problema y experimenta con estas implementaciones. ¡Hazme saber si te gustaría más profundidad o algoritmos adicionales en alguna categoría.