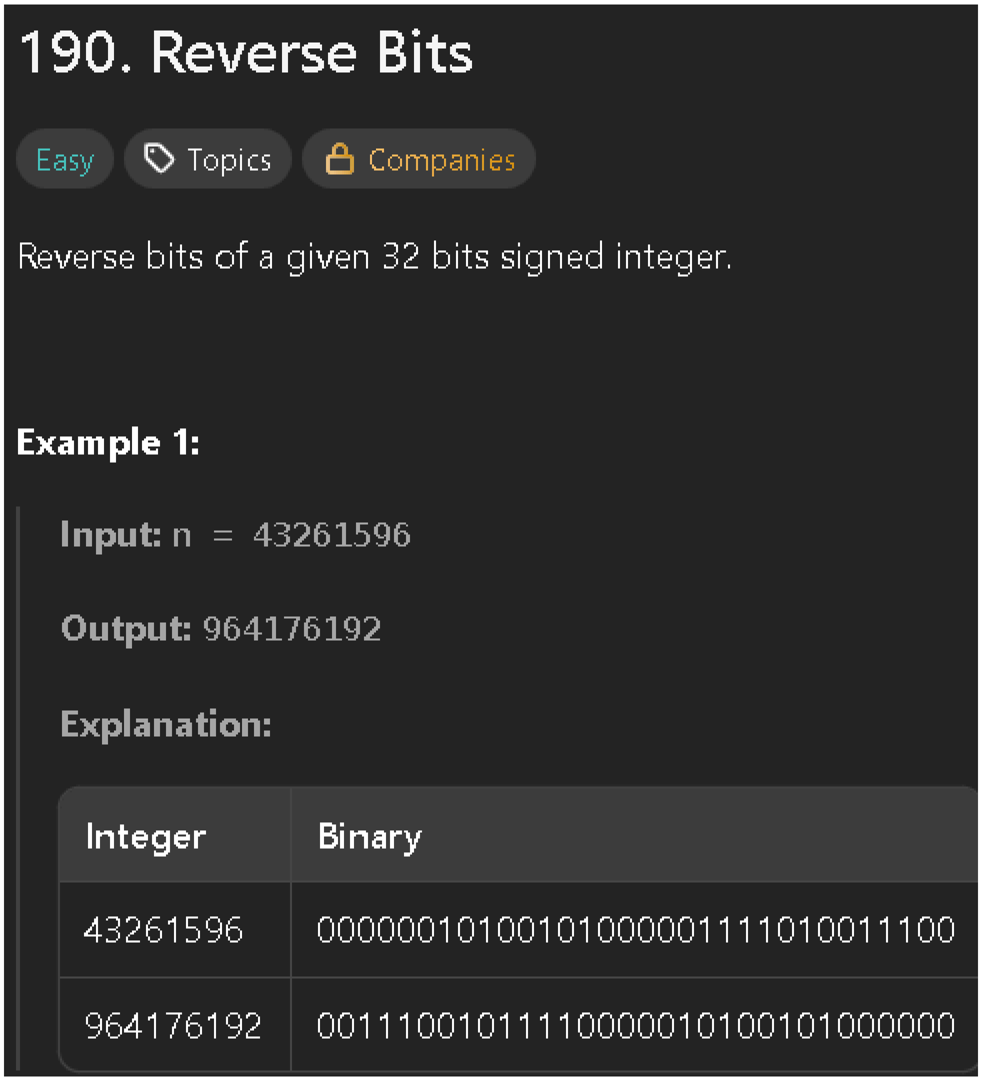


190. Reverse Bits

Easy Topics Companies

Reverse bits of a given 32 bits signed integer.

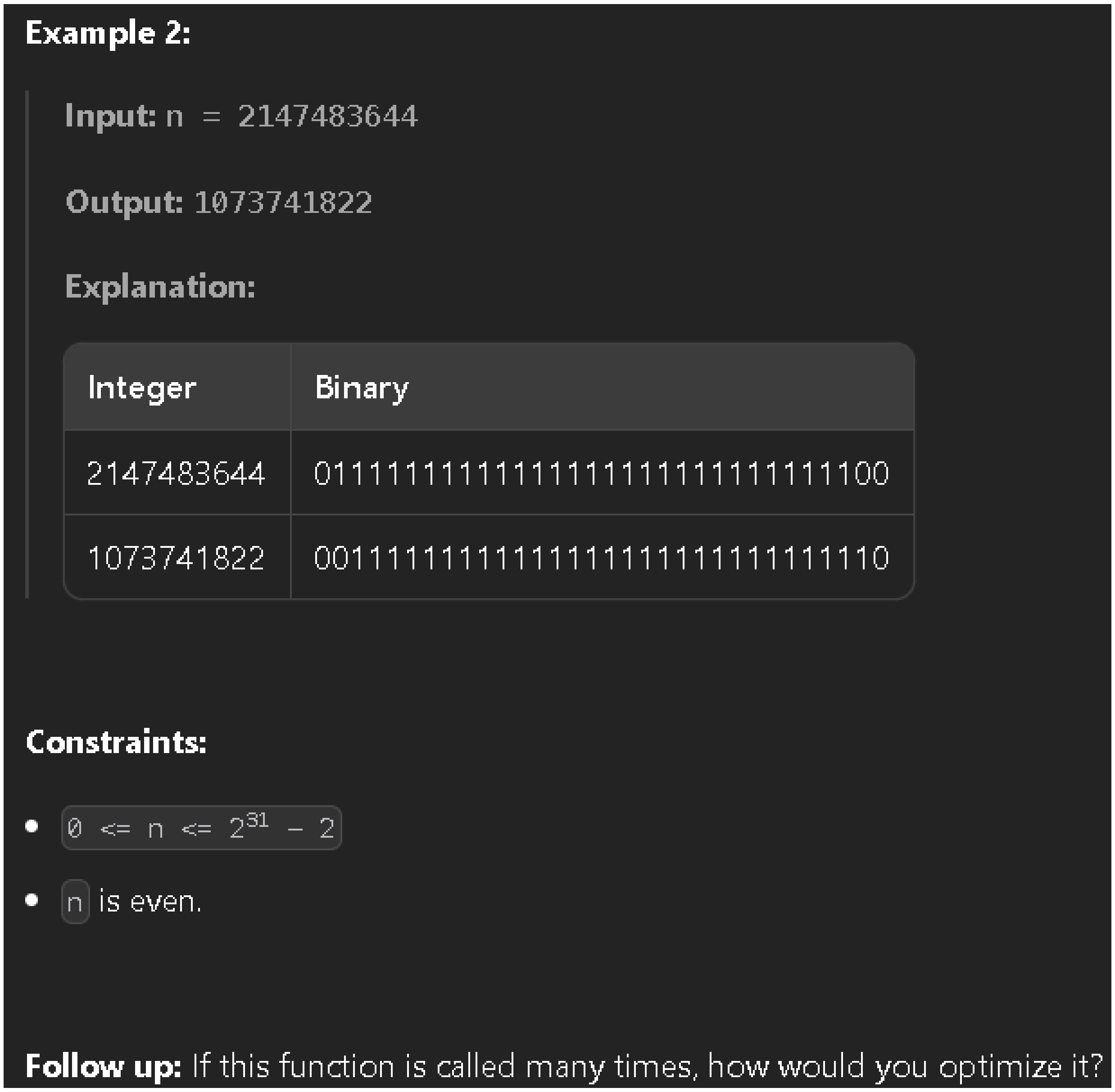
Example 1:

Input: n = 43261596

Output: 964176192

Explanation:

Integer	Binary
43261596	00000010100101000001111010011100
964176192	00111001011110000010100101000000



Example 2:

> **Input:** n = 2147483644
>
> **Output:** 1073741822
>
> **Explanation:**
>
> | Integer | Binary |
> |---|---|
> | 2147483644 | 01111111111111111111111111111100 |
> | 1073741822 | 00111111111111111111111111111110 |

Constraints:

- `0 <= n <= 2^31 - 2`
- `n` is even.

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

Python:

```
class Solution:
    def reverseBits(self, n: int) -> int:
        result = 0
        for _ in range(32):
            # Shift result left and add the last bit of n
            result = (result << 1) | (n & 1)
            # Shift n right to process the next bit
            n >>= 1
        return result
```

JavaScript:

```javascript
/**
 * @param {number} n
 * @return {number}
 */
var reverseBits = function(n) {
    let result = 0;

    for (let i = 0; i < 32; i++) {
        result <<= 1;          // Shift result left to make space
        result |= (n & 1);     // Add the last bit of n
        n >>= 1;               // Drop the last bit of n
    }

    // >>> 0 ensures unsigned 32-bit integer
    return result >>> 0;
};
```

Java:

```java
class Solution {
    public int reverseBits(int n) {
        int result = 0;

        for (int i = 0; i < 32; i++) {
            // Shift result left to make space
            result <<= 1;

            // Add the last bit of n
            result |= (n & 1);

            // Shift n right (unsigned shift to avoid sign extension)
            n >>>= 1;
        }

        return result;
    }
}
```