0190. 颠倒二进制位

ITCharge 大约 1 分钟

- 标签：位运算、分治
- 难度：简单

题目链接

- 0190. 颠倒二进制位 - 力扣

题目大意

描述：给定一个 32 位无符号整数 n。

要求：将 n 所有二进位进行翻转，并返回翻转后的整数。

说明：

- 输入是一个长度为 32 的二进制字符串。

示例：

- 示例 1：

```py
输入：n = 00000010100101000001111010011100
输出：964176192 (00111001011110000010100101000000)
解释：输入的二进制串 00000010100101000001111010011100 表示无符号整数 43261596，
     因此返回 964176192，其二进制表示形式为 00111001011110000010100101000000。
```

- 示例 2：

```py
输入：n = 11111111111111111111111111111101
输出：3221225471 (10111111111111111111111111111111)
解释：输入的二进制串 11111111111111111111111111111101 表示无符号整数 4294967293，
     因此返回 3221225471 其二进制表示形式为 10111111111111111111111111111111。
```

解题思路

思路 1：逐位翻转

1. 用一个变量 res 存储翻转后的结果。
2. 将 n 不断进行右移（即 `n >> 1`），从低位到高位进行枚举，此时 n 的最低位就是我们枚举的二进位。
3. 同时 res 不断左移（即 `res << 1`），并将当前枚举的二进位翻转后的结果（即 `n & 1`）拼接到 res 的末尾（即 `(res << 1) | (n & 1)`）。

思路 1：代码

```py
class Solution:
    def reverseBits(self, n: int) -> int:
        res = 0
        for i in range(32):
            res = (res << 1) | (    1)
            n >>= 1
        return res
```

思路 1：复杂度分析

- **时间复杂度**：$O(\log n)$。
- **空间复杂度**：$O(1)$。