0050. Pow(x, n)

ITCharge 大约 1 分钟

- 标签：递归、数学
- 难度：中等

题目链接

- 0050. Pow(x, n) - 力扣

题目大意

描述：给定浮点数 x 和整数 n。

要求：计算 x 的 n 次方（即 x^n）。

说明：

- $-100.0 < x < 100.0$。
- $-2^{31} \leq n \leq 2^{31} - 1$。
- n 是一个整数。
- $-10^4 \leq x^n \leq 10^4$。

示例：

- 示例 1：

```py
输入：x = 2.00000, n = 10
输出：1024.00000
```

- 示例 2：

```py
输入：x = 2.00000, n = -2
输出：0.25000
解释：2-2 = 1/22 = 1/4 = 0.25
```

解题思路

思路 1：分治算法

常规方法是直接将 x 累乘 n 次得出结果，时间复杂度为 $O(n)$。

我们可以利用分治算法来减少时间复杂度。

根据 n 的奇偶性，我们可以得到以下结论：

1. 如果 n 为偶数，$x^n = x^{n/2} \times x^{n/2}$。
2. 如果 n 为奇数，$x^n = x \times x^{(n-1)/2} \times x^{(n-1)/2}$。

$x^{(n/2)}$ 或 $x^{(n-1)/2}$ 又可以继续向下递归划分。

则我们可以利用低纬度的幂计算结果，来得到高纬度的幂计算结果。

这样递归求解，时间复杂度为 $O(\log n)$，并且递归也可以转为递推来做。

需要注意如果 n 为负数，可以转换为 $'\ n)$。

思路 1：代码

```py
class Solution:
    def myPow(self, x: float, n: int) -> float:
        if x == 0.0:
            return 0.0
        res = 1
        if n < 0:
            x = 1/x
            n = -n
        while n:
            if n & 1:
                res *= x
            x *= x
            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**：$O(1)$。