Pow(x, n)

For this problem, the key is to compute x^n efficiently, especially for large n, including negative powers.

Key Idea: Fast Exponentiation (Exponentiation by Squaring):

Why?

- Naive method: Multiply x n times $\rightarrow O(n)$ (too slow for large n).
- Fast exponentiation: Repeatedly square x, halve n $\rightarrow O(\log n)$.

Handling Negative n

- $x^{-n} = 1/x^n$
- Convert n to positive, compute power, then take reciprocal.

Algorithm:

1. If $n < 0$:
   - Set $x = 1/x$
   - Use $n = -n$ (careful with INT_MIN, use long long to avoid overflow)
2. Initialize result = 1
3. While $n > 0$:
   - If n is odd, multiply result by x
   - Square x and halve n.

Complexity:

- Time: $O(\log n)$
- Space: $O(1)$