# 16. 数值的整数次方

## 题目链接

[牛客网](https://www.nowcoder.com/practice/1a834e5e3e1a4b7ba251417554e07c00?tpId=13&tqId=11165&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking&from=cyc_github)

## 题目描述

给定一个 double 类型的浮点数 x和 int 类型的整数 n，求 x 的 n 次方。

## 解题思路

最直观的解法是将 x 重复乘 n 次，x\*x\*x...\*x，那么时间复杂度为 O(N)。因为乘法是可交换的，所以可以将上述操作拆开成两半 (x\*x..\*x)\* (x\*x..\*x)，两半的计算是一样的，因此只需要计算一次。而且对于新拆开的计算，又可以继续拆开。这就是分治思想，将原问题的规模拆成多个规模较小的子问题，最后子问题的解合并起来。

本题中子问题是 xn/2，在将子问题合并时将子问题的解乘于自身相乘即可。但如果 n 不为偶数，那么拆成两半还会剩下一个 x，在将子问题合并时还需要需要多乘于一个 x。

因为 (x\*x)n/2 可以通过递归求解，并且每次递归 n 都减小一半，因此整个算法的时间复杂度为 O(logN)。

public double Power(double x, int n) {
 boolean isNegative = false;
 if (n < 0) {
 n = -n;
 isNegative = true;
 }
 double res = pow(x, n);
 return isNegative ? 1 / res : res;
}

private double pow(double x, int n) {
 if (n == 0) return 1;
 if (n == 1) return x;
 double res = pow(x, n / 2);
 res = res \* res;
 if (n % 2 != 0) res \*= x;
 return res;
}