第26题——数值的整数次方

实现 pow(x, n)，即计算 x 的 n 次幂函数（即，xn）。不得使用库函数，同时不需要考虑大数问题。

示例 1:

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

示例 2:

```
输入：x = 2.10000, n = 3
输出：9.26100
```

示例 3:

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

提示:

- $-100.0 < x < 100.0$
- $-2^{31} <= n <= 2^{31}-1$
- $-10^4 <= x^n <= 10^4$

快速幂（1）

比如要求x^{11}，正常的乘积需要循环乘11次，时间复杂度为 $O(n)$

快速幂的思想就是将指数11 可以转成二进制数1011，则原来的式子可以转化成

$x^{11} = x^{2^3+2^1+2^0} = x^{2^3} \times x^{2^1} \times x^{2^0}$，此时只运算了3次乘积，时间复杂度降至 $O(logn)$

下方代码中的 `x *= x` 是一个累乘的过程，得到四位二进制数，对应的四个权重，x，$x* = x$，$x^2 * = x^2$，$x^4 * = x^4$

1011二进制数，从右至左分别为1 1 0 1 ，**只有在1的位置上，才有相应的权重**，这也就是为什么需要通过与运算： `(b & 1) == 1` 判断最后一位是否为1。

$x - -- > x^{2^0}$ ---------> 1

$x^2 - -- > x^{2^1}$ -------------> 1

$x^4 - -- > x^{2^2}$ -----------------> 0

$x^8 - -- > x^{2^3}$ -----------------------> 1

最终的结果就是将每一位的1 所对应的权重相乘即可：$x^{2^0} \times x^{2^1} \times x^{2^3}$

代码如下：

```
double myPow(double x, int n){
    if(x==0){
        return 0;
    }
    double res=1.0;
    long b=n;//幂次必须转换为长整型数据
    if(b<0){
        x=1/x;
        b=-b;
    }
    while(b>0){
        if((b&1)==1)
        {
            res =res*x;
        }
        x =x*x;
        b>>=1;
    }
    return res;
}
```