Pow(x, n)

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

Example 1:

```
Input: x = 2.00000, n = 10

Output: 1024.00000
```

Example 2:

```
Input: x = 2.10000, n = 3

Output: 9.26100
```

Example 3:

```
Input: x = 2.00000, n = -2

Output: 0.25000

Explanation: 2^-2 = 1/2^2 = 1/4 = 0.25
```

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

Navie solution:

```
public double myPow(double x, int n) {
      double res=1;
     long t=n;
      t=Math.abs(t);
      for(int i=1;i<=t;i++)
      {
         res=res*x;
      }
     return n>0?res:(1/res);
}
```

Time Complexity: O(n)
Space Complexity: O(1)

Efficient Approach:

Binary exponentiation

```
public double myPow(double x, int n) {
        double res=1;
       long t=n;
        t=Math.abs(t);
        while(t>0)
        {
           if((t&1)==1)
           {
              res=res*x;
           }
           x=x*x;
           t=t>>1;
        }
             return n>0?res:(1/res);


    }
```

Time Complexity: O(log|n|)
Auxiliary Space: O(1)