数值的整数次方

题目描述

给定一个double类型的浮点数base和int类型的整数exponent。求base的exponent次方。

1.首先要处理特殊情况:

---exponent>0时,下一步计算,可以让新定义的指数变量直接等于exponent

---exponent<0时,底数不能为0,因为计算一个数的负数次幂要先算相应的正数次幂后再取倒数,0不能做 分母

---exponent==0,返回结果直接为1,任何数的0次幂都为1.

下面的循环可以通过两个方向解读:

a.奇偶数

3exp20 = 9exp10 = 81exp5 = 81*6054*2 = 81*36650916

如果指数为偶数,计算结果相当于当前底数翻倍,指数除2,如果指数为偶数,可以先把当前底数单独拆出一个来提前乘到结果上.

b.二进制

假设我们要求a^b，那么其实b是可以拆成二进制的，该二进制数第i位的权为2^(i-1)，例如当b==11时a11=a(2^0+2^1+2^3)

11的二进制是1011，$11 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1$，因此，我们将$a^{11}$转化为算 a2^0*a2^1*a2^3，也就是a1*a2*a8 ，看出来快的多了吧原来算11次，现在算三次

由于是二进制，很自然地想到用位运算这个强大的工具：&和>>&运算通常用于二进制取位操作

```
public class Solution {
public double Power(double base, int exponent) {
    double  res=1;
    double curr = base;
    int n =0;
       if(exponent>0){
           n = exponent;
       }else if(exponent<0){
           if(base==0){
               throw new RuntimeException("分母不能为零");
           }else{
               n = -exponent;
           }
       }else{
           return 1;
       }
    while(n!=0){
           //如果是奇数,把当前的底数curr单独拆出一个来先乘到结果上
           if((n&1)==1){
               res *= curr;
           }
           curr *= curr;//底数翻倍
           n >>= 1;//因为底数翻倍了,所以指数要除以二,即使此处n是奇数,这样除过之后也是我们想要的,比如5/2=2
       }
    return exponent>0?res:1/res;
```

```
    }
}
```