题目 数值的整数次方

考点 代码的完整性 热点指数 79138 通过率 31.19%

具体题目

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。

/*剑指书中细节： *1.当底数为0且指数<0时 *会出现对0求倒数的情况，需进行错误处理，设置一个全局变量； *2.判断底数是否等于0 *由于base为double型，不能直接用==判断 *3.优化求幂函数 *当n为偶数，a^n =（a^n/2）*（a^n/2）*当n为奇数，a^n = a^[(n-1)/2] * a^[(n-1)/2] * a *时间复杂度O(logn) */

```
public class Solution {
      boolean invalidInput=false;
      public double Power(double base, int exponent) {
        if(equal(base,0.0)&&exponent<0){
            invalidInput=true;
            return 0.0;
        }
        int absexponent=exponent;
        if(exponent<0)
            absexponent=-exponent;
        double res=getPower(base,absexponent);
        if(exponent<0)
            res=1.0/res;
        return res;
  }
    boolean equal(double num1,double num2){
        if(num1-num2>-0.000001&&num1-num2<0.000001)
            return true;
        else
            return false;
    }
    double getPower(double b,int e){
        if(e==0)
            return 1.0;
        if(e==1)
            return b;
        double result=getPower(b,e>>1);
        result*=result;
        if((e&1)==1)
            result*=b;
        return result;
    }
}
```

第一种方法：使用递归，时间复杂度O(logn) 当n为偶数，a^n =（a^n/2）*（a^n/2） 当n为奇数，a^n = a^[(n-1)/2] * a^[(n-1)/2] * a 举例： 2^11 = 2^1 * 2^2 * 2^8 2^1011 = 2^0001 * 2^0010 * 2^1000 第二种方法：累乘，时间复杂度为O(n) 【参考代码】

```
package javaTest;
public class JavaTest {
    public static void main(String[] args) {
        System.out.println(power(3, 3));
        System.out.println(powerAnother(3, 3));
    }
    // 使用递归
    public static double power(double base, int exponent) {
```

```
        int n = Math.abs(exponent);
        double result = 0.0;
        if (n == 0)
            return 1.0;
        if (n == 1)
            return base;

        result = power(base, n >> 1);
        result *= result;
        if ((n & 1) == 1) // 如果指数n为奇数，则要再乘一次底数base
            result *= base;
        if (exponent < 0) // 如果指数为负数，则应该求result的倒数
            result = 1 / result;

        return result;
    }
    // 使用累乘
    public static double powerAnother(double base, int exponent) {
        double result = 1.0;
        for (int i = 0; i < Math.abs(exponent); i++) {
            result *= base;
        }
        if (exponent >= 0)
            return result;
        else
            return 1 / result;
    }

}
```