29. Divide Two Integers

Description | Hints | Submissions | Discuss | Solution

Pick One

Given two integers `dividend` and `divisor`, divide two integers without using multiplication, division and mod operator.

Return the quotient after dividing `dividend` by `divisor`.

The integer division should truncate toward zero.

Example 1:

```
Input: dividend = 10, divisor = 3
Output: 3
```

Example 2:

```
Input: dividend = 7, divisor = -3
Output: -2
```

Note:

- Both dividend and divisor will be 32-bit signed integers.
- The divisor will never be 0.
- Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: $[-2^{31}, 2^{31} - 1]$. For the purpose of this problem, assume that your function returns $2^{31} - 1$ when the division result overflows.

Seen this question in a real interview before? Yes No

```java
public class L29 {
    /*
     * 这是一个两数相除的问题，而且规定我们不能用乘法、除法和取余操作，那么可以用位操作，
     * 思路是，如果被除数大于或等于除数，则进行如下循环，定义变量t等于除数，定义计数p，当t的两倍小于
     * 等于被除数时，进行如下循环，t扩大一倍，p扩大一倍，然后更新res和m。这道题的oj给的一写test case，如被除数
     * 是-2147483648，除数是-1，这样结果就超出了int范围，需要返回Integer.MAX_VALUE。然后我们还要根据
     * 被除数和除数的正负来确定返回值的正负，这里采用长整型long来完成所有的计算，最后返回值乘以符号即可。
     */
    public int divide(int dividend, int divisor) {
        long res = 0;
        long m = Math.abs((long)dividend), n =  Math.abs((long)divisor);
        if (m < n) return 0;
        long t = n, p = 1;
        while (m > (t << 1)) {
            t <<= 1;
            p <<= 1;
        }
        res += p + divide((int)(m - t), (int)n);  //假如被除数为10，除数为2，第一次循环后，t=8，p=4，最后被除数还剩2，这个递归
        if ((dividend < 0) ^ (divisor < 0)) res = -res; //^符号是异或符号
        return res > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)res;
    }

    public static void main(String [] args) {
        System.out.println(new L29().divide(-2147483648, -1));
    }
}
```