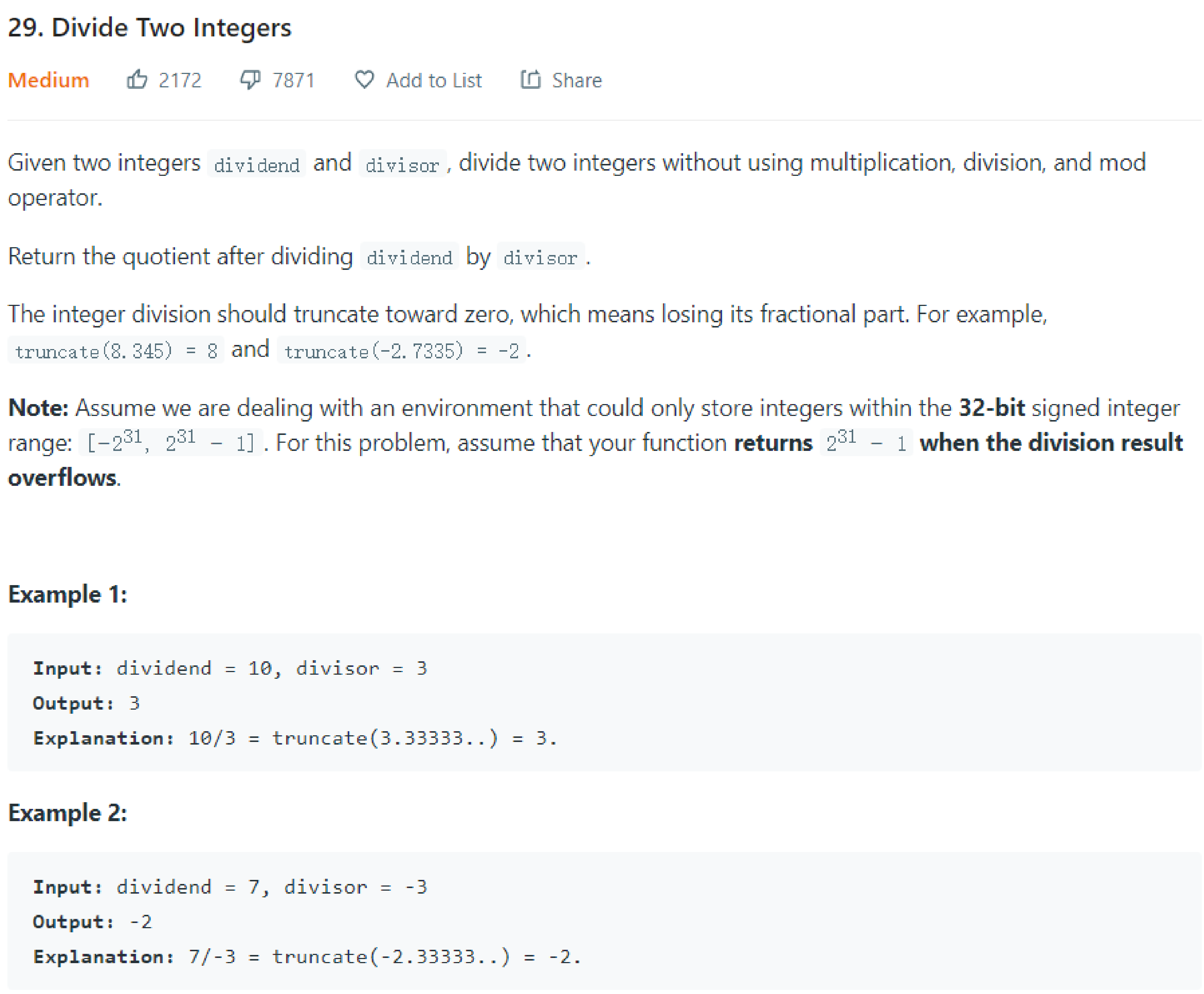
Divide Two Integers

Yuxiao Chen 6118492



29. Divide Two Integers

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, which means losing its fractional part. For example, `truncate(8.345) = 8` and `truncate(-2.7335) = -2`.

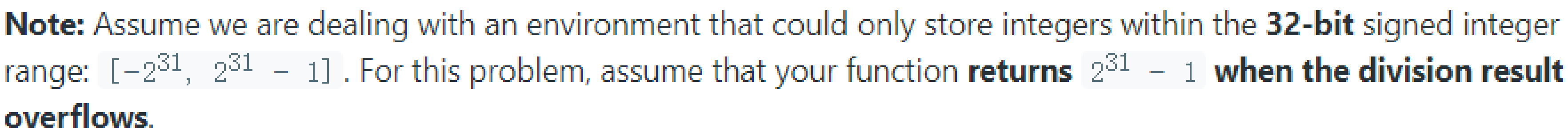
Note: Assume we are dealing with an environment that could only store integers within the **32-bit** signed integer range: $[-2^{31}, 2^{31} - 1]$. For this problem, assume that your function **returns** $2^{31} - 1$ **when the division result overflows**.

Example 1:

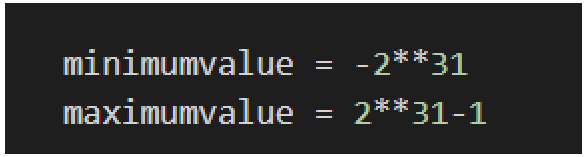
```
Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = truncate(3.33333..) = 3.
```

Example 2:

```
Input: dividend = 7, divisor = -3
Output: -2
Explanation: 7/-3 = truncate(-2.33333..) = -2.
```

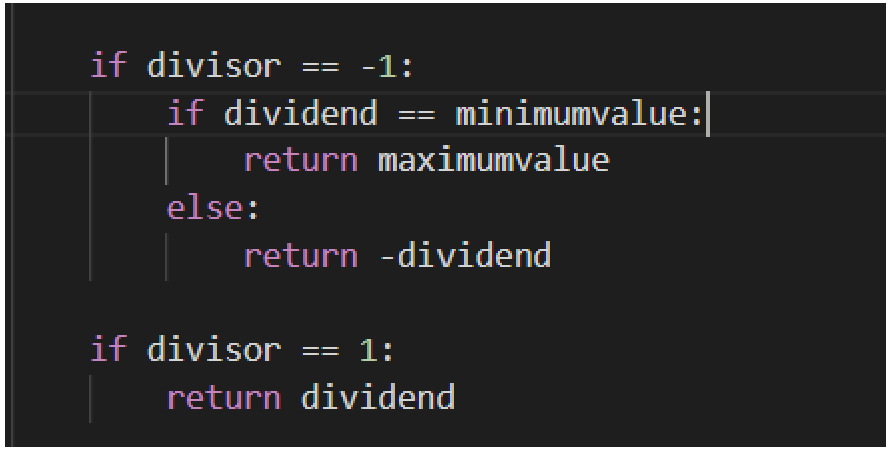


Note: Assume we are dealing with an environment that could only store integers within the **32-bit** signed integer range: $[-2^{31}, 2^{31} - 1]$. For this problem, assume that your function **returns** $2^{31} - 1$ **when the division result overflows**.

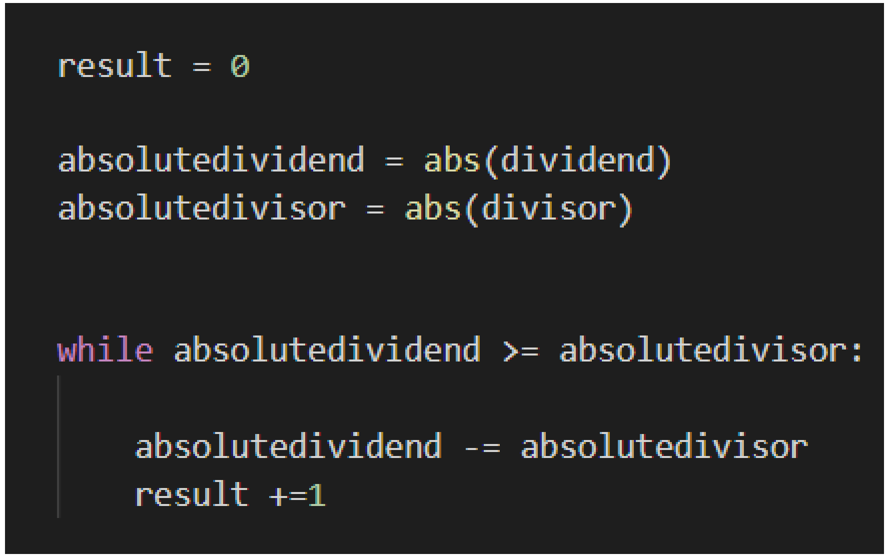


```
minimumvalue = -2**31
maximumvalue = 2**31-1
```

If the divisor is 1, the output should be the same with the dividend.
If the divisor is -1, the output should be the –dividend.
If the divisor is -1, the dividend is the minimum value, it should be output the maximum value.

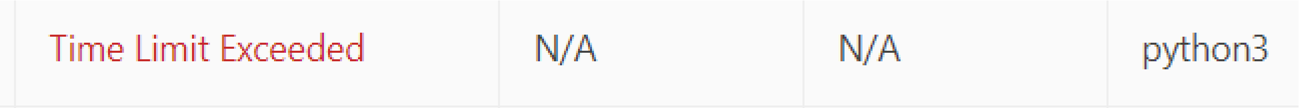


```
if divisor == -1:
    if dividend == minimumvalue:
        return maximumvalue
    else:
        return -dividend

if divisor == 1:
    return dividend
```



```
result = 0

absolutedividend = abs(dividend)
absolutedivisor = abs(divisor)


while absolutedividend >= absolutedivisor:

    absolutedividend -= absolutedivisor
    result +=1
```

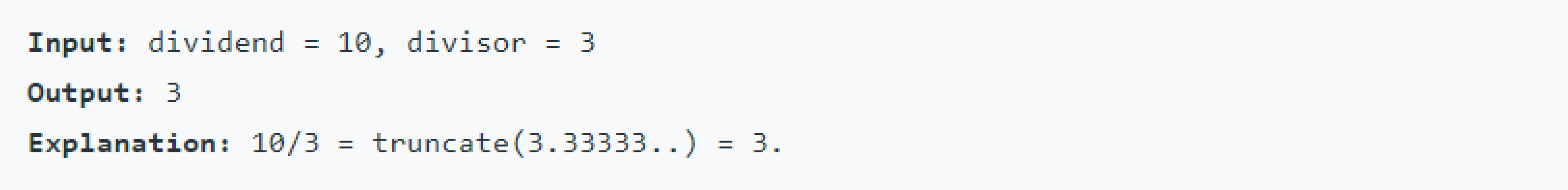
Use absolute number to determine the size of dividend and divisor.



Time Limit Exceeded	N/A	N/A	python3

If the dividend is very large and the divisor is very small, it will be execute for a long time.

- Then change the mind from how to make dividend to be the same with divisor to how to make the divisor be the same with the dividend.



```
Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = truncate(3.33333..) = 3.
```

-31 – 31

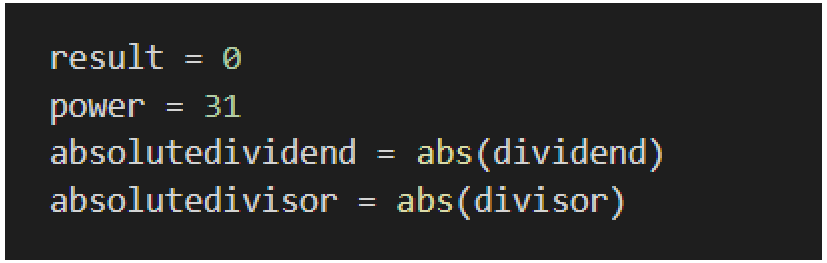
$$\underset{3}{\text{divisor}} \times 2^{\boxed{n}} \rightarrow \underset{10}{\text{dividend}}$$

$$3 \times 2^2 = 12$$

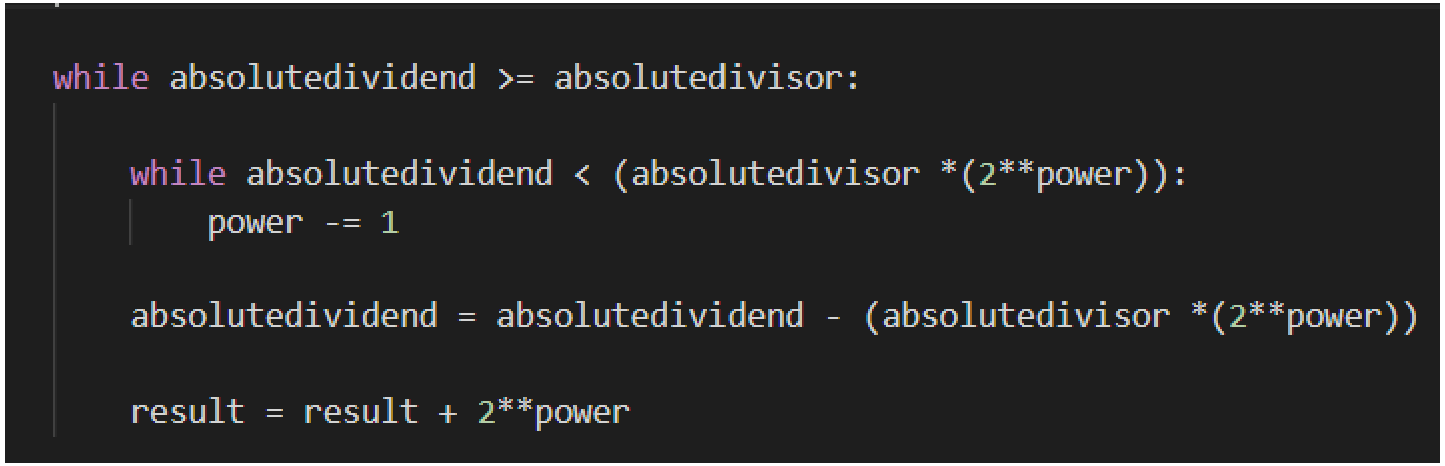
$$3 \times \boxed{2^1} = 6 \rightarrow 10 - 6 = 4 > \text{divisor}$$

$$3 \times \boxed{2^0} = 3 \qquad 4 - 3 = 1 < \text{divisor}$$

$$10/3 = 2^1 + 2^0 = 3$$

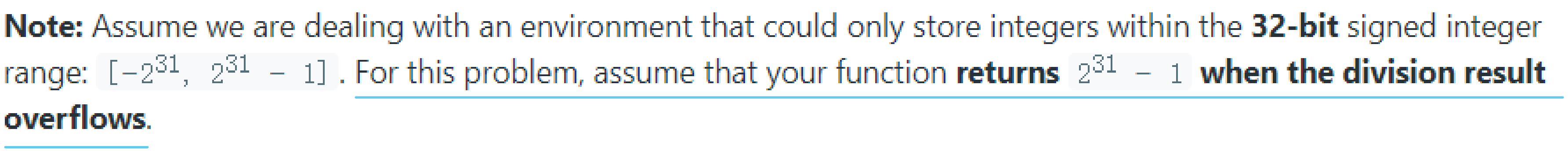


```
result = 0
power = 31
absolutedividend = abs(dividend)
absolutedivisor = abs(divisor)
```

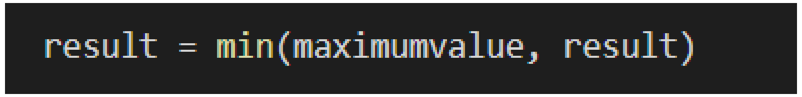


```
while absolutedividend >= absolutedivisor:

    while absolutedividend < (absolutedivisor *(2**power)):
        power -= 1

    absolutedividend = absolutedividend - (absolutedivisor *(2**power))

    result = result + 2**power
```

If absolute dividend < absolute divisor, result will be 0.XX, then it will output 0.

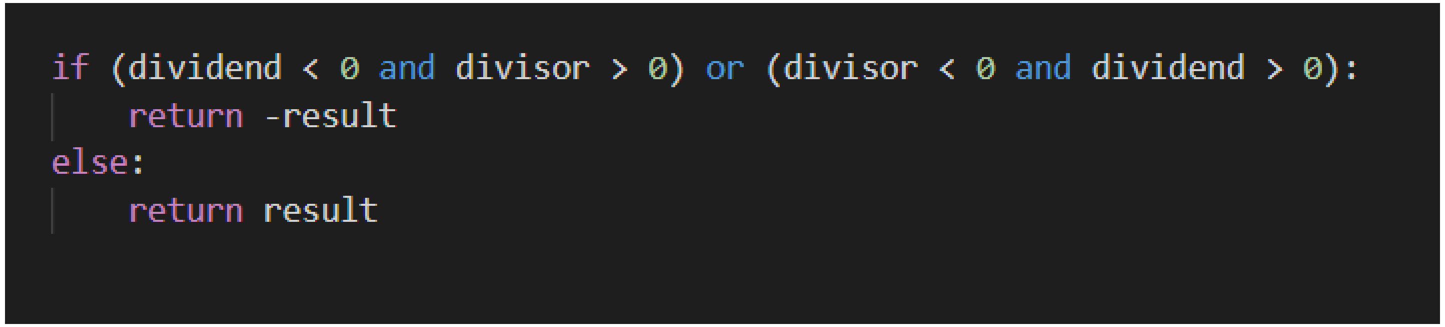


Note: Assume we are dealing with an environment that could only store integers within the **32-bit** signed integer range: $[-2^{31}, 2^{31} - 1]$. For this problem, assume that your function **returns** $2^{31} - 1$ **when the division result overflows**.



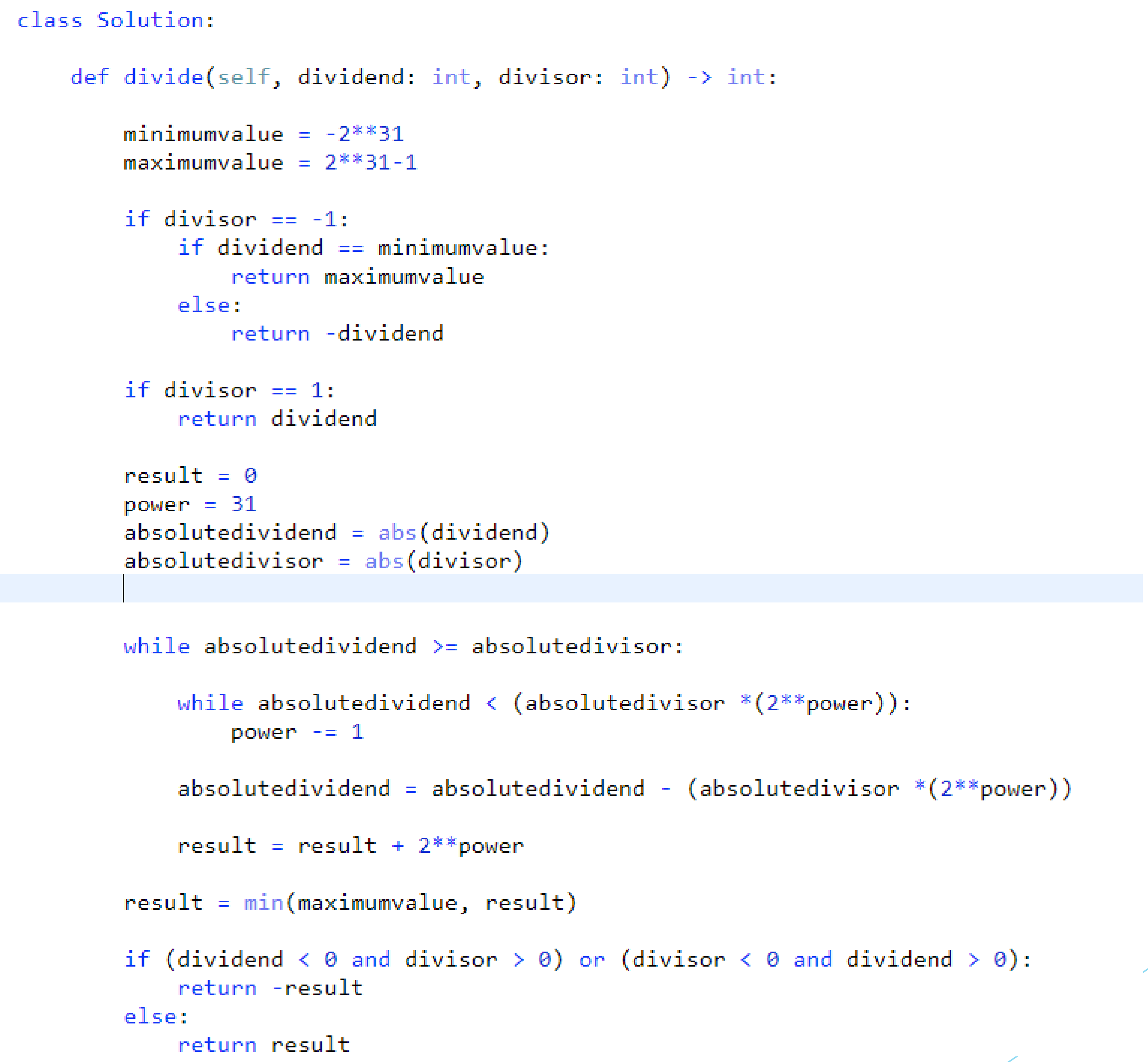
```
result = min(maximumvalue, result)
```

When the division result > maximum value, it will output the maximum value instead.

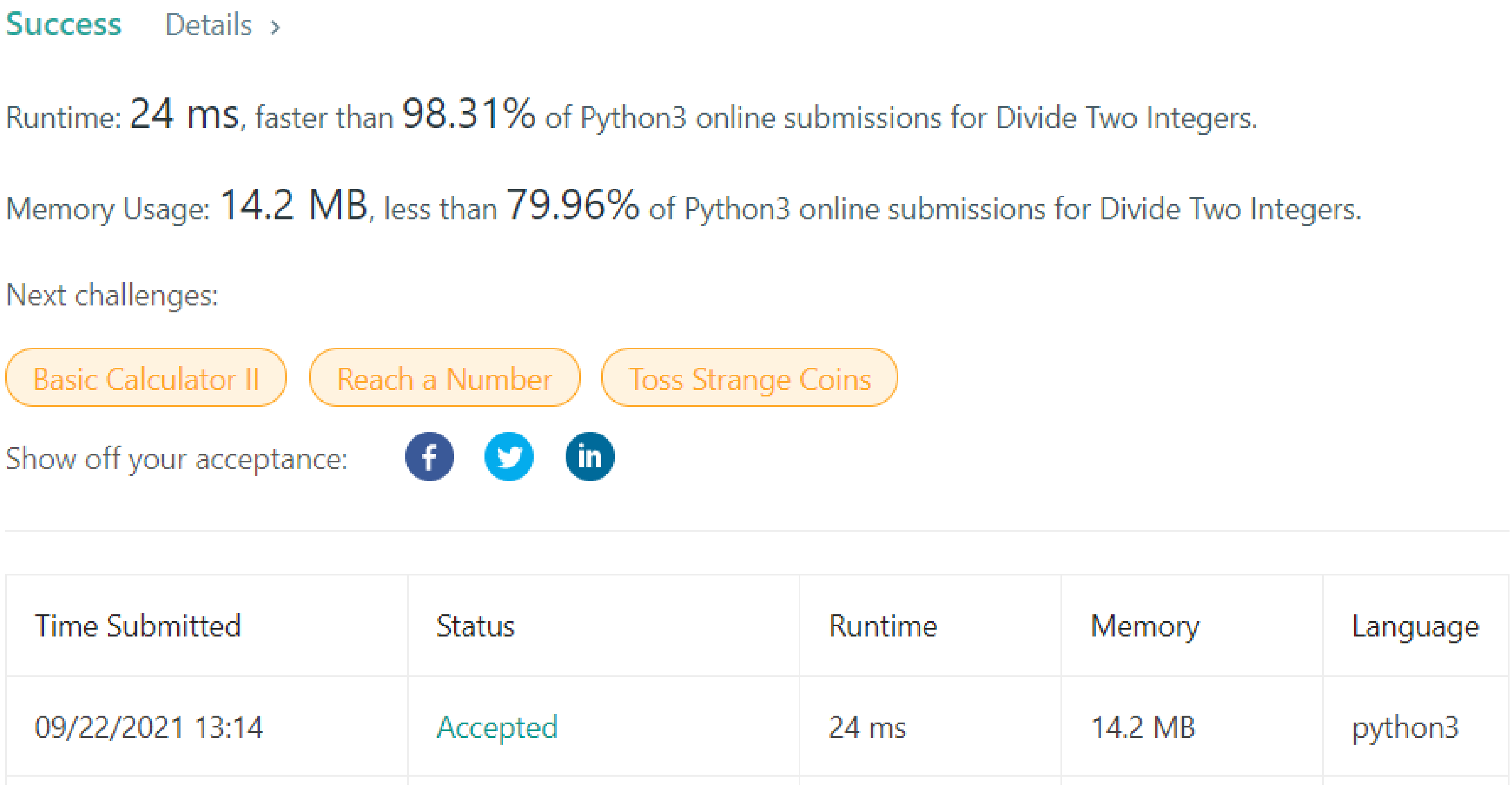


```
if (dividend < 0 and divisor > 0) or (divisor < 0 and dividend > 0):
    return -result
else:
    return result
```

To judge the result is positive or negative by simple math way.



```
class Solution:

    def divide(self, dividend: int, divisor: int) -> int:

        minimumvalue = -2**31
        maximumvalue = 2**31-1

        if divisor == -1:
            if dividend == minimumvalue:
                return maximumvalue
            else:
                return -dividend

        if divisor == 1:
            return dividend

        result = 0
        power = 31
        absolutedividend = abs(dividend)
        absolutedivisor = abs(divisor)


        while absolutedividend >= absolutedivisor:

            while absolutedividend < (absolutedivisor *(2**power)):
                power -= 1

            absolutedividend = absolutedividend - (absolutedivisor *(2**power))

            result = result + 2**power

        result = min(maximumvalue, result)

        if (dividend < 0 and divisor > 0) or (divisor < 0 and dividend > 0):
            return -result
        else:
            return result
```



Runtime: 24 ms, faster than 98.31% of Python3 online submissions for Divide Two Integers.

Memory Usage: 14.2 MB, less than 79.96% of Python3 online submissions for Divide Two Integers.

Next challenges:

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
09/22/2021 13:14	Accepted	24 ms	14.2 MB	python3

Thank you