Description | Editorial | Solutions (3K) | Submissions

29. Divide Two Integers

Medium | 4.3K | 13.4K

Companies

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

The integer division should truncate toward zero, which means losing its fractional part. For example, `8.345` would be truncated to `8`, and `-2.7335` would be truncated to `-2`.

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

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, if the quotient is **strictly greater than** $2^{31} - 1$, then return $2^{31} - 1$, and if the quotient is **strictly less than** -2^{31}, then return -2^{31}.

Example 1:

```
Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = 3.33333.. which is truncated to 3.
```

Example 2:

```
Input: dividend = 7, divisor = -3
Output: -2
Explanation: 7/-3 = -2.33333.. which is truncated to
-2.
```

Constraints:

Java | Auto

```java
class Solution {
    public int divide(int dividend, int divisor) {
        if (dividend == 1<<31 && divisor == -1) return Integer.MAX_VALUE;

        boolean sign = (dividend>=0) == (divisor >=0) ? true : false;
        dividend = Math.abs(dividend);
        divisor = Math.abs(divisor);
        int result =0;

        while(dividend - divisor >=0){
            int count =0;
            while(dividend - (divisor << 1 << count) >= 0 ){
                count++;
            }
            result +=1 <<count;
            dividend -= divisor <<count;
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

dividend =
10

divisor =
3

Output

3

Expected

Console | Run | Submit

Description | Editorial | Solutions (1.9K) | Submissions

1603. Design Parking System

Hint

Easy | 1.6K | 409

Companies

Design a parking system for a parking lot. The parking lot has three kinds of parking spaces: big, medium, and small, with a fixed number of slots for each size.

Implement the `ParkingSystem` class:

- `ParkingSystem(int big, int medium, int small)` Initializes object of the `ParkingSystem` class. The number of slots for each parking space are given as part of the constructor.
- `bool addCar(int carType)` Checks whether there is a parking space of `carType` for the car that wants to get into the parking lot. `carType` can be of three kinds: big, medium, or small, which are represented by `1`, `2`, and `3` respectively. **A car can only park in a parking space of its** `carType`. If there is no space available, return `false`, else park the car in that size space and return `true`.

Example 1:

Input
`["ParkingSystem", "addCar", "addCar", "addCar", "addCar"]`
`[[1, 1, 0], [1], [2], [3], [1]]`
Output
`[null, true, true, false, false]`

Explanation
```
ParkingSystem parkingSystem = new ParkingSystem(1, 1, 0);
parkingSystem.addCar(1); // return true because there is 1 available slot for a big car
```

Java | Auto

```java
class ParkingSystem {
    int[] space;

    public ParkingSystem(int big, int medium, int small) {
        space = new int[]{big, medium, small};
    }

    public boolean addCar(int carType) {
        if(space[carType - 1] == 0){
            return false;
        }

        space[carType - 1]--;
        return true;
    }
}
```

Press Esc to exit full screen

Testcase | Result

Accepted Runtime: 0 ms

Case 1

Input

`["ParkingSystem","addCar","addCar","addCar","addCar"]`

`[[1,1,0],[1],[2],[3],[1]]`

Output

`[null,true,true,false,false]`

Expected

`[null,true,true,false,false]`

Console | Run | Submit