

## Divide Two Integers

This problem is tricky because multiplication and modulus operators are disallowed. We must handle signs, overflow and efficiently carefully.

- [Key Points:]
1. Constraints: 32-bit signed integer range  $[-2^{31}, 2^{31}-1]$
  2. Truncate toward zero (same as integer division in C++)
  3. Special Case:  $\text{INT\_MIN} / -1$  overflows  $\rightarrow$  return  $\text{INT\_MAX}$

[Optimal Approach: Bit Manipulation / Repeated Doubling]:

- [Idea:] Use subtraction, but optimised with bit shifts (doubling strategy) for  $O(\log n)$  complexity:
- Repeatedly subtract the largest multiple of divisor (power of 2 factor) from dividend.
  - Accumulate the result.

- [Steps:]
1. Handle overflow ( $\text{INT\_MIN} / -1$ ).
  2. Determine sign of the result.
  3. Work with absolute values (long long to avoid overflow during abs).
  4. While dividend  $\geq$  divisor:
    - Find largest power of two multiple of divisor that fits into dividend.
    - Subtract it and add the multiple to result.
  5. Apply sign and clamp to 32-bit range.

- [Complexity:]
- Time:  $O(\log(|\text{dividend}|))$  - doubling reduces dividend quickly.
  - Space:  $O(1)$ .