

Reverse Signed Number - Solution Report

★ Approach

The problem requires reversing the digits of a 32-bit signed integer while ensuring the reversed number does not exceed the 32-bit signed integer range. The approach involves:

1. Handling the sign of the number separately.
2. Reversing the digits of the absolute value of the number.
3. Checking for overflow before returning the result.
4. Combining the sign with the reversed digits.

★ Solution Code

python

```
def reverse(n: int) -> int:
    sign = -1 if n < 0 else 1
    reversed_num = 0
    n = abs(n)
    while n > 0:
        digit = n % 10
        reversed_num = reversed_num * 10 + digit
        n = n // 10
    reversed_num *= sign
    if reversed_num < -2**31 or reversed_num > 2**31 - 1:
        return 0
    return reversed_num
...
```

★ Explanation

1. ****Sign Handling****: The sign of the input number is stored separately to reapply it after reversing the digits.
2. ****Digit Reversal****: The absolute value of the number is processed to reverse its digits. This is done by repeatedly extracting the last digit of the number and building the reversed number.
3. ****Overflow Check****: After reversing, the result is checked to ensure it lies within the 32-bit signed integer range. If it exceeds this range, the function returns 0.
4. ****Result Construction****: The stored sign is applied to the reversed number, and the final result is returned.

.