Reverse Integer

Problem

Given a 32-bit signed integer, reverse digits of an integer.

example 1:

Input: 123
Output: 321

example 2:

Input: -123
Output: -321

example 3:

Input: 120
Output: 21

Note:
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 0 when the reversed integer overflows.

Solution

Approach 1: Pop and Push Digits & Check before Overflow

Intuition

We can build up the reverse integer one digit at a time. While doing so, we can check beforehand whether or not appending another digit would cause overflow.

Algorithm

We want to repeatedly "pop" the last digit off of xx and "push" it to the back of the rev. In the end, rev will be the reverse of the x.
To explain, lets assume that rev is positive.

- If $temp = rev * 10 + pop$ causes overflow, then it must be that $rev \geq \frac{INTMAX}{10}$
- If $rev > \frac{INTMAX}{10}$, then $temp = rev * 10 + pop$ is guaranteed to overflow.
- If $rev == \frac{INTMAX}{10}$, then $temp = rev * 10 + pop$ will overflow if and only if $pop > 7$

Java(Official)

```
class Solution {
    public int reverse(int x) {
```

```
        int rev = 0;
        while (x != 0) {
            int pop = x % 10;
            x /= 10;
            if (rev > Integer.MAX_VALUE/10 || (rev == Integer.MAX_VALUE / 10 && pop > 7))
return 0;
            if (rev < Integer.MIN_VALUE/10 || (rev == Integer.MIN_VALUE / 10 && pop < -8))
return 0;
            rev = rev * 10 + pop;
        }
        return rev;
    }
}
```

Java(My Own)

```
class Solution {
    public int reverse(int x) {
        Boolean negative = false;
        if(x < 0){
            x = -1 * x;
            negative = true;
        }
        int y = 0;
        while(x > 0){
            if(y> 214748364)
                return 0;
            y = (x % 10) + y * 10;
            x = x / 10;
        }
        if(negative)
            y = -1 * y;

        return y;
    }
}
```

Note: actually when rev > 214748364, pop won't be larger than 1 because the input must be a reasonable Integer.

Python

```
class Solution(object):
    def reverse(self, x):
    """
    :type x: int
    :rtype: int
    """
    overflow_pos=pow(2,31)-1
    overflow_neg=float(-pow(2,31))

        rev=0
        while x!=0:
            trunc=int(float(x)/10)
            pop=x-trunc*10
            x=trunc
            if rev > overflow_pos/10 or (rev==overflow_pos//10 and pop > 7): return 0
            if rev < overflow_neg/10 or (rev==int(overflow_neg/10) and pop < -8): return 0
            rev=rev*10+pop
        return rev
```