Palindrome Number

Problem

Determine whether an integer is a palindrome. An integer is a palindrome when it reads the same backward as forward.

example 1:

Input: 121
Output: true

example 2:

Input: -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

example 3:

Input: 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

Solution

Approach 1: Revert the number itself

Intuition

An idea would be reverting the number itself, and then compare the number with original number, if they are the same, then the number is a palindrome. However, if the reversed number is larger than int.MAX, we will hit integer overflow problem. So we could check the overflow problem before completly reversing the number. Same as the Reverse Integer Problem.
Java

```
class Solution {
    public boolean isPalindrome(int x) {
        if(x < 0)
            return false;
        if(x == reverse(x))
            return true;
        else
            return false;
    }

    public int reverse(int x){
        int rev = 0;
        int pop;
        while(x > 0){
            pop = x % 10;
            if (rev > Integer.MAX_VALUE / 10)
                return 0;
            rev = rev * 10 + pop;
            x = x / 10;
        }
        return rev;
    }
}
```

Complexity Analysis

- Time Complexity: $O(log_{10}(n))$
- Space Complexity: $O(1)$

Approach 2: Revert half of the number

Intuition

To avoid the overflow problem, we could only revert half of the \text{int}int number.

Algorithm

First of all we should take care of some edge cases. All negative numbers are not palindrome.
Then we have to check if we have reached the half of the number. Since we divided the number by 10, and multiplied the reversed number by 10, when the original number is less than the reversed number, it means we've processed half of the number digits.

Java

```java
class Solution {
    public boolean isPalindrome(int x) {
        if(x < 0 || (x % 10 == 0 && x != 0)) {
            return false;
        }

        int revertedNumber = 0;
        while(x > revertedNumber) {
            revertedNumber = revertedNumber * 10 + x % 10;
            x /= 10;
        }

        return x == revertedNumber || x == revertedNumber/10;
    }
}
```

Python

```python
class Solution(object):
    def isPalindrome(self, x):
        """
        :type x: int
        :rtype: bool
        """
        if x < 0 or (x != 0 and x % 10 == 0):
            return False
        revertedNumber = 0
        while x > revertedNumber:
            revertedNumber = revertedNumber * 10 + x % 10
            x /= 10

        return x == revertedNumber or x == revertedNumber / 10
```

Complexity Analysis

- Time Complexity: $O(log_{10}(n))$
- Space Complexity: $O(1)$