Evaluate Reverse Polish Notation

Difficulty	Medium
Category	Stack
Question	https://leetcode.com/problems/evaluate-reverse-polish-notation/description/
Solution	https://www.youtube.com/watch?v=iu0082c4HDE
Status	Done

Question

> You are given an array of strings `tokens` that represents an arithmetic expression in a Reverse Polish Notation.
>
> Evaluate the expression. Return *an integer that represents the value of the expression*.
>
> **Note** that:
>
> - The valid operators are `'+'`, `'-'`, `'*'`, and `'/'`.
> - Each operand may be an integer or another expression.
> - The division between two integers always **truncates toward zero**.
> - There will not be any division by zero.
> - The input represents a valid arithmetic expression in a reverse polish notation.
> - The answer and all the intermediate calculations can be represented in a **32-bit** integer.

Example

Example 1:

```
Input: tokens = ["2","1","+","3","*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

```
Input: tokens = ["4","13","5","/","+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6
```

Idea

Use two Stacks: One for the normal implementation, another one to hold the minimum of the stack at the top always

Solution

```
class Solution:
    def evalRPN(self, tokens: List[str]) -> int:
        # Initialize an empty stack to store operands.
        stack = []

        # Iterate through each token in the input list of tokens.
        for c in tokens:
            if c == "+":
                # If the token is "+", pop the top two elements from the stack,
                # add them, and push the result back onto the stack.
                stack.append(stack.pop() + stack.pop())

            elif c == "-":
                # If the token is "-", pop the top two elements from the stack,
                # subtract the first popped element from the second popped element,
                # and push the result back onto the stack.
                a, b = stack.pop(), stack.pop()
                stack.append(b - a)

            elif c == "*":
                # If the token is "*", pop the top two elements from the stack,
                # multiply them, and push the result back onto the stack.
                stack.append(stack.pop() * stack.pop())

            elif c == "/":
                # If the token is "/", pop the top two elements from the stack,
                # perform division, convert the result to an integer (floor division),
                # and push the result back onto the stack.
                a, b = stack.pop(), stack.pop()
                stack.append(int(float(b / a)))
```

```
            else:
                # If the token is not an operator, convert it to an integer and push it onto the stack.
                stack.append(int(c))

        # After processing all tokens, the result will be the only element left on the stack.
        return stack[0]
```

💡 The **Time complexity** of the provided code is **O(n).**

💡 The **Space complexity** of the code is **O(n)** as well.