[Click on the logo to find **Problem Statement**]

Intuition

This code implements the evaluation of Reverse Polish Notation (RPN) expressions using a stack.The algorithm iterates through each element in the input list of tokens. For each token, it checks whether it is an operator (+, -, *, /) or a number. If it's an operator, it performs the corresponding operation using the top elements of the stack and updates the stack. If it's a number, it simply pushes it onto the stack. The final result is the only element left in the stack after processing all tokens. The stack efficiently keeps track of the intermediate results, reflecting the nature of postfix notation.

Approach

1. **Initialize Stack:**
    - Create an empty stack to keep track of intermediate results during the evaluation.
2. **Iterate through Tokens:**
    - Start iterating through each token in the given list.
3. **Token Type Check:**
    - For each token, check whether it is an operator (+, -, *, /) or a number.
4. **Operator Handling:**
    - If the token is an operator, pop the top two operands from the stack, perform the corresponding operation, and push the result back onto the stack. For subtraction and division, ensure the correct operand order by popping them in reverse.
5. **Number Handling:**
    - If the token is a number, convert it to an integer and push it onto the stack.
6. **Final Result:**
    - The final result is the only element left in the stack after processing all tokens.

Complexity

- Time complexity: O(n)
- Space complexity: O(n)

Code

```
class Solution:
    def evalRPN(self, tokens: List[str]) -> int:
        stack = []
        for i in tokens:
            if i == "+":
                stack.append(stack.pop() + stack.pop())
            elif i == "-":
                a,b = stack.pop(), stack.pop()
                stack.append(b - a)
            elif i == "*":
                stack.append(stack.pop() * stack.pop())
            elif i == "/":
                a,b = stack.pop(), stack.pop()
                stack.append(int(b / a))
            else:
                stack.append(int(i))
        return stack[0]
```

If you want to see more solutions to coding problems, you can visit: