227. Basic Calculator II

Given a string `s` which represents an expression, *evaluate this expression and return its value*.

The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of `[-2<sup>31</sup>, 2<sup>31</sup> - 1]`.

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as `eval()`.

Example 1:

```
Input: s = "3+2*2"
Output: 7
```

Example 2:

```
Input: s = " 3/2 "
Output: 1
```

Example 3:

```
Input: s = " 3+5 / 2 "
Output: 5
```

Constraints:

- `1 <= s.length <= 3 * 10<sup>5</sup>`
- `s` consists of integers and operators `('+', '-', '*', '/')` separated by some number of spaces.
- `s` represents **a valid expression**.
- All the integers in the expression are non-negative integers in the range `[0, 2<sup>31</sup> - 1]`.
- The answer is **guaranteed** to fit in a **32-bit integer**.

```
class Solution:
    def calculate(self, s: str) -> int:
        precedence = {
            '+':1,
            '-':1,
```

```
            '*':3,
            '/':3
        }
        s = ''.join(s.split(' '))
        number = []
        operators = []
        i = 0
        while i<len(s):
            ch = s[i]
            if ch not in precedence:
                num,idx = self.getNumber(i,s)
                number.append(int(num))
                i = idx
            elif ch in precedence:
                while len(operators) and precedence[ch]
<=precedence[operators[-1]]:
                    v2 = number.pop()
                    v1 = number.pop()
                    op = operators.pop()
                    temp = self.evaluate(v1,v2,op)
                    number.append(temp)
                operators.append(ch)
                i+=1

        while len(operators):
            v2 = number.pop()
            v1 = number.pop()
            op = operators.pop()
            temp = self.evaluate(v1,v2,op)
            number.append(temp)
        return number[-1]


    def getNumber(self,idx,s):
        num = ''
        temp = idx
        while temp < len(s):
            ch = s[temp]
            if ch in {'1','2','3','4','5','6','7','8','9','0'}:
                num+=s[temp]
                temp+=1
            else:
                break
```

```
        return num,temp




    def evaluate(self,v1,v2,char):
        if char == '+':
            return v1+v2
        elif char == '-':
            return v1-v2
        elif char == '*':
            return v1*v2
        else:
            return v1//v2
```