```
def evalPostfix(pharse):
    try:
        stack = Stack()
        l = pharse.split(' ')
        history = []
        history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
        i = 0
        while i < len(l): ⇨ O(n) بار فراخوانی n
            item = l[i]
            if item == '+':
                stack.push(stack.pop() + stack.pop())
                for __ in range(3):
                    l.pop(i)
                    i -= 1
                i += 1
                l = l[:i] + [str(stack.top())] + l[i:]
                history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
            elif item == '-':
                stack.push(-1 * (stack.pop() - stack.pop()))
                for __ in range(3):
                    l.pop(i)
                    i -= 1
                i += 1
                l = l[:i] + [str(stack.top())] + l[i:]
                history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
            elif item == '*':
                stack.push(stack.pop() * stack.pop())
                for __ in range(3):
                    l.pop(i)
                    i -= 1
```

```
        i += 1
        l = l[:i] + [str(stack.top())] + l[i:]
        history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
    elif item == '/':
        x, y = stack.pop(), stack.pop()
        stack.push(y / x)
        for __ in range(3):
            l.pop(i)
            i -= 1
        i += 1
        l = l[:i] + [str(stack.top())] + l[i:]
        history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
    elif item == '^':
        x, y = stack.pop(), stack.pop()
        stack.push(math.pow(y, x))
        for __ in range(3):
            l.pop(i)
            i -= 1
        i += 1
        l = l[:i] + [str(stack.top())] + l[i:]
        history.append(postfixToInfix(l)) ⇨ O(n)  فراخوانی تابع با پیچیدگی زمانی
    else:
        counter = 0
        for char in item:
            if char == '.':
                counter += 1
        if counter > 1:
            raise Exception('The number is wrong.')
        stack.push(float(item))
    i += 1
```

```
        return history
    except:
        return 'error'
```

پس پیچیدگی تابع می شود $O(n^2)$

```
def postfixToInfix(list):
    changer = Stack()
    for k in list: ⇨ O(n) بار فراخوانی n
        if k in ['+', '-', '*', '/', '^']:
            b = changer.pop()
            a = changer.pop()
            add_str = '(' + a + k + b + ')'
            changer.push(add_str)
        else:
            changer.push(k)
    return changer.pop()
```

پس پیچیدگی تابع می شود $O(n)$