



数据结构和算法 (Python描述)

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕!

讲义照片均为郭炜拍摄



栈的应用



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

栈的概念和实现



北京坡峰岭红叶

栈的概念

- 类似于子弹匣，后压进去的子弹，先射出去

- 支持四种操作：

<code>top()</code>	返回栈顶元素
<code>push(x)</code>	将x压入栈中
<code>pop()</code>	弹出并返回栈顶元素
<code>isEmpty ()</code>	看栈是否为空

- 要求上面操作复杂度都是 $O(1)$

栈的实现

- 用列表可以实现栈
- 四种操作的实现(stack为一个列表)：

<code>top()</code>	<code>stack[-1]</code>
<code>push(x)</code>	<code>stack.append(x)</code>
<code>pop()</code>	<code>stack.pop()</code>
<code>isEmpty ()</code>	<code>len(stack) == 0</code>



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

例题1 字符串中的括号配对



天山大峡谷

例题1：字符串中的括号配对

●字符串中可能有三种成对的括号，"()"、"[]"、"{}".判断字符串的括号是否都正确配对了(不存在括号也算正确配对)。括号交叉算不正确配对，例如"12{34[78}ab]"。但是一对括号被包含在另一对括号里面，例如"12{ab[8]}"不影响正确性。

解题思路：

从头到尾扫描字符串，碰到左括号就入栈。碰到右括号，就要求栈顶必须是一个和它配对的左括号，如果不是，则断定字符串不符合要求。如果是，则弹出栈顶。

字符串扫描结束时，栈为空则为正确，不为空则为错误

例题1：字符串中的括号配对

```
def match(s):    #复杂度O(n)
    stack = []
    pairs = {")": "(", "]" : "[", "]" : "{" }
    for x in s:
        if x in "([{":
            stack.append(x)
        elif x in ")]}":
            if len(stack) == 0 or stack[-1] != pairs[x]:
                return False
            stack.pop()
    return len(stack) == 0

print(match(input()))
```




北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

例题2 后序表达式求值



喀纳斯湖

例题2: 后序表达式求值

●运算符: $+$ $-$ $*$ $/$ 其中 $*$ $/$ 优先级高于 $+$ $-$

●原子: 整数或者小数

●后序表达式递归定义:

1) 一个原子是一个后序表达式

2) 两个后序表达式 a 、 b 加上一个运算符 c , 是一个后序表达式。

a b c 的计算方法和传统的中序表达式 (a) c (b) 一样

●以下都是后序表达式 (原子、运算符之间用空格分隔)

3.4

5

5 3.4 + 等价于 $5 + 3.4$

5 3.4 + 6 / 等价于 $(5 + 3.4) / 6$

5 3.4 + 6 * 3 + 等价于 $(5 + 3.4) * 6 + 3$

例题2: 后序表达式求值

- 求给定后序表达式的值

从左到右扫描一遍后序表达式，碰到原子就入栈，碰到运算符，就取出栈顶两个元素进行运算，并将结果压入栈中。

扫描结束时，栈里应该只有一个元素，就是后序表达式的值

例题2: 后序表达式求值

```
def countSuffix(s):    #计算后序表达式s的值, 复杂度O(n)
    s = s.split()
    stack = []
    for x in s:
        if x in "+-*/*":
            a,b = stack.pop(),stack.pop()
            stack.append(eval(str(b) + x + str(a)))
        else:
            stack.append(float(x))
    return stack[0]
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

例题3

中序表达式转换成后序
表达式



黄花城水长城

例题3: 中序表达式转换成后序表达式

给定一个中序表达式，求转成后序表达式。
原子、运算符和括号之间用空格分隔。

输入样例：

3 * (4 + 5) * (6 + 7) + (5 - 2) * 1 + 3

输出样例：

3 4 5 + * 6 7 + * 5 2 - 1 * + 3 +

例题3: 中序表达式转换成后序表达式

思路:

- ① 一个最简单的两原子中序表达式, 转换成后序表达式时, 原子顺序不变。 因此复杂的中序表达式, 转换成后序表达式时, 原子顺序也不变。
- ② 用列表result存放转换的结果, 则扫描中序表达式时, 碰到原子, 就添加到result
- ③ 每个运算符都有两个操作数, 操作数为原子或带括号的中序表达式。扫描时碰到运算符, 不能直接将其添加到result (因为其第二个操作数还未出现), 应存放在某处(栈中), 以便合适的时候找出来添加到result
- ④ 扫描到运算符a时, 如果发现栈顶的运算符优先级不_低于a, 则将栈顶运算符添加到result (此刻result的尾部应该是一个原子或者一个刚生成的后序表达式), 直到栈为空或者栈顶运算符优先级低于 a, 然后将a入栈
- ⑤ 扫描结束时, 将栈中所有元素取出添加到result

括号表达式的处理:

- 1) 括号表达式当作一个中序表达式处理
- 2) 碰到 "(" 则将其入栈, 然后开始处理中序表达式的过程。碰到 ")" 时, 认为该中序表达式处理结束, 应该将栈顶元素不停弹出添加到 result, 直到 ")" 出栈

例题3: 中序表达式转换成后序表达式

```
def midToSuffix(s):
    s = s.split()
    stack, result = [], []
    priority = {"/": 1, "*": 1, "+": 2, "-": 2}
    for x in s:
        if x == "(":
            stack.append(x)
        elif x == ")":
            while stack[-1] != "(":
                result.append(stack.pop())
            stack.pop()
        elif x in "/*+-":
            while len(stack) >= 1 and \
                stack[-1] != '(' and priority[stack[-1]] <= priority[x]:
                result.append(stack.pop())
            stack.append(x)
        else:
            result.append(x)
```

例题3: 中序表达式转换成后序表达式

```
while stack != []:  
    result.append(stack.pop())  
return " ".join(result)
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

例题4 中序表达式求值



富士山

例题4: 中序表达式求值

要求不能转换成后序表达式再求。

- ① 维护原子栈stkNum和运算符栈stkOp两个栈
- ② 碰到原子就加入原子栈
- ③ 碰到运算符a，如果stkOp顶部运算符x优先级小于a，则x的第二个操作数应该是a运算进行后的结果，此时还没有出现，因此应该将a压入stkOp；如果stkOp顶部运算符优先级不小于a，则取出该运算符，并取出stkNum顶部两个原子进行运算，运算结果压回stkNum。直到stkOp顶部运算符优先级小于a，或stkOp顶部为"("，则将a压入stkOp
- ④ 扫描结束后，从stkOp不停弹出运算符进行运算，直到stkOp为空，则stkNum中应该只剩下一个元素，即整个表达式的值
- ⑤ "("压入运算符栈，然后开始一个中序表达式的处理过程。碰到")"视为一个中序表达式扫描结束，要对stkOp进行退栈并运算，直到 "("被弹出

例题4: 中序表达式求值

```
def countMid(s):  
    s = s.split()  
    stkNum, stkOp = [], [] #原子栈、运算符栈  
    priority = {"/": 1, "*": 1, "+": 2, "-": 2}  
    for x in s:  
        if x == "(":  
            stkOp.append(x)  
        elif x == ")":  
            while stkOp[-1] != "(":  
                op = stkOp.pop()  
                a, b = stkNum.pop(), stkNum.pop()  
                stkNum.append(eval(str(b) + op + str(a)))  
            stkOp.pop()  
        elif x in "/*+-":  
            while len(stkOp) >= 1 and \  
                stkOp[-1] != '(' and priority[stkOp[-1]] <= priority[x]:  
                op = stkOp.pop()
```

例题4: 中序表达式求值

```
        a,b = stkNum.pop(),stkNum.pop()
        stkNum.append(eval(str(b) + op + str(a)))
    stkOp.append(x)
    else:
        stkNum.append(float(x))
while len(stkOp) > 0:
    op = stkOp.pop()
    a,b = stkNum.pop(),stkNum.pop()
    stkNum.append(eval(str(b) + op + str(a)))
return stkNum[0]

print(countMid("3 * ( 4 + 5 ) * ( 6 + 7 ) + ( 5 - 2 ) * 1 + 3"))
```