# 如何判定括号合法性




相关推荐：

- 回溯算法解题套路框架
- Linux shell 的实用小技巧

读完本文，你不仅学会了算法套路，还可以顺便去 LeetCode 上拿下如下题目：

20.有效的括号

---

对括号的合法性判断是一个很常见且实用的问题，比如说我们写的代码，编辑器和编译器都会检查括号是否正确闭合。而且我们的代码可能会包含三种括号 `[](){}`，判断起来有一点难度。

本文就来聊一道关于括号合法性判断的算法题，相信能加深你对**栈**这种数据结构的理解。

题目很简单，输入一个字符串，其中包含 `[](){}` 六种括号，请你判断这个字符串组成的括号是否合法。

```
Input: "()[]{}"
Output: true

Input: "([)]"
Output: false

Input: "{[]}"
Output: true
```

解决这个问题之前，我们先降低难度，思考一下，**如果只有一种括号** `()`，应该如何判断字符串组成的括号是否合法呢？

## 一、处理一种括号

字符串中只有圆括号，如果想让括号字符串合法，那么必须做到：

**每个右括号 `)` 的左边必须有一个左括号 `(` 和它匹配。**

比如说字符串 `())(()` 中，中间的两个右括号**左边**就没有左括号匹配，所以这个括号组合是不合法的。

那么根据这个思路，我们可以写出算法：

```
bool isValid(string str) {
    // 待匹配的左括号数量
    int left = 0;
    for (char c : str) {
        if (c == '(')
            left++;
        else // 遇到右括号
            left--;

        if (left < 0)
            return false;
    }
    return left == 0;
}
```

如果只有圆括号，这样就能正确判断合法性。对于三种括号的情况，我一开始想模仿这个思路，定义三个变量 `left1`，`left2`，`left3` 分别处理每种括号，虽然要多写不少 if else 分支，但是似乎可以解决问题。

但实际上直接照搬这种思路是不行的，比如说只有一个括号的情况下 `(())` 是合法的，但是多种括号的情况下，`[()]` 显然是不合法的。

仅仅记录每种左括号出现的次数已经不能做出正确判断了，我们要加大存储的信息量，可以利用栈来模仿类似的思路。

## 二、处理多种括号

栈是一种先进后出的数据结构，处理括号问题的时候尤其有用。

我们这道题就用一个名为 `left` 的栈代替之前思路中的 `left` 变量，**遇到左括号就入栈，遇到右括号就去栈中寻找最近的左括号，看是否匹配**。

```
bool isValid(string str) {
    stack<char> left;
    for (char c : str) {
        if (c == '(' || c == '{' || c == '[')
            left.push(c);
        else // 字符 c 是右括号
            if (!left.empty() && leftOf(c) == left.top())
                left.pop();
            else
                // 和最近的左括号不匹配
                return false;
    }
    // 是否所有的左括号都被匹配了
    return left.empty();
}

char leftOf(char c) {
```

```
    if (c == '}') return '{';
    if (c == ')') return '(';
    return '[';
}
```

---

**刷算法，学套路，认准 labuladong，公众号和 在线电子书 持续更新最新文章。**

**本小抄即将出版，微信扫码关注公众号，后台回复「小抄」限时免费获取，回复「进群」可进刷题群一起刷题，带你搞定 LeetCode。**

==其他语言代码==

## Python3

```
def isValid(self, s: str) -> bool:
    left = []
    leftOf = {
        ')':'(',
        ']':'[',
        '}':'{'
    }
    for c in s:
        if c in '([{':
            left.append(c)
        elif left and leftOf[c]==left[-1]: # 右括号 + left不为空 + 和最近左括号能匹配
            left.pop()
        else: # 右括号 + （left为空 / 和堆顶括号不匹配）
            return False

    # left中所有左括号都被匹配则return True 反之False
    return not left
```

```
//基本思想：每次遇到左括号时都将相对应的右括号')', ']'或'}'推入堆栈
//如果在字符串中出现右括号，则需要检查堆栈是否为空，以及顶部元素是否与该右括号相同。如果不是，则该字符串无效。
```

```java
//最后，我们还需要检查堆栈是否为空
public boolean isValid(String s) {
  Deque<Character> stack = new ArrayDeque<>();
  for(char c : s.toCharArray()){
    //是左括号就将相对应的右括号入栈
    if(c=='(') {
      stack.offerLast(')');
    }else if(c=='{'){
      stack.offerLast('}');
    }else if(c=='['){
      stack.offerLast(']');
    }else if(stack.isEmpty() || stack.pollLast()!=c){//出现右括号，检查堆栈是否为空，以及顶部元素是否与该右括号相同
      return false;
    }
  }
  return stack.isEmpty();
}
```