题目描述

(1+(2+3)*(3+(8+0))+1-2)这是一个简单的数学表达式,今天不是计算它的值,而是比较它的括号匹配是否正确。

前面这个式子可以简化为((()(())))这样的括号我们认为它是匹配正确的,

而(((()这样的我们就说他是错误的。注意括号里面的表达式可能是错的,也可能有多个空格,对于这些我们是不用去管的,

我们只关心括号是否使用正确。

输入描述

给出一行表达式(长度不超过 100)。

输出描述

如果匹配正确输出括号的对数,否则输出-1。

用例

输入	(1+(2+3)*(3+(8+0))+1-2)
输出	4
说明	无

题目解析

本题就是括号匹配的变种题,只是加入了一些干扰字符,我们可以用正则去掉非()的字符,然后利用栈结构校验括号是否成对,可以参考

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const regExp = /[^\(\)]/g;
  line = line.replace(regExp, "");
  let count = 0;
  const stack = [];
  for (let c of line) {
    if (stack.length && c === ")") {
      if (stack.at(-1) === "(") {
        stack.pop();
        count++;
        continue;
      } else {
        return console.log(-1);
      }
    }
    stack.push(c);
  }

  if (stack.length) return console.log(-1);

  return console.log(count);
});
```

Java算法源码

```java
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String s = sc.nextLine();
    System.out.println(getResult(s));
  }

  public static int getResult(String s) {
    s = s.replaceAll("[^\\(\\)]", "");

    int count = 0;
    LinkedList<Character> stack = new LinkedList<>();

    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);

      if (stack.size() > 0 && c == ')') {
        if (stack.getLast() == '(') {
          stack.removeLast();
          count++;
          continue;
        }
        return -1;
      }

      stack.add(c);
    }

    if (stack.size() > 0) return -1;
    return count;
  }
}
```

Python算法源码

```python
# 输入获取
import re

s = input()


# 算法入口
def getResult(s):
    s = re.sub(r"[^\(\)]", "", s)

    count = 0
    stack = []

    for c in s:
        if len(stack) > 0 and c == ')':
            if stack[-1] == '(':
                stack.pop()
                count += 1
                continue
            return -1

        stack.append(c)

    if len(stack) > 0:
        return -1

    return count


# 算法调用
print(getResult(s))
```