

04-D

栈与队列

括号匹配

邓俊辉

deng@tsinghua.edu.cn

实例

❖ $(a[i-1][j+1] + a[i+1][j-1]) * 2$ //失配

$(a[i-1][j+1] + a[i+1][j-1]) * 2$ //匹配

❖ 观察：除了各种括号，其余符号均可暂时忽略

$([] []) [] [])$ //失配

$([] [] [] [])$ //匹配

❖ 从简单入手，先来考查只有一种括号的情况...

尝试：由外而内

0) 平凡： 无括号的表达式是匹配的

1) 减治？ E 匹配，仅当 (E) 匹配

2) 分治？ E 和 F 均匹配，仅当 E F 匹配

❖ 然而，根据以上性质，却不易直接应用已知的策略

❖ 究其根源在于，1) 和2) 均为**充分性**，比如反例：

(() ()) () = (() ()) ()

(() ()) () = (() ()) ()

❖ 而为使问题有效简化，必须发现并借助**必要性**

构思：由内而外

❖ 颠倒以上思路：消去一对**紧邻**的左右括号，不影响全局的匹配判断

亦即：

L

()

R

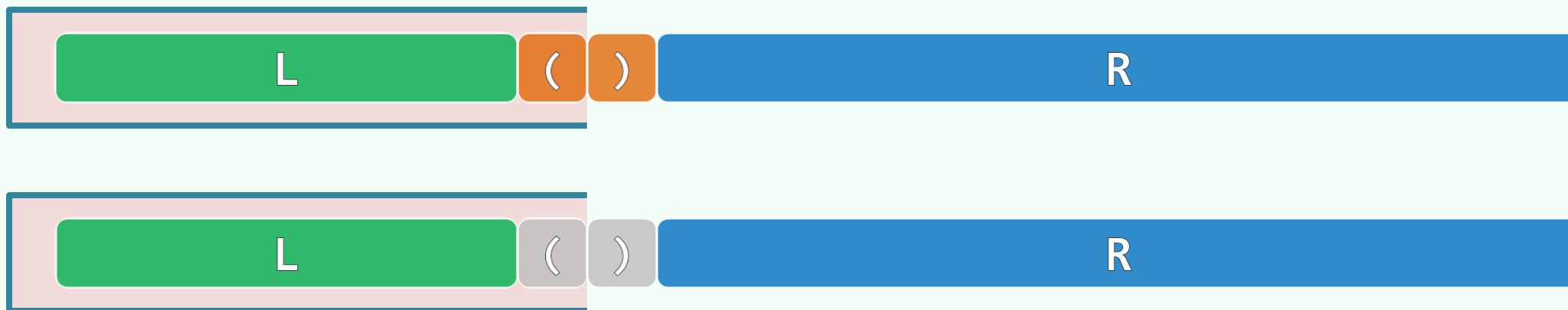
 匹配，仅当

L

R

 匹配

❖ 那么，如何**找到**这对括号？再者，如何使问题的这种简化得以**持续**进行？



❖ 顺序扫描表达式，用栈记录已扫描的部分

//实际上只需记录左括号

反复迭代：凡遇"**(**"，则进栈；凡遇"**)**"，则出栈

实现

```
bool paren( const char exp[], Rank lo, Rank hi ) { //exp[lo, hi)

    Stack<char> S; //使用栈记录已发现但尚未匹配的左括号

    for ( Rank i = lo; i < hi; i++ ) //逐一检查当前字符

        if ( '(' == exp[i] ) S.push( exp[i] ); //遇左括号：则进栈

        else if ( ! S.empty() ) S.pop(); //遇右括号：若栈非空，则弹出对应的左括号

        else return false; //否则（遇右括号时栈已空），必不匹配

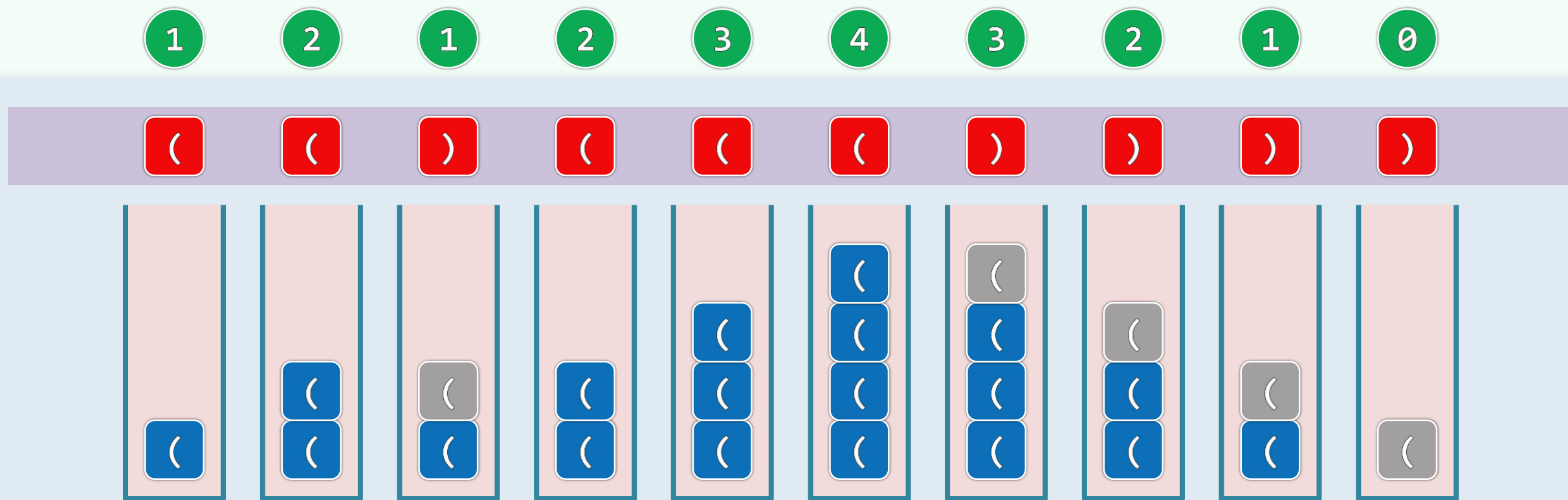
    return S.empty(); //最终栈空，当且仅当匹配

}
```

实例：一种括号

❖ 实际上，若仅考虑一种括号，只需一个**计数器**足矣：`s.size()`

❖ 一旦转负，则为失配（右括号多余）；最后归零，即为匹配（否则左括号多余）



拓展：多类括号

❖ 以上思路及算法，可否推广至多种括号并存的情况？反例： `[(])`

❖ 甚至，只需约定“括号”的**通用**格式，而不必事先**固定**括号的类型与数目

比如：`<body>|</body>`，`<h1>|</h1>`，`|`，`<p>|</p>`，`|`，...

