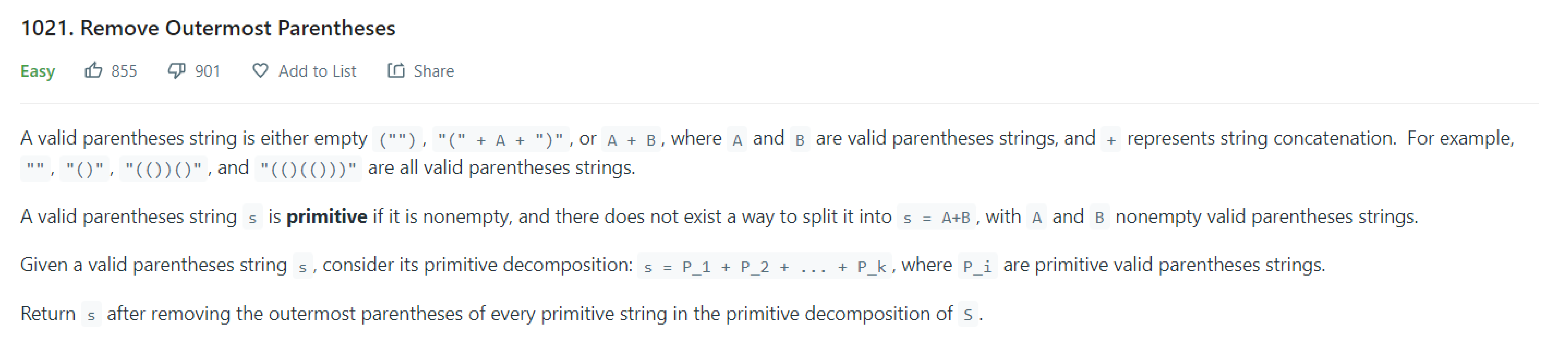


1021. Remove Outermost Parentheses

A valid parentheses string is either empty `("")`, `"(" + A + ")"`, or `A + B`, where `A` and `B` are valid parentheses strings, and `+` represents string concatenation. For example, `""`, `"()"`, `"(())()"`, and `"(()(()))"` are all valid parentheses strings.

A valid parentheses string `s` is **primitive** if it is nonempty, and there does not exist a way to split it into `s = A+B`, with `A` and `B` nonempty valid parentheses strings.

Given a valid parentheses string `s`, consider its primitive decomposition: `s = P_1 + P_2 + ... + P_k`, where `P_i` are primitive valid parentheses strings.

Return `s` after removing the outermost parentheses of every primitive string in the primitive decomposition of `s`.

Example 1:

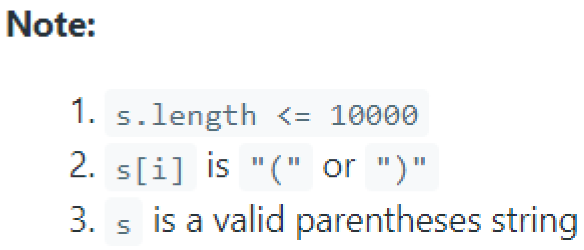
```
Input: s = "(()())(())"
Output: "()()()"
Explanation:
The input string is "(()())(())", with primitive decomposition "(()())" + "(())".
After removing outer parentheses of each part, this is "()()" + "()" = "()()()".
```

Example 2:

```
Input: s = "(()())(())(()(()))"
Output: "()()()()(())"
Explanation:
The input string is "(()())(())(()(()))", with primitive decomposition "(()())" + "(())" + "(()(()))".
After removing outer parentheses of each part, this is "()()" + "()" + "()(())" = "()()()()(())".
```

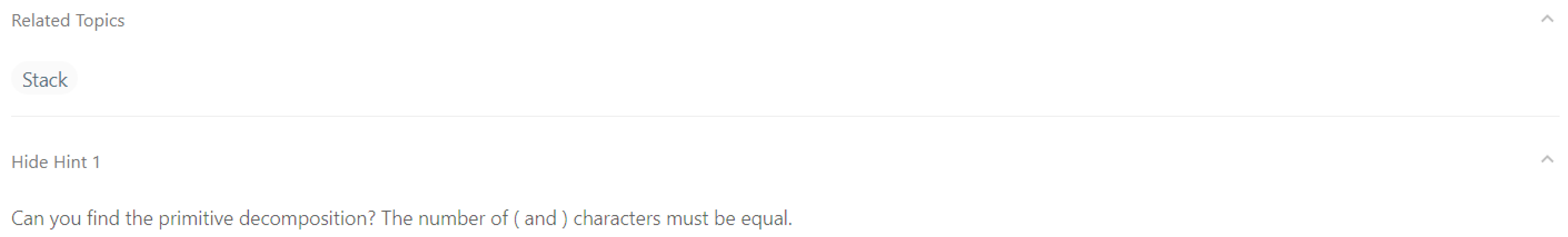
Example 3:

```
Input: s = "()()"
Output: ""
Explanation:
The input string is "()()", with primitive decomposition "()" + "()".
After removing outer parentheses of each part, this is "" + "" = "".
```



Note:

1. `s.length <= 10000`
2. `s[i]` is `"("` or `")"`
3. `s` is a valid parentheses string



Related Topics ^

Stack

Hide Hint 1 ^

Can you find the primitive decomposition? The number of (and) characters must be equal.