

1190. Reverse Substrings Between Each Pair of Parentheses

You are given a string `s` that consists of lower case English letters and brackets.

Reverse the strings in each pair of matching parentheses, starting from the innermost one.

Your result should **not** contain any brackets.

Example 1:

Input: `s = "(abcd)"`

Output: `"dcba"`

Example 2:

Input: `s = "(u(love)i)"`

Output: `"iloveu"`

Explanation: The substring "love" is reversed first, then the whole string is reversed.

Example 3:

Input: `s = "(ed(et(oc))el)"`

Output: `"leetcode"`

Explanation: First, we reverse the substring "oc", then "etco", and finally, the whole string.

Example 4:

Input: `s = "a(bcdefghijkl(mno)p)q"`

Output: `"apmnoIkjihgfedcbq"`

Constraints:

- `0 <= s.length <= 2000`
- `s` only contains lower case English characters and parentheses.
- It's guaranteed that all parentheses are balanced

```
from collections import deque
class Solution:
    def reverseParentheses(self, s: str) -> str:
        stack = []
```

```
queue = deque()
for i in range(len(s)):
    if s[i]!=')':
        stack.append(s[i])
    else:
        if s[i]==')':
            while stack[-1]!='(':
                queue.append(stack.pop())
            stack.pop()
            while len(queue)>0:
                temp = queue.popleft()
                stack.append(temp)
return ''.join(stack)
```