18) VALID SUBSTRING

METHOD 1 (USING STACKS - O(N) space complexity):
LINK OF THE EXPLANATION: Valid Substring || GeeksforGeeks || Problem of the Day || S…
In this sum, we will be using the stacks as the sum is based on the balanced parenthesis substring. So In this sum we take the help of two stacks,one storing the indices of the pushed element and other storing the characters of the actual string.
We perform the entire process of the balanced parentheses string and while pushing any character into the character stack we simultaneously push the index of that character into the integer stack.
Once we are done iterating the string,if the stack is empty we return the length of the string because in that case the entire string will be balanced and if the stack is not empty then we start popping the elements out from the integer stack and start finding the maximum difference among the popped indices and return the maximum of the difference.

METHOD 2:(Efficient)
LINK OF EXPLANATION: Longest Valid Parentheses | Live Coding with Explanation | Leetc…

CODE OF THE METHOD 1:

```
class Solution {
  public:
    int findMaxLen(string s) {
        stack<char> sc;
        stack<int> si;
        for(int i=0;i<s.length();i++){
            if(s[i]=='('){
                sc.push(s[i]);
                si.push(i);
            }
            else{
                if(sc.size()!=0 && sc.top()=='('){
                    sc.pop();
                    si.pop();
                }
                else{
                    sc.push(s[i]);
                    si.push(i);
                }
            }
        }
        if(sc.size()==0){
            return s.length();
        }
        else{
            int previous=s.length();
            int maximum=0;
            while(si.size()!=0){
                maximum=max(maximum,previous-1-si.top());
                previous=si.top();
                si.pop();
                sc.pop();
            }
            maximum=max(maximum,previous);
            return maximum;
        }
    }
};
```

CODE OF METHOD 2:

```
class Solution {
  public:
    int findMaxLen(string s) {
        int open=0,close=0;
        int maximum=0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='('){
                open++;
            }
            else{
                close++;
            }
            if(open==close){
                maximum=max(maximum,open*2);
            }
            else if(close>open){
                open=0;
                close=0;
            }
        }
        open={close=0};
        for(int i=s.length()-1;i>=0;i--){
            if(s[i]=='('){
                open++;
            }
            else{
                close++;
            }
            if(close==open){
                maximum=max(maximum,open*2);
            }
            else if(close<open){
                open=0;
                close=0;
            }
        }
        return maximum;
    }
};
```