921. Minimum Add to Make Parentheses Valid

A parentheses string is valid if and only if:

- It is the empty string,
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or
- It can be written as `(A)`, where `A` is a valid string.

You are given a parentheses string `s`. In one move, you can insert a parenthesis at any position of the string.

- For example, if `s = "()))"`, you can insert an opening parenthesis to be `"(()))"` or a closing parenthesis to be `"())))"`.

Return *the minimum number of moves required to make* `s` *valid*.

Example 1:

```
Input: s = "())"
Output: 1
```

Example 2:

```
Input: s = "((("
Output: 3
```

Constraints:

- `1 <= s.length <= 1000`
- `s[i]` is either `'('` or `')'`.

921. Minimum Add to Make Parentheses Valid

```
/*
    Stack
    Time complexity: O(n)
    Space complexity: O(n)
*/
class Solution {
    public:
        int minAddToMakeValid(std::string s) {
            std::stack<char> st;
            bool push;
            for(char& c: s){
                push=true;
                if(c==')'){
                    if(!st.empty() && st.top()=='(')
st.pop(),push=false;
                }
                if(push) st.push(c);
            }
            return st.size();
        }
};
```

921. Minimum Add to Make Parentheses Valid

```
/*
    Space optimization
    Time complexity: O(n)
    Space complexity: O(1)
*/
class Solution {
    public:
        int minAddToMakeValid(std::string s) {
            int open=0;
            int close=0;
            for (char& c : s){
                if (c=='(') open++;
                else open>0?open--:close++;
            }
            return close+open;
        }
};
```