Educational Codeforces Round 4

Problema C: *Replace To Make Regular Bracket Sequence*

Prof. Edson Alves – UnB/FGA

Problema

You are given string s consists of opening and closing brackets of four kinds `<>`, `{}`, `[]`, `()`. There are two types of brackets: opening and closing. You can replace any bracket by another of the same type. For example, you can replace `<` by the bracket `{`, but you can't replace it by `)` or `>`.

The following definition of a regular bracket sequence is well-known, so you can be familiar with it.

Let's define a regular bracket sequence (RBS). Empty string is RBS. Let s_1 and s_2 be a RBS then the strings `<`s_1`>`s_2, {s_1} s_2, [s_1]s_2, (s_1)s_2 are also RBS.

For example the string "`[[(){}]<>]`"is RBS, but the strings "`[)()`"and "`][()()`"are not.

Determine the least number of replaces to make the string s RBS.

Entrada e saída

Input

The only line contains a non empty string s, consisting of only opening and closing brackets of four kinds. The length of s does not exceed 10^6.

Output

If it's impossible to get RBS from s print `Impossible`.

Otherwise print the least number of replaces needed to get RBS from s.

Exemplo de entradas e saídas

Sample Input	Sample Output
`[<})){}`	`2`
`{()}[]`	`0`
`]]`	`Impossible`

Solução com complexidade $O(N)$

- Para que a string seja uma RBS, é preciso que cada símbolo aberto seja fechado pelo símbolo correspondente
- Em tais expressões, o símbolo de fechar será associado ao símbolo de abrir correspondente mais próximo à esquerda
- Logo, os símbolos ainda em aberto devem ser armazenados em uma pilha
- A cada símbolo de fechar, o topo da pilha deve ser consultado: se não for o símbolo correspondente, uma substituição deve ser feita
- Caso a pilha esteja vazia, ou tenha ao menos um símbolo pendente após o processamento de toda string s, a sequência não corresponde a uma RBS
- Como cada caractere é avaliado, no máximo, 2 vezes (uma na inserção e outra na remoção da pilha), esta solução tem complexidade $O(N)$

Solução AC com complexidade $O(N)$

```cpp
#include <bits/stdc++.h>

using namespace std;

map<char, char> open { { '>', '<' }, { ']', '[' }, { '}', '{' }, { ')', '(' } };

int solve(const string& S)
{
    stack<char> st;
    int ans = 0;

    for (const auto& c : S)
    {
        switch (c) {
        case '(':
        case '<':
        case '{':
        case '[':
            st.push(c);
            break;
```

Solução AC com complexidade $O(N)$

```
22          default:
23              if (st.empty())
24                  return -1;
25
26              ans += (open[c] == st.top() ? 0 : 1);
27              st.pop();
28          }
29      }
30
31      return st.empty() ? ans : -1;
32  }
33
34  int main()
35  {
36      ios::sync_with_stdio(false);
37
38      string S;
39      cin >> S;
40
41      auto ans = solve(S);
```

Solução AC com complexidade $O(N)$

```
43      if (ans == -1)
44          cout << "Impossible\n";
45      else
46          cout << ans << '\n';
47
48      return 0;
49  }
```