2696. Minimum String Length After Removing Substrings

You are given a string `s` consisting only of **uppercase** English letters.

You can apply some operations to this string where, in one operation, you can remove **any** occurrence of one of the substrings `"AB"` or `"CD"` from `s`.

Return *the **minimum** possible length of the resulting string that you can obtain*.

Note that the string concatenates after removing the substring and could produce new `"AB"` or `"CD"` substrings.

Example 1:

```
Input: s = "ABFCACDB"
Output: 2
Explanation: We can do the following operations:
- Remove the substring "ABFCACDB", so s = "FCACDB".
- Remove the substring "FCACDB", so s = "FCAB".
- Remove the substring "FCAB", so s = "FC".
So the resulting length of the string is 2.
It can be shown that it is the minimum length that we can obtain.
```

Example 2:

```
Input: s = "ACBBD"
Output: 5
Explanation: We cannot do any operations on the string so the length remains the same.
```

Constraints:

- `1 <= s.length <= 100`
- `s` consists only of uppercase English letters.

2696. Minimum String Length After Removing Substrings

```
/*
    Straight forward approach
    Time complexity: O(n^2)
    Space complexity: O(1)
*/
class Solution {
    public:
        int minLength(std::string s){
            bool flag=true;
            while(flag){
                flag=false;
                string::size_type i=s.find("AB");
                if (i!=string::npos){
                    flag=true;
                    s.erase(i,2);
                }
                i=s.find("CD");
                if (i!=string::npos){
                    flag=true;
                    s.erase(i,2);
                }
            }
            return s.size();
        }
};
```

2696. Minimum String Length After Removing Substrings

```
/*
   Stack
   Time complexity: O(n)
   Space complexity: O(n)
*/
class Solution {
   public:
      int minLength(std::string s){
         std::stack<char> st;
         bool push=true; // To know what to push
         for(char& c: s){
            push=true; // By default we will push the current character
            if(!st.empty() && c=='B'){ // if current character is 'B'
               // if the previuos character is 'A', remove it and
               // no need to push 'B'
               if(st.top()=='A') st.pop(),push=false;
            }
            else if(!st.empty() && c=='D'){ // if current character is 'D'
               // if the previuos character is 'C', remove it and
               // no need to push 'D'
               if(st.top()=='C') st.pop(),push=false;
            }

            if(push) st.push(c);
         }

         return st.size();
      }
};
```