Problem 1896: Minimum Cost to Change the Final Value of Expression

Problem Information

Difficulty: Hard
Acceptance Rate: 51.23%
Paid Only: No
Tags: Math, String, Dynamic Programming, Stack

Problem Description

You are given a **valid** boolean expression as a string `expression` consisting of the characters `'1'`,`'0'`,`'&'` (bitwise **AND** operator),`'|'` (bitwise **OR** operator),`'('`, and `')'`.

* For example, `"()1|1"` and `"(1)&()"` are **not valid** while `"1"`, `"(((1))|(0))"`, and `"1|(0&(1))"` are **valid** expressions.

Return _the**minimum cost** to change the final value of the expression_.

* For example, if `expression = "1|1|(0&0)&1"`, its **value** is `1|1|(0&0)&1 = 1|1|0&1 = 1|0&1 = 1&1 = 1`. We want to apply operations so that the**new** expression evaluates to `0`.

The **cost** of changing the final value of an expression is the **number of operations** performed on the expression. The types of **operations** are described as follows:

* Turn a `'1'` into a `'0'`. * Turn a `'0'` into a `'1'`. * Turn a `'&'` into a `'|'`. * Turn a `'|'` into a `'&'`.

Note: `'&'` does **not** take precedence over `'|'` in the **order of calculation**. Evaluate parentheses **first** , then in **left-to-right** order.

Example 1:

Input: expression = "1&(0|1)" **Output:** 1 **Explanation:** We can turn "1&(0 _**|**_ 1)" into "1&(0 _**&**_ 1)" by changing the '|' to a '&' using 1 operation. The new expression evaluates to 0.

Example 2:

Input: expression = "(0&0)&(0&0&0)" **Output:** 3 **Explanation:** We can turn "(0 _**& 0**_)**_&_**(0&0&0)" into "(0 _**|1**_)_**|**_(0&0&0)" using 3 operations. The new expression evaluates to 1.

Example 3:

Input: expression = "(0|(1|0&1))" **Output:** 1 **Explanation:** We can turn "(0|(_**1**_ |0&1))" into "(0|(_**0**_ |0&1))" using 1 operation. The new expression evaluates to 0.

Constraints:

* `1 <= expression.length <= 105` * `expression` only contains `'1'`,`'0'`,`'&'`,`'|'`,`'('`, and `')'` * All parentheses are properly matched. * There will be no empty parentheses (i.e: `"()"` is not a substring of `expression`).

Code Snippets

C++:

```
class Solution {
public:
int minOperationsToFlip(string expression) {

}
};
```

Java:

```
class Solution {
public int minOperationsToFlip(String expression) {

}
}
```

Python3:

```
class Solution:
def minOperationsToFlip(self, expression: str) -> int:
```