Boolean Parenthesization

Given a boolean expression **S** of length **N** with following symbols.
Symbols
'T' ---> true
'F' ---> false
and following operators filled between symbols
Operators
& ---> boolean AND
| ---> boolean OR
^ ---> boolean XOR
Count the number of ways we can parenthesize the expression so that the value of expression evaluates to true.

Example 1:

```
Input: N = 7
S = T|T&F^T
Output: 4
Explaination: The expression evaluates
to true in 4 ways ((T|T)&(F^T)),
(T|(T&(F^T))), (((T|T)&F)^T) and (T|((T&F)^T)).
```

Example 2:

```
Input: N = 5
S = T^F|F
Output: 2
Explaination: ((T^F)|F) and (T^(F|F)) are the
only ways.
```

Your Task:
You do not need to read input or print anything. Your task is to complete the function **countWays()** which takes N and S as input parameters and returns number of possible ways modulo 1003.

Expected Time Complexity: $O(N^3)$

Expected Auxiliary Space: $O(N^2)$

```
class Solution:
    def countWays(self, N, S):
        # code here
```

```
        operators = []
        exps = []
        for ele in S:
            if ele in ('T','F'):
                exps.append(ele)
            else:
                operators.append(ele)
        n = len(exps)
        dpTrue = [[0]*n for _ in range(n)]
        dpFalse = [[0]*n for _ in range(n)]
        for gap in range(n):
            i = 0
            j = gap
            while j<n:
                if gap==0:
                    if exps[i] == 'T':
                        dpTrue[i][j] = 1
                        dpFalse[i][j] = 0
                    else:
                        dpTrue[i][j] = 0
                        dpFalse[i][j] = 1
                else:
                    for k in range(i,j):
                        oprt = operators[k]
                        ltc = dpTrue[i][k]
                        rtc = dpTrue[k+1][j]
                        lfc = dpFalse[i][k]
                        rfc = dpFalse[k+1][j]
                        if oprt == '&':
                            dpTrue[i][j] += ltc*rtc
                            dpFalse[i][j] += lfc*rtc + ltc*rfc + lfc*rfc
                        elif oprt == '|':
                            dpTrue[i][j] += ltc*rtc + lfc*rtc + ltc*rfc
                            dpFalse[i][j] += lfc*rfc
                        else:
                            dpTrue[i][j] += ltc*rfc + lfc*rtc
                            dpFalse[i][j] += lfc*rfc + ltc*rtc
                i = i+1
                j  = j+1
        return dpTrue[0][-1]%1003
```