Boolean Evaluation to true (similar to MCM)

expression: "T | T & F"

OR → |

AND → &

XOR → ^

Problem Statement

Suggest Edit

You are given an expression *'exp'* in the form of a string where operands will be : (TRUE or FALSE), and operators will be : (AND, OR or XOR).

Now you have to find the number of ways we can parenthesize the expression such that it will evaluate to TRUE.

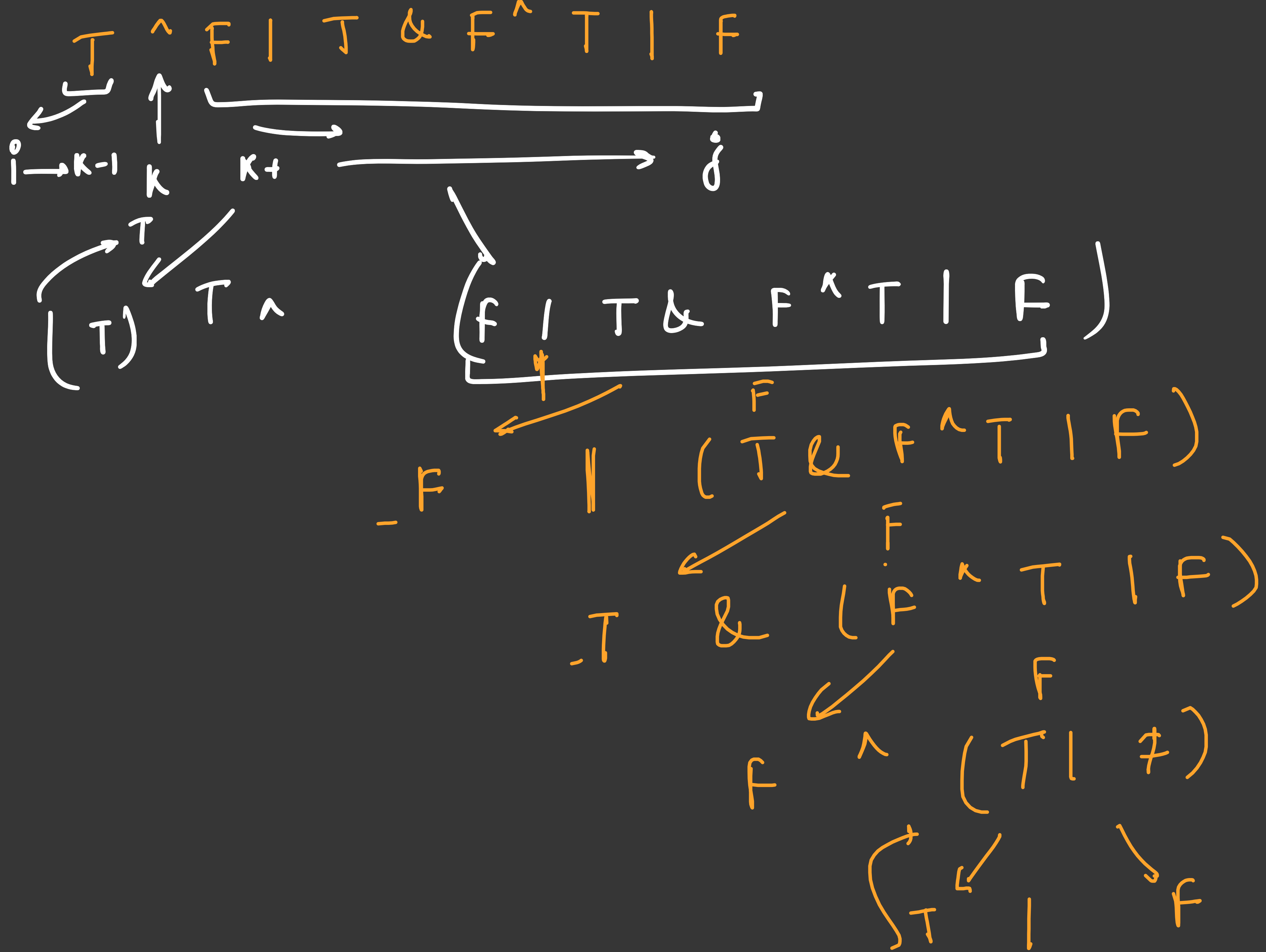
As the answer can be very large, return the output modulo 1000000007.

eg: F | T ^ F

(F | T) ^ F → T ^ F → T (1) → way

F | (T ^ F) → F | T → T (2) → way

} 2 ways



T ^ F | T & F ^ T | F
i → K-1
K
K+1
j
T
(T)
T ^
(F | T & F ^ T | F)
F
F | (T & F ^ T | F)
F
T & (F ^ T | F)
F
F ^ (T | F)
T | F

We use partition algorithm

+2 +2 +2 +2

T ^ F | T & F ^ T | F

i j

if(&) → left & right

for (k = i+1; k < j; k += 2)

&

(T)

left & right

T & y

x * y

|

left | right

T	T
F	F
x_1	x_2
x_3	x_4

| (OR)

$x_1 * x_2$

$x_1 * x_4$

$x_3 * x_2$

$x_3 * x_4$

^

left ^ right

T	T
F	F
x_1	x_3
x_2	x_4

$x_1 * x_4$

$x_2 * x_3$

T	T	T
T	F	F
F	F	F
F	T	F

T	T	T
T	F	T
F	T	T
F	F	F

T	T	→ F
T	F	→ T
F	T	→ T
F	F	→ F

Recurrence relation:

$f(0, n-1, 1)$

```
f(i, j, istrue)

if (i>j) return 0;

if (i==j) {
    if (istrue == 0) return arr[i] == 'T';
    else return arr[i] == 'F';
```

If we looking for True

if we are looking for false.

```
for (k = i+1; k < j; k += 2)
    leftT = f(i, k-1, 1)
    leftF = f(i, k-1, 0)
```

right T = f(k+1, j, 1)

right F = f(k+1, j, 0)

if(arr[k] == '&')

if(isTrue) ways += LT × rT;

else ways += (LT × rF) + (LF × rT) + (LF × rF);

else if(arr[k] == '|')

if(isTrue) ways += (LT × rT) + (LF × rT) + (LT + rF)

else ways += (LF × rF);

else{ if(isTrue) ways += (rF × LT) + (LF × rT)

else ways = (rF × LF) + (rT × LT),

return ways