Problem

- Let us define a binary operation $\otimes$ on three symbols a, b, c according to the following table; thus $a \otimes b = b$, $b \otimes a = c$, and so on. Notice that the operation defined by the table is neither associative nor commutative.

$\otimes$	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

Describe an efficient algorithm that examines a string of these symbols, say *bbbbac* , and decides whether or not it is possible to parenthesize the string in such a way that the value of the resulting expression is $p = a$. For example, on input *bbbbac* your algorithm should return *yes* because $((b \otimes (b \otimes b)) \otimes (b \otimes a)) \otimes c = a$.

Solution

- For $1 \leq i \leq j \leq n$, we let $T[i, j] \subseteq \{a, b, c\}$ denote the set of symbols e for which there is a parenthesization of $x_i..x_j$ yielding e . We let $e \otimes b$ denote the product of e and b using the table. $p \in \{a, b, c\}$ is the symbol we are considering.
- Pseudocode

```
for i ← 1 to n do
    T[i, i] ← x_i
for s ← 1 to n-1 do
    for i ← 1 to n-s do
        T[i,i + s] ← ∅
        for k ← i to i+s-1 do
            for each e ∈ T[i,k] do
                for each b ∈ T[k + 1,i + s] do
                    T[i,i+s] ← T[i,i+s] ∪e ⊗ b
if p ∈ T[1,n] then
    return yes
    else
        return no
```