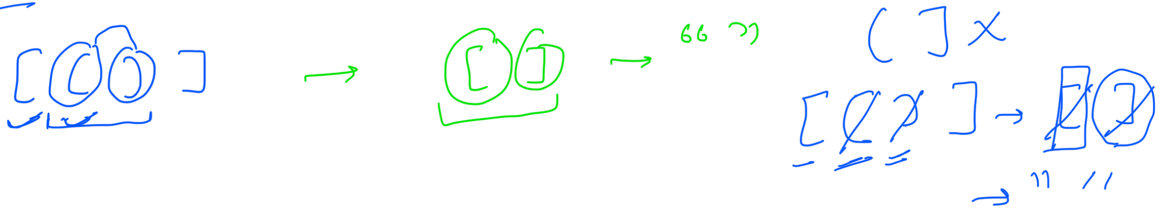
(5) → Valid parenthesis -

1. Question -

Given a string ((, [, {,),], }) determine if the string is valid (or) not.

Valid if -

- Open brackets must be closed by the same type of brackets ( ) { } [ ]
- Open brackets must be closed in the correct order.



2. Examples -

S = ( ) ✓

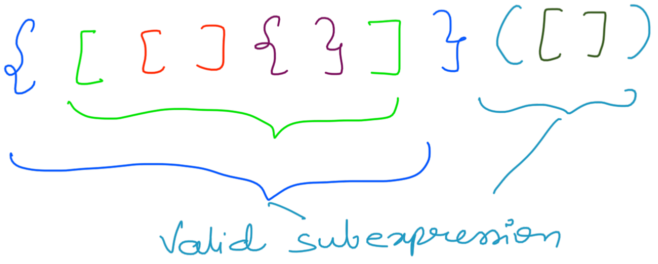
→ true

S = ( ) [ ] { } " "

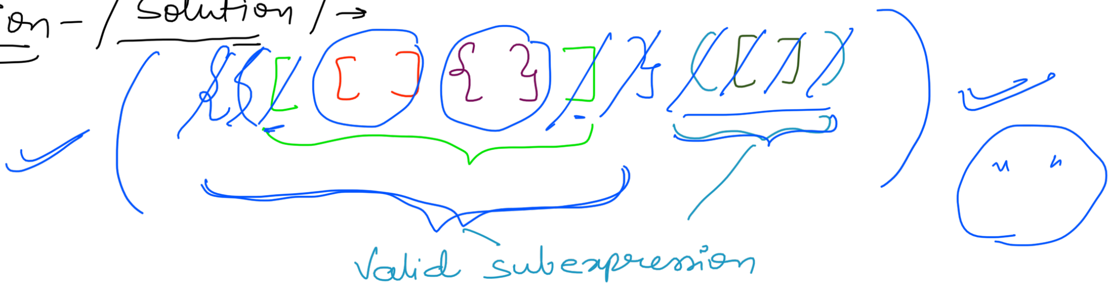
→ true

S = ( ]

→ false



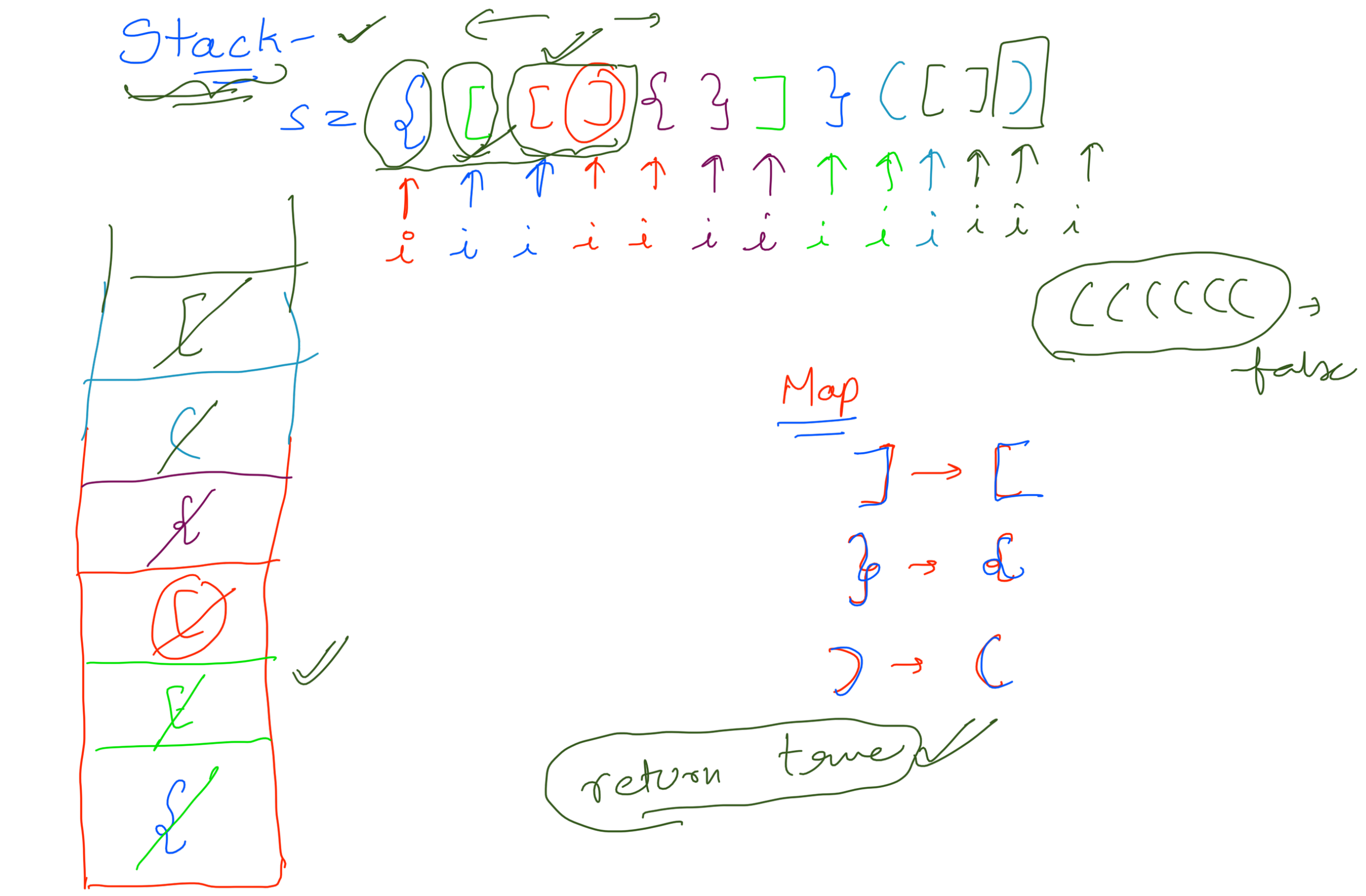
3. Intuition - / Solution / →



- will be one big valid expression
-(or) will be multiple valid sub-expressions

If we remove all valid sub-expressions, we are left with an empty string. ✓

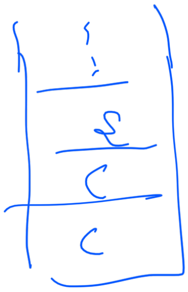
This holds recursive structure ...



4. Time complexity

Time = O(n) → n

Space = O(n) / Stack size → n ?



}

([{([{([{...

5. Additional info-

What if there is only one type of bracket?

Do we need stack?

()(((())()))

T = O(n)

S = O(1)

( )

)

countOpP ↓

((?

(

countOpP ↑

== 0 &&)

6. Code walk through-

map

c	o
]	[
}	{
)	[

```
boolean isValid (s) {
    Stack<Character> stack...
    for (Character bracket : s) {
        if (map.containsKey(bracket)) {
            Character top = stack.isEmpty() ?
                'x' : stack.peek();
            if (top != map.get(bracket)) {
                return false;
            }
            stack.pop();
        } else {
            stack.push(bracket);
        }
    }
    return stack.isEmpty();
}
```

o B

true : false

({ })
(