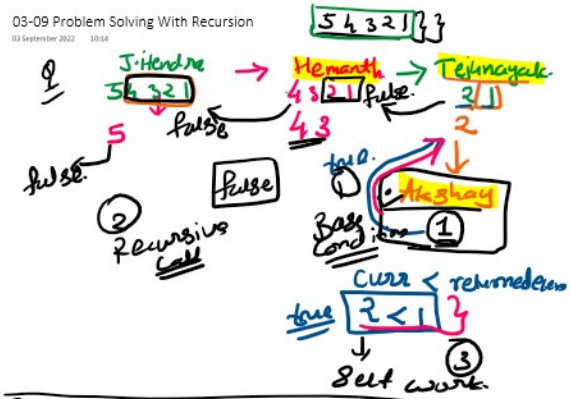
03-09 Problem Solving With Recursion

03 September 2022 10:14



Q Generate Parenthesis recursively:-

I/P Count of parenthesis = 2

O/P ( ) ( ) (( ))

2 * n

2 * 2

Count = 3

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

2 * 3 = 6

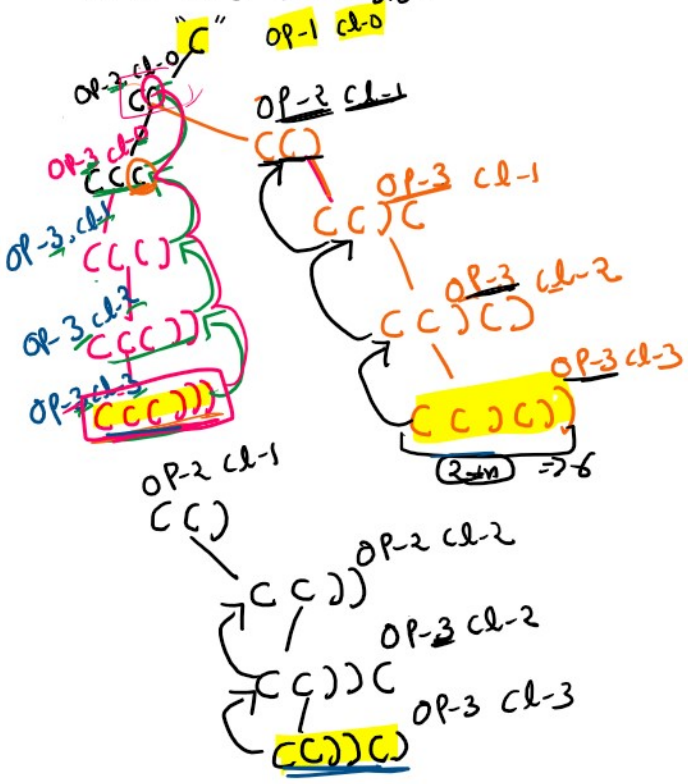
(( )) ( ) (( )) ( )

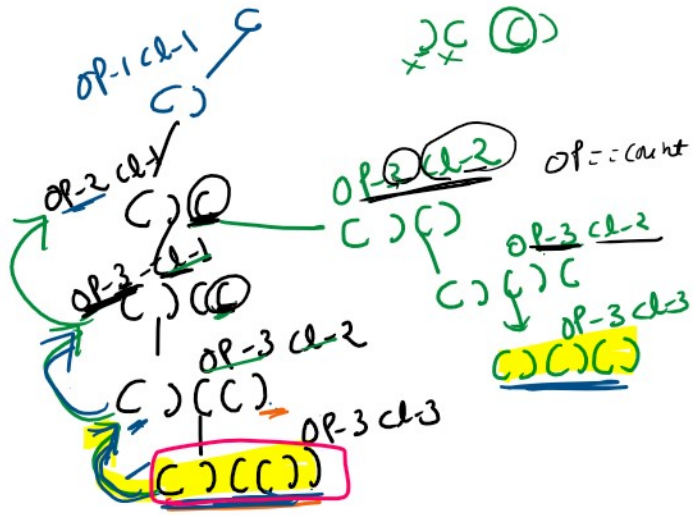
(( )) ( ) (( )) .

1. If we are said, for n = 3 => ideal no. of total chars => 2 * n => 2 * 3 => 6
2. no. of opening brackets == no. of closing brackets
3. closing bracket can't come before opening bracket
4. count of opening brackets can't be more than n.

Count -> 3

No. of valid Parenthesis:-





function PrintValidParenthesis (result, open, close, n)

{ Base case ['(', ')', '(', ')'] (())

```
if(result.length == 2*n)
{
   console.log(result.join(''));
   return;
}
```

```
if(open < n)
{
   result.push('(');
   PrintValidParenthesis(____);  // fn call
   result.pop();  // self work
}
if(close < open)
{
   result.push(')')
   PrintValidParenthesis(____);  // fn calls
   result.pop();                 // self work.
}
```

Q:- Lexicographical order:-

.sort()

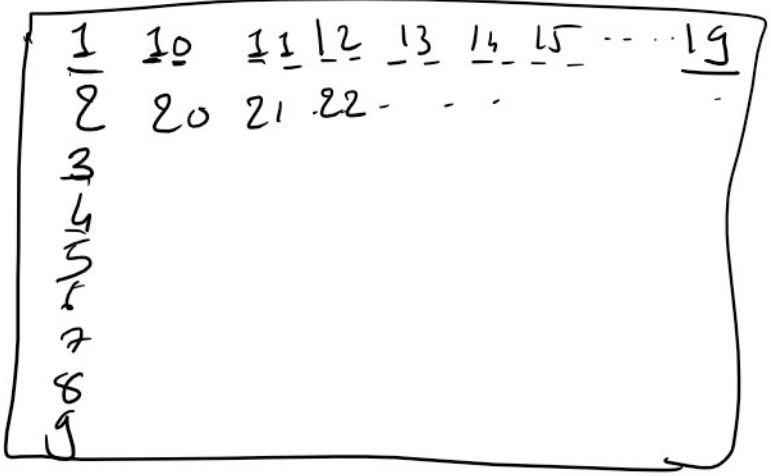
[1 2 3 11 21 4] ⇒

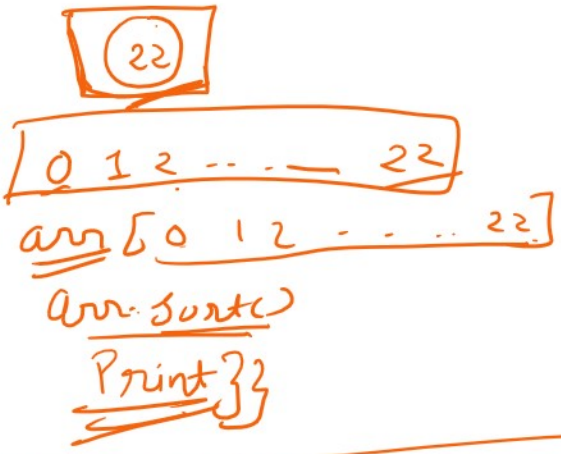
Sort()

1 2 3 4 11 21

```
arr.sort()
▶ (7) [1, 11, 2, 21, 3, 4, 5]
```

n = 22



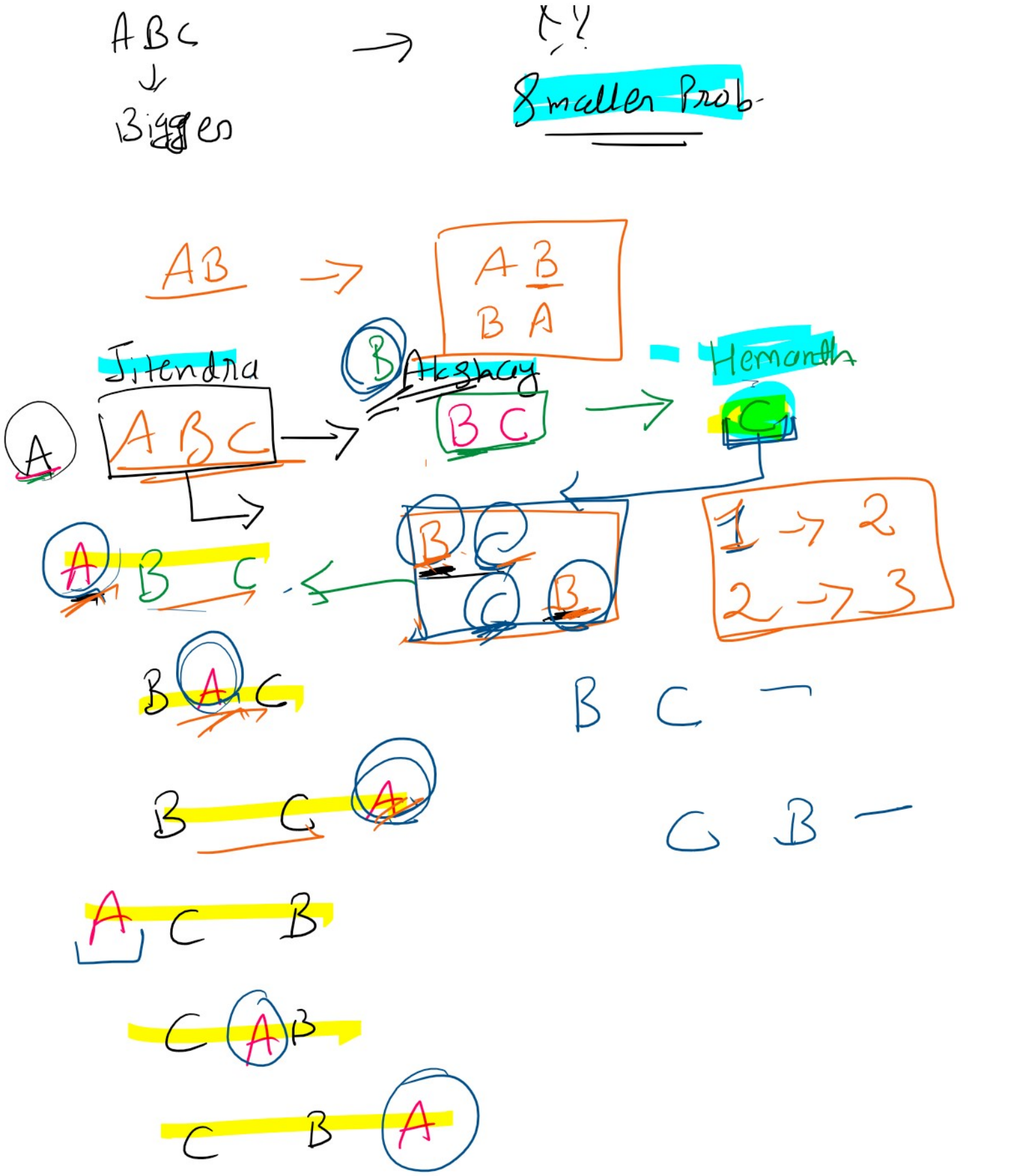


Q

Input = ABC

ABC
ACB
BAC
CAB
BCA
CBA

ABC → ??


ABC
Biggies
Smaller Prob.
AB
A B
B A
Jitendra
ABC
Akshay
BC
Hemanth
C
1 → 2
2 → 3
A
B C
B A C
B C A
A C B
C A B
C B A
B C
C B

n → (+1)

Code:-

(str, index) → ("abc", 0)

Bigger problem

Smaller Prob ("abc", 1)

1. Base case

if (index == str.length-1)

{

let br = [ ]

br.push([str.charAt(index)])

return br; // c

}

2. Recursion

"abc" → "bc", "cb" → 2

let res = Permutations (str, index+1)

let mr = [ ]

for(let i=0; i < res.length; i++) (+1)

{

let val = res[i] // "bc" 2

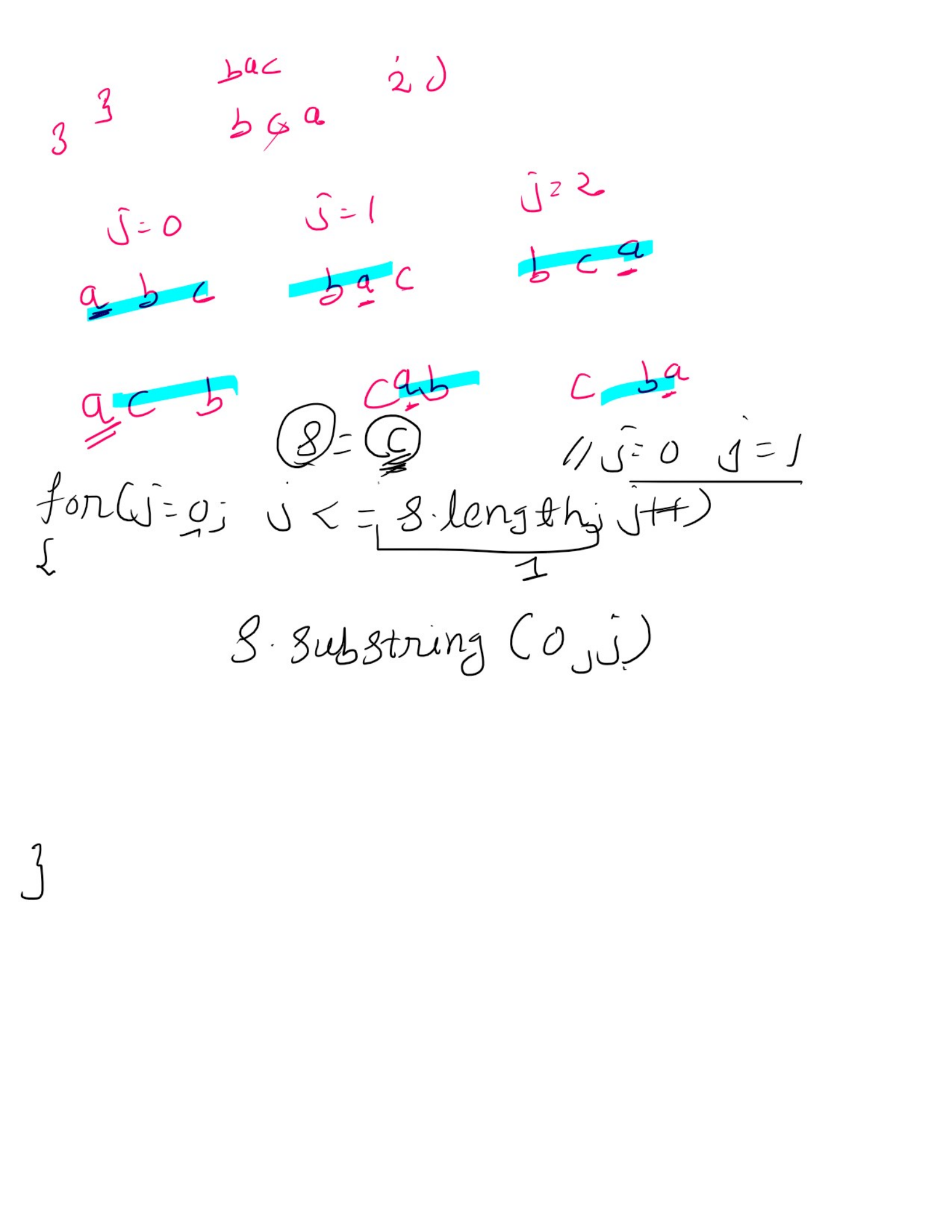
for(j=0; j<=val[0].length; j++)

a b c

b a c

0, 1, 2 } 3 times

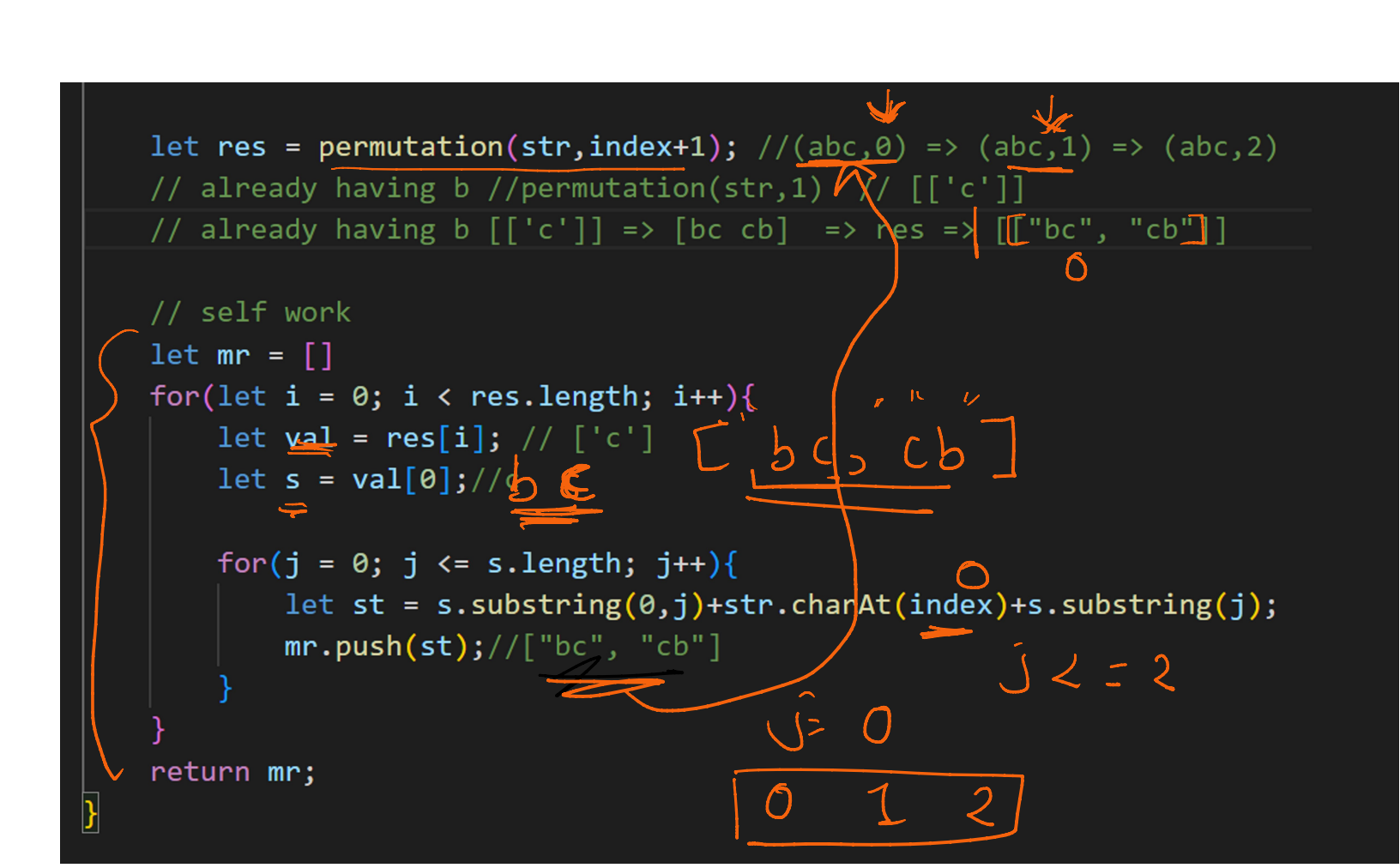
}


bac
2 0
3 3
b & a
j = 0
j = 1
j = 2
a b c
b a c
b c a
a c b
c a b
c b a
s = c
// j = 0 j = 1
for(j = 0; j <= s.length-1; j++)
{
s.substring(0, j);
}

```
    let res = permutation(str,index+1); //(abc,0) => (abc,1) => (abc,2)
    // already having b //permutation(str,1) // [['c']]
    // already having b [['c']] => [bc cb]  => res => [["bc", "cb"]]

    // self work
    let mr = []
    for(let i = 0; i < res.length; i++){
        let val = res[i]; // ['c']
        let s = val[0];//c

        for(j = 0; j <= s.length; j++){
            let st = s.substring(0,j)+str.charAt(index)+s.substring(j);
            mr.push(st);//["bc", "cb"]
        }
    }
    return mr;
}
```



j = 0 j = 1 j = 2