SCALER

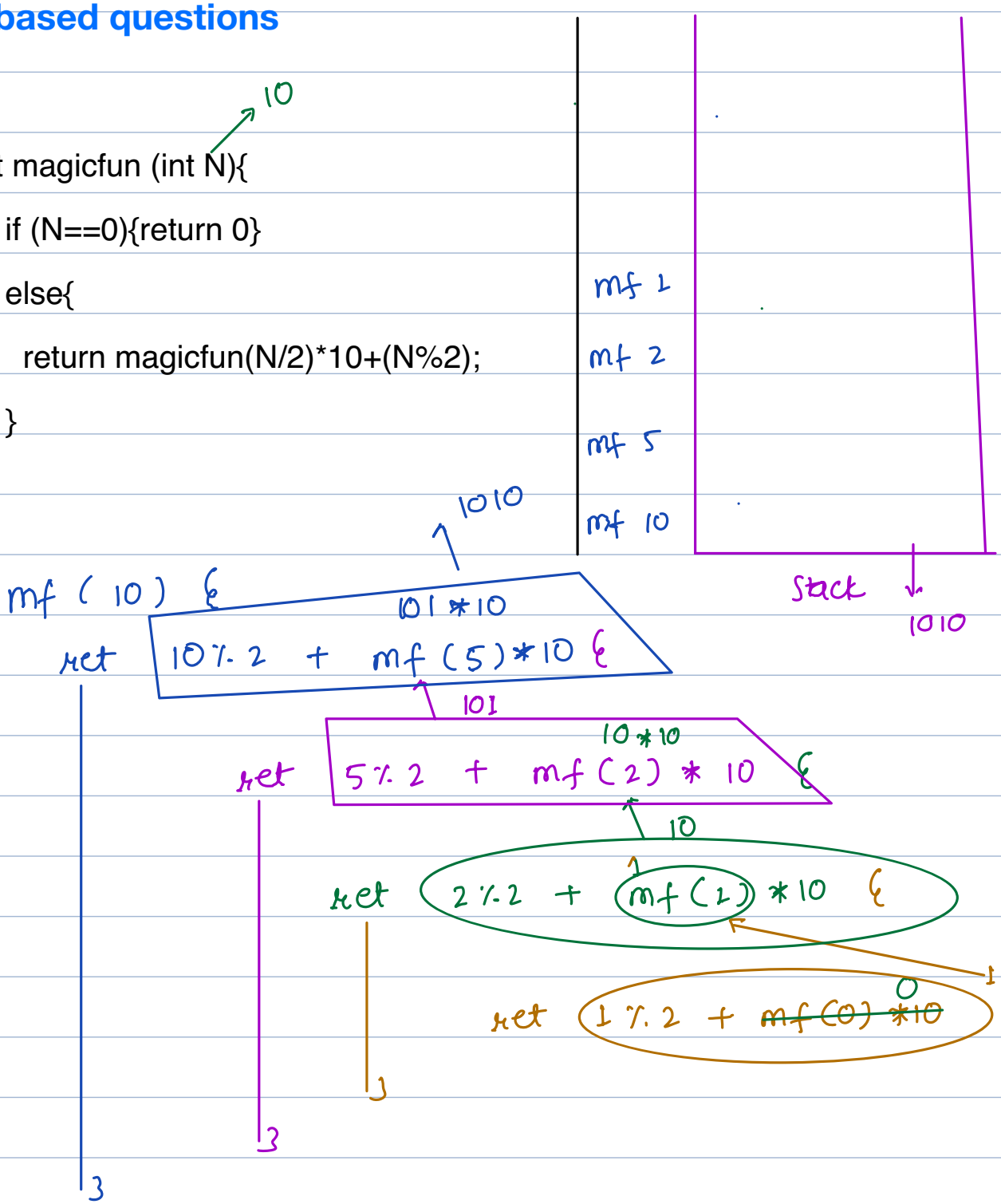
Backtracking

TABLE OF CONTENTS

Output based questions

1.

```
int magicfun (int N){
    if (N==0){return 0}
    else{
      return magicfun(N/2)*10+(N%2);
    }
}
```



```
void fun (int N) {
    if (N == 0) return
    print(N)      ①
    fun(N-1)      ②
    print(N)      ③
}
```

output

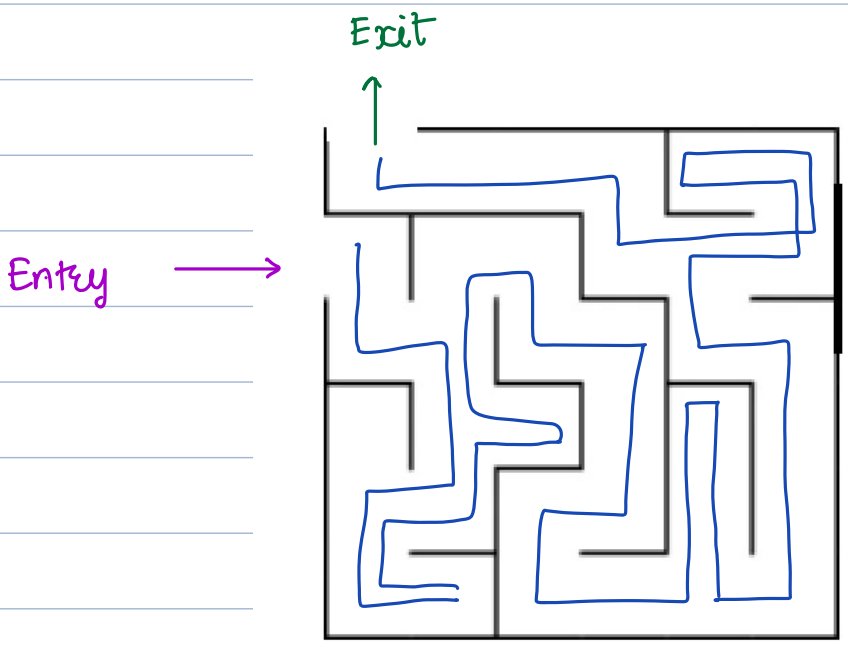
2
1
1
2

```
fun(2) {
    print(2)      ①
    fun(1) {
        print(1)      ①
        fun(0)        ②
        print(1)      ③
    }
    print(2)      ③
}
```

Backtracking

Trying out all possibilities using recursion

healthy pizza chocho



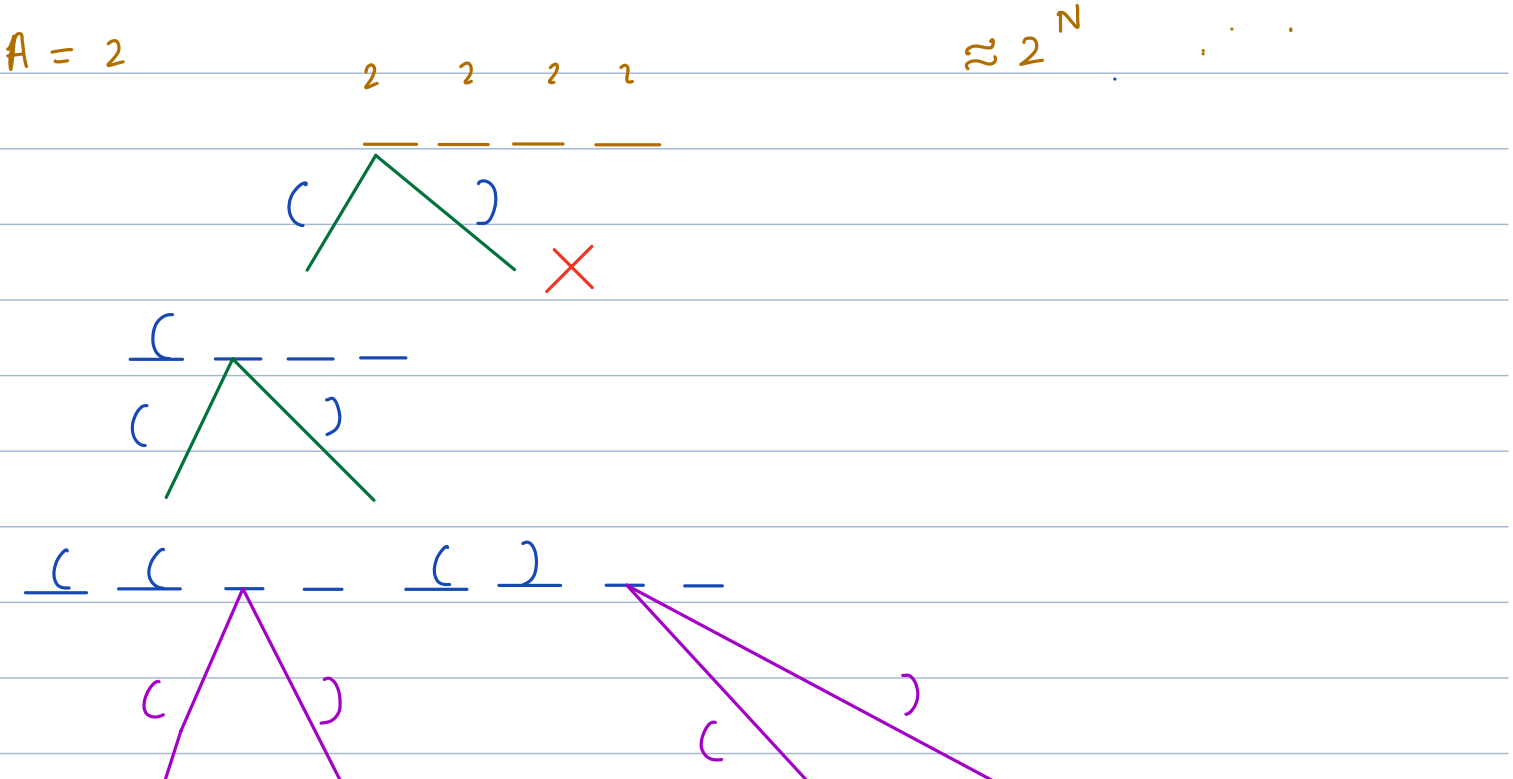
Q 1> Print valid Parenthesis

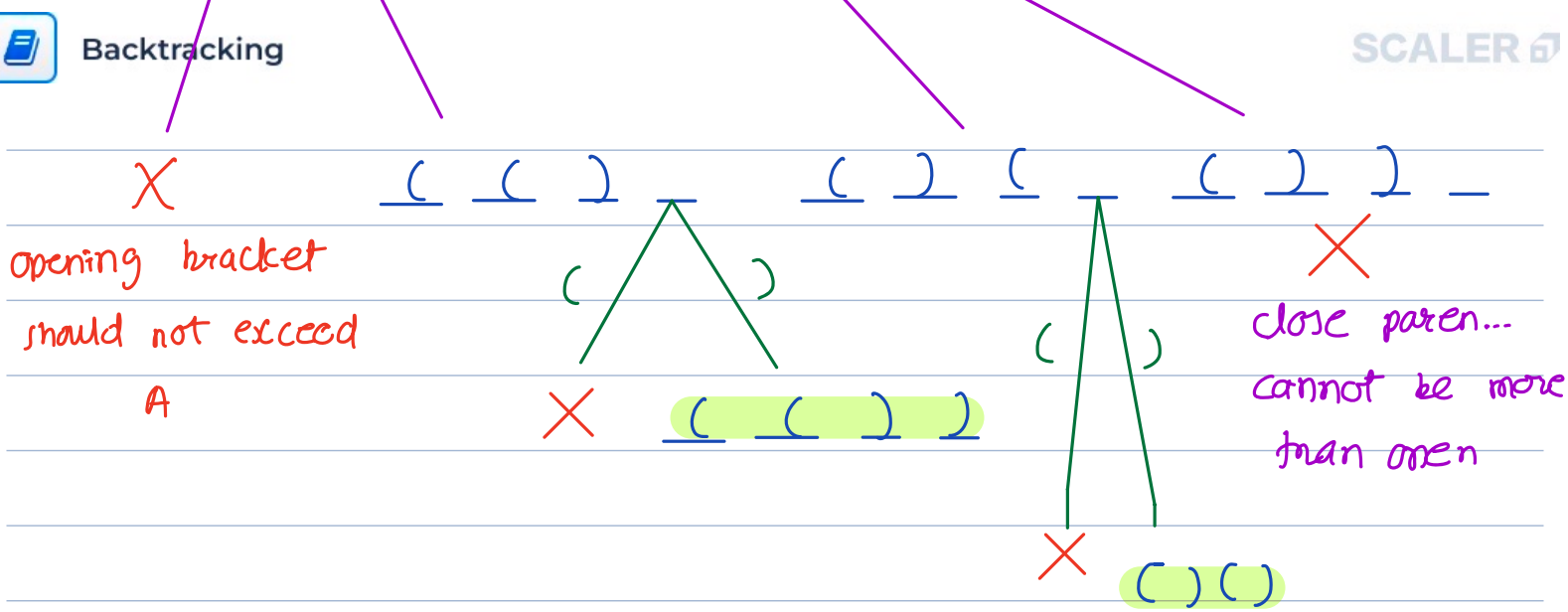
Given an int A.

write a function to generate all combinations of well formed parentheses of length 2A ( )

Input	Output
A = 1	["()"]
A = 2	["()()", "(())"]
A = 3	["()()()", "((()))", "()(())", "(())()", "(()())"]

A = 2





Pseudo code

```
A global
ans global

void generateVP ( S , open , close ) {
    // Base cond"
    if ( open == A && close == A ) {          // S.length == 2A
        ans.add(s) ; return ;
    }
    // use open bracket
    if (open < A) {
        generateVP( s + "(" , open+1, close )
    }
    // use close bracket
    if ( close < open) {
        generateVP ( s + ")" , open , close +1)
    }
}
```

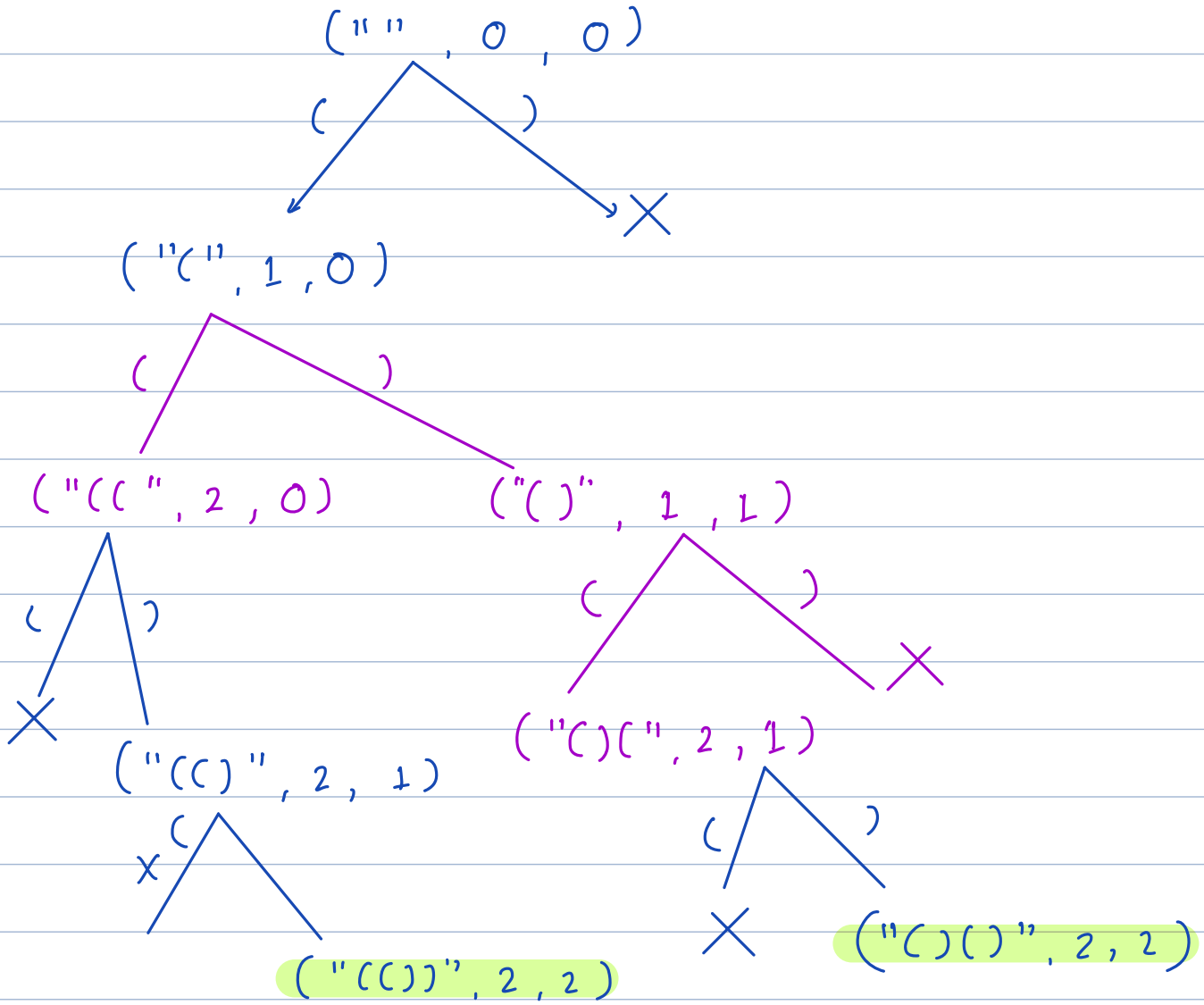
```
main (A) {
    this.A = A
    this.ans = new ArrayList<>()
    generateVP("", 0, 0)
    return ans
}
```

TC: $O(n * 2^n)$

SC: $O(n)$

Dry Run

A = 2



No. of fn call * TC per call

2^n * $O(N)$

TC: $O(n * 2^n)$

Definition of subset and subsequence

[2 8 12 5 3 16]

subset ⟶ Any selection of elements of an array in any order

[] ✓

[8 2] ✓ == [2 8] ✓

[12 3 16] ✓

subsequence ⟶ It is a subset with order

[3 5] ✗

[2 12 5] ✓

[12 5 8] ✗

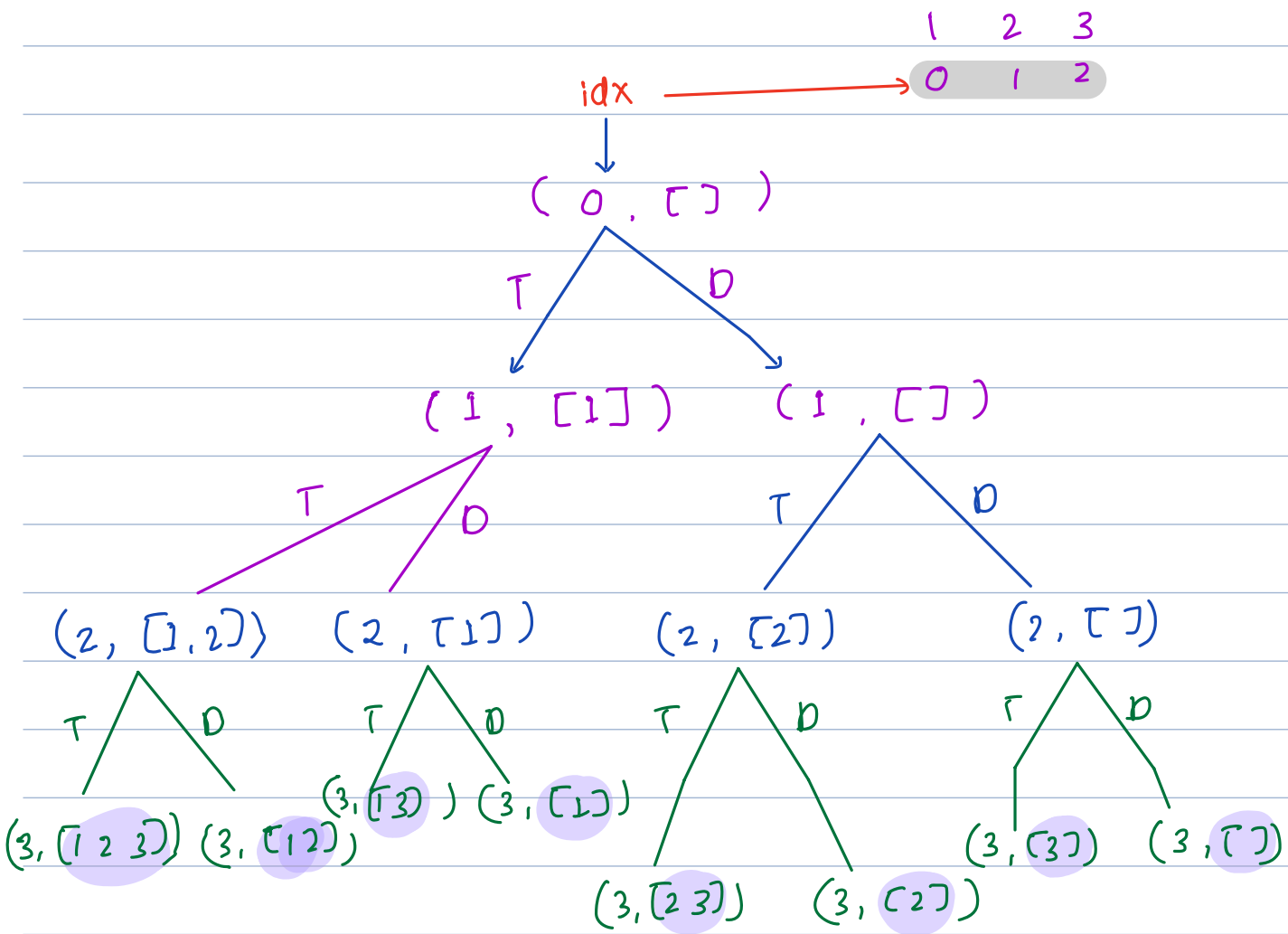
2. Generate all subsets of the given arr[ ] { distinct }

output

A[] = [5 8] [] [5] [8] [5, 8]

A[] = [1 2 3] [] [1 3]
[1] [2 3]
[2] [1 2]
[3] [1 2 3]

subsets for A.length == n = 2^n



Pseudo code

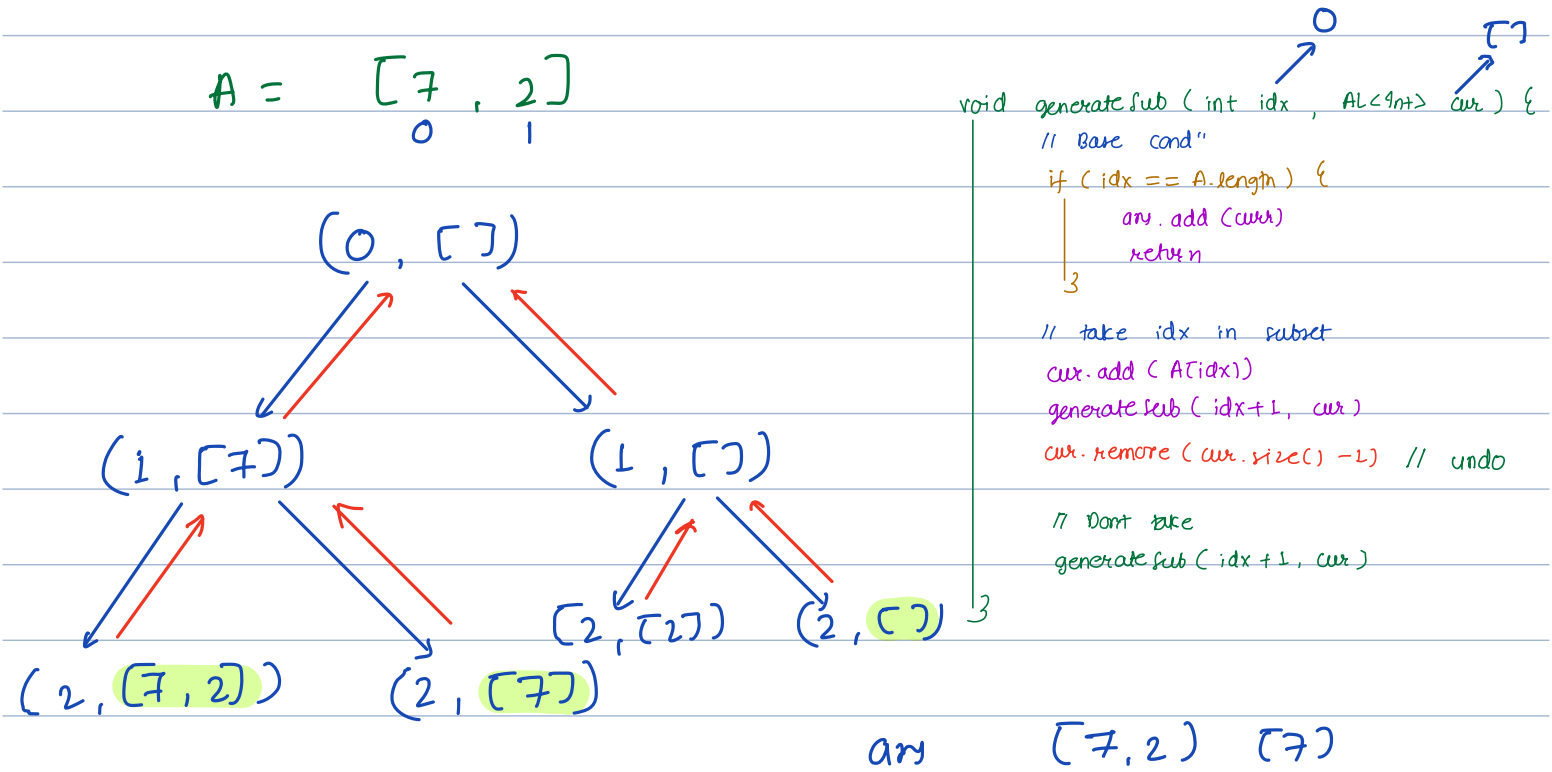
```
A[] global
ans global
void generateSub ( int idx , AL<Int> cur ) {
    // Base cond"
    if ( idx == A.length ) {
        ans.add (curr)        // shallow copy
        return
    }
    ans.add ( new ArrayList<>(curr))
                              deep copy
    // take idx in subset
    cur.add ( A[idx])         //  do
    generateSub ( idx+1, cur )
    cur.remove ( cur.size() -1)  // undo

    // Dont take
    generateSub ( idx + 1, cur )
}
```

There are 2 issues in the above code ?

TC: $O(n * 2^n)$

SC: $O(N)$

Break : 10:46



Template Backtracking

generate all possibilities via recursion

DO

recurse

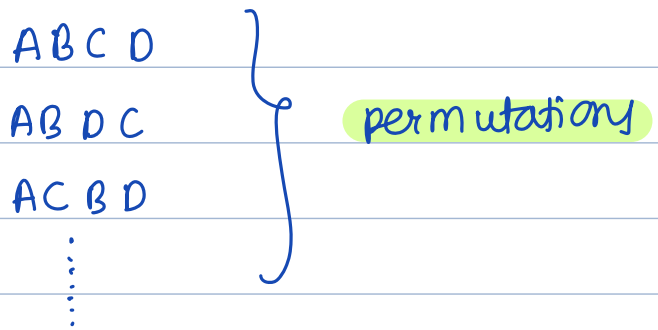
UNDO

Fitness Meets Variety

A popular Fitness app FitBit, is looking to make workouts more exciting for its users. The app has noticed that people get bored when the same exercises are shown in the same order every time they work out. To mix things up, FitBit wants to show all the different ways the exercises can be arranged so that each workout feels new.

Your challenge is to write a program for FitBit that takes a string A as input, where each character in the string represents a different exercise. Your program should then find and display all possible arrangements of these exercises.

Example:
A = Push-ups
B = Squats
C = Burpees
D = Planks



Permutations

Given a string with distinct elements.
Print all permutations of it without modifying input

Eg

Input

"abc"

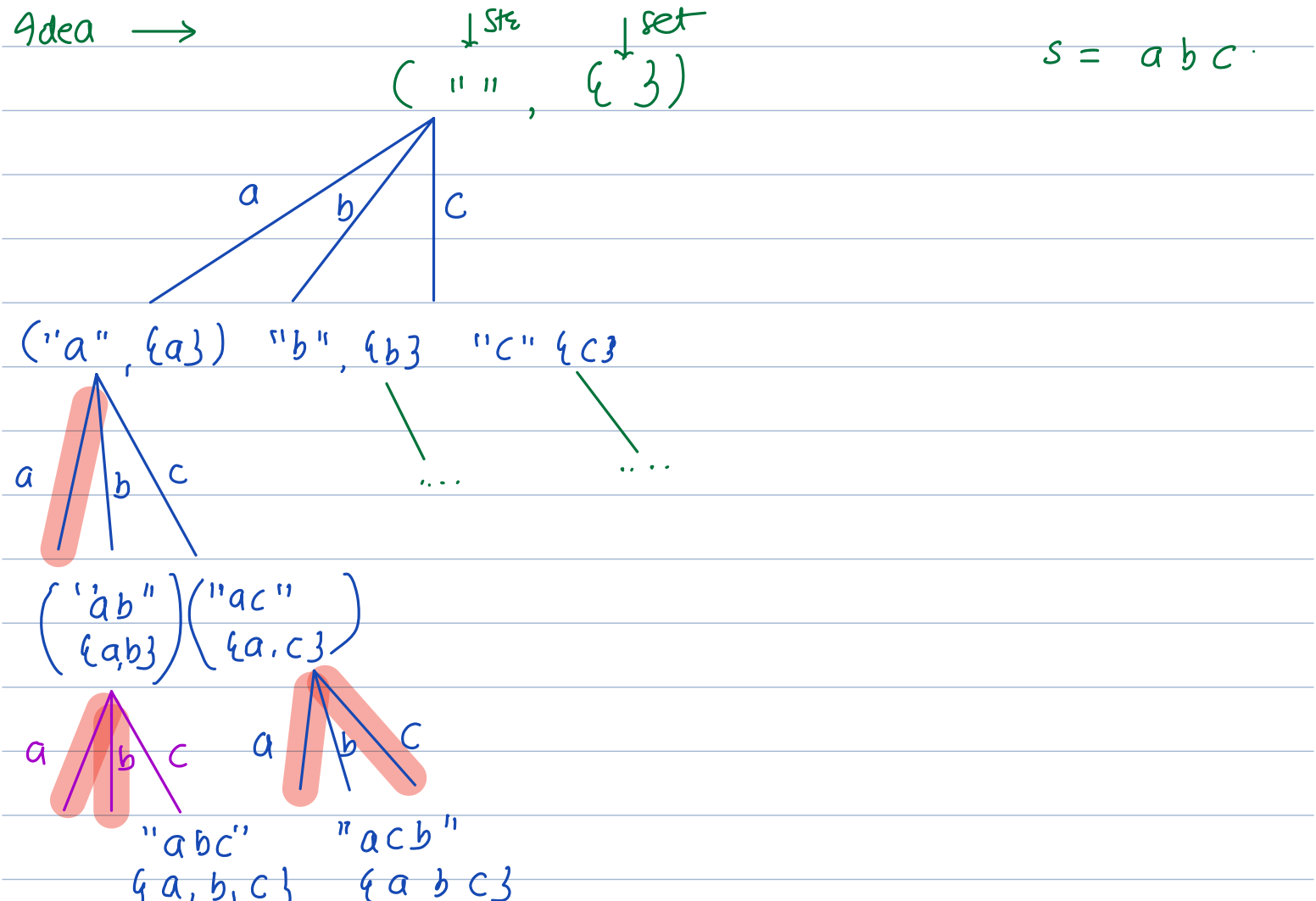
Output

abc
acb
bac
bca
cab
cba

$3! = 6$

We use set to maintain characters used.

Idea →



s = abc

S is global

```
void generatePerm(String cur, HashSet set) {
    // Base cond"
    if (cur.length() == S.length) {
        print(cur)
        ret
    }

    for i ⟶ 0 to S.length - 1 {
        ch = S.charAt(i)
        if (! set.contains(ch)) {
            // DO
            set.add(ch)
            generatePerm(cur + ch, set)
            // Undo
            set.remove(ch)
        }
    }
}
```

""

{3

equivalent

TC: O(n * n!) ⟶ O(n!)

SC: O(n)

Doubt session

A[] = 7, 8
0 1

DT
~
T
x