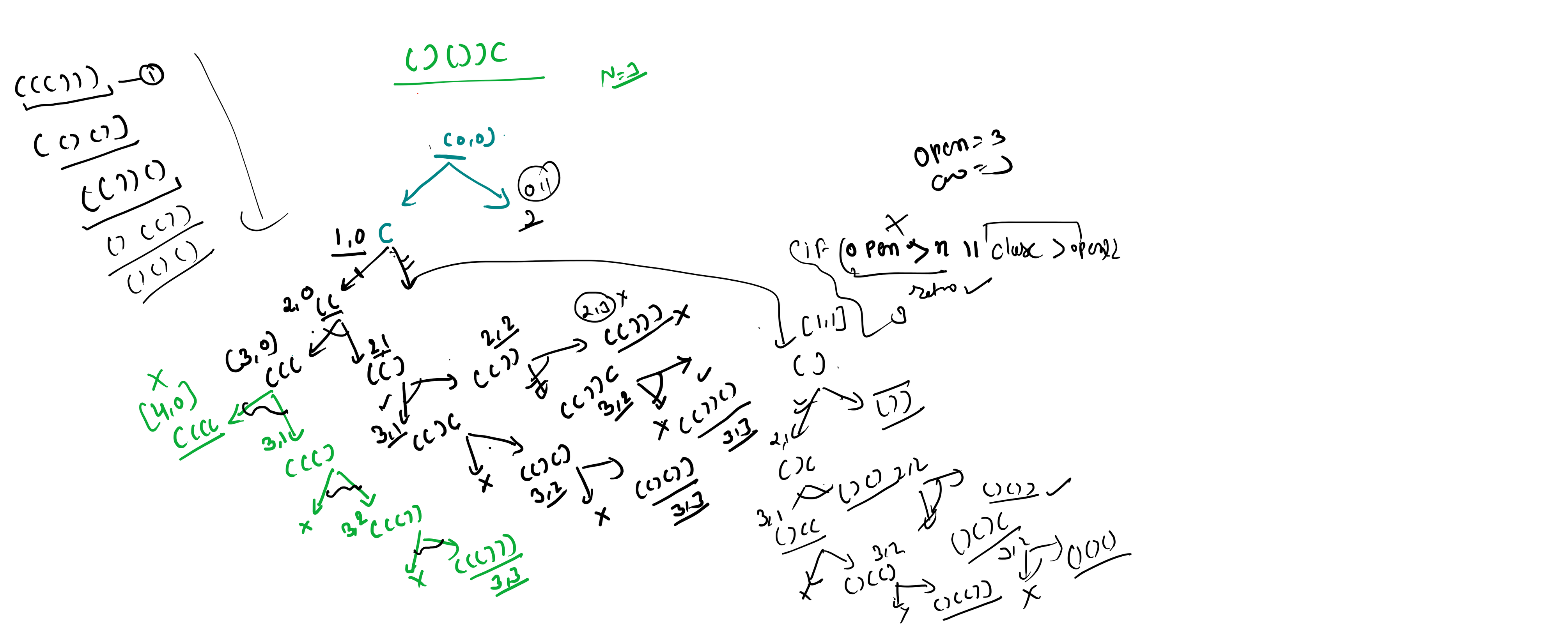
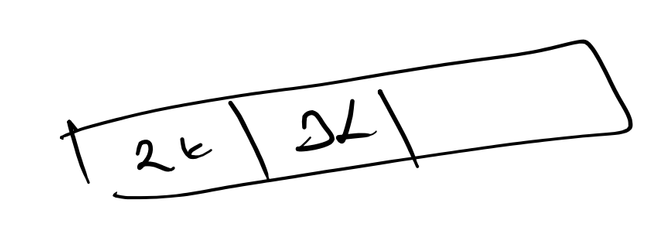
Given n pairs of parentheses, write a function to *generate all combinations of well-formed parentheses*.

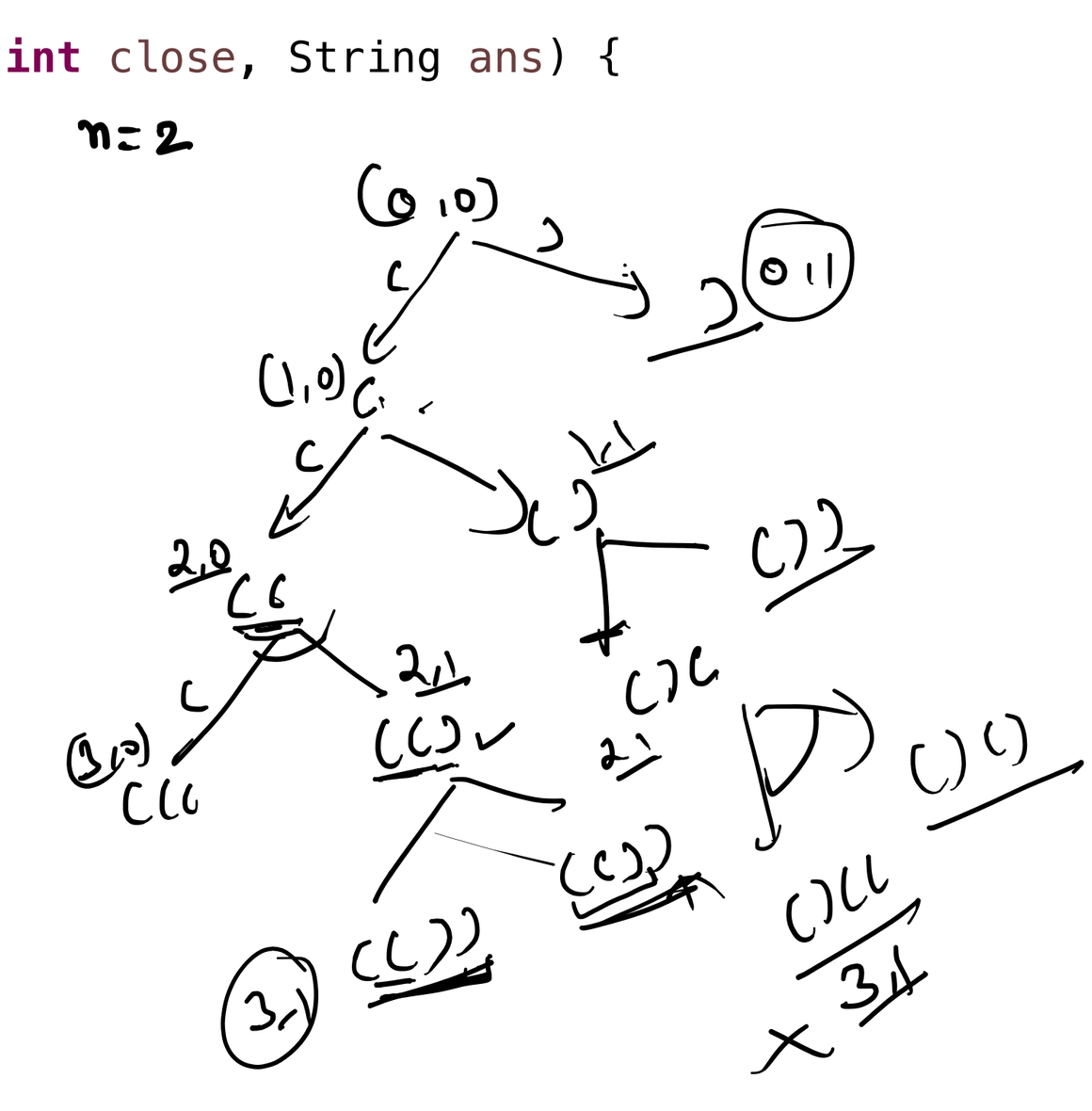
Example 1:

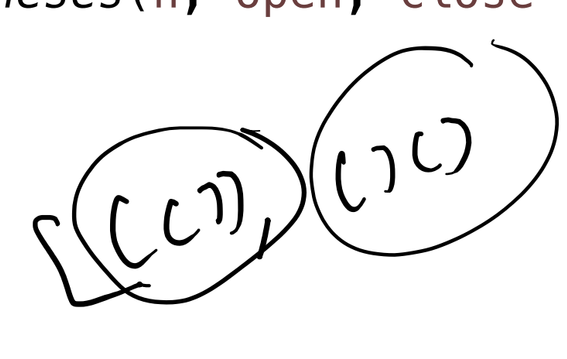
```
Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]
```



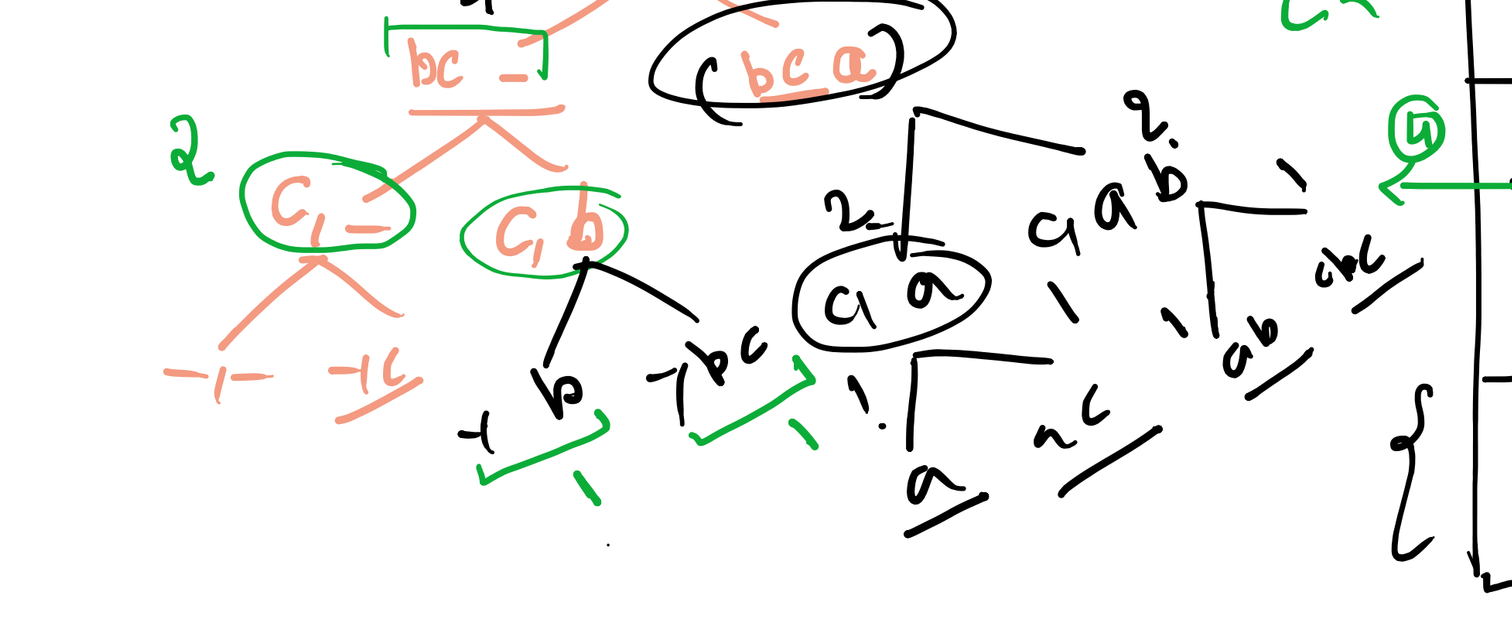


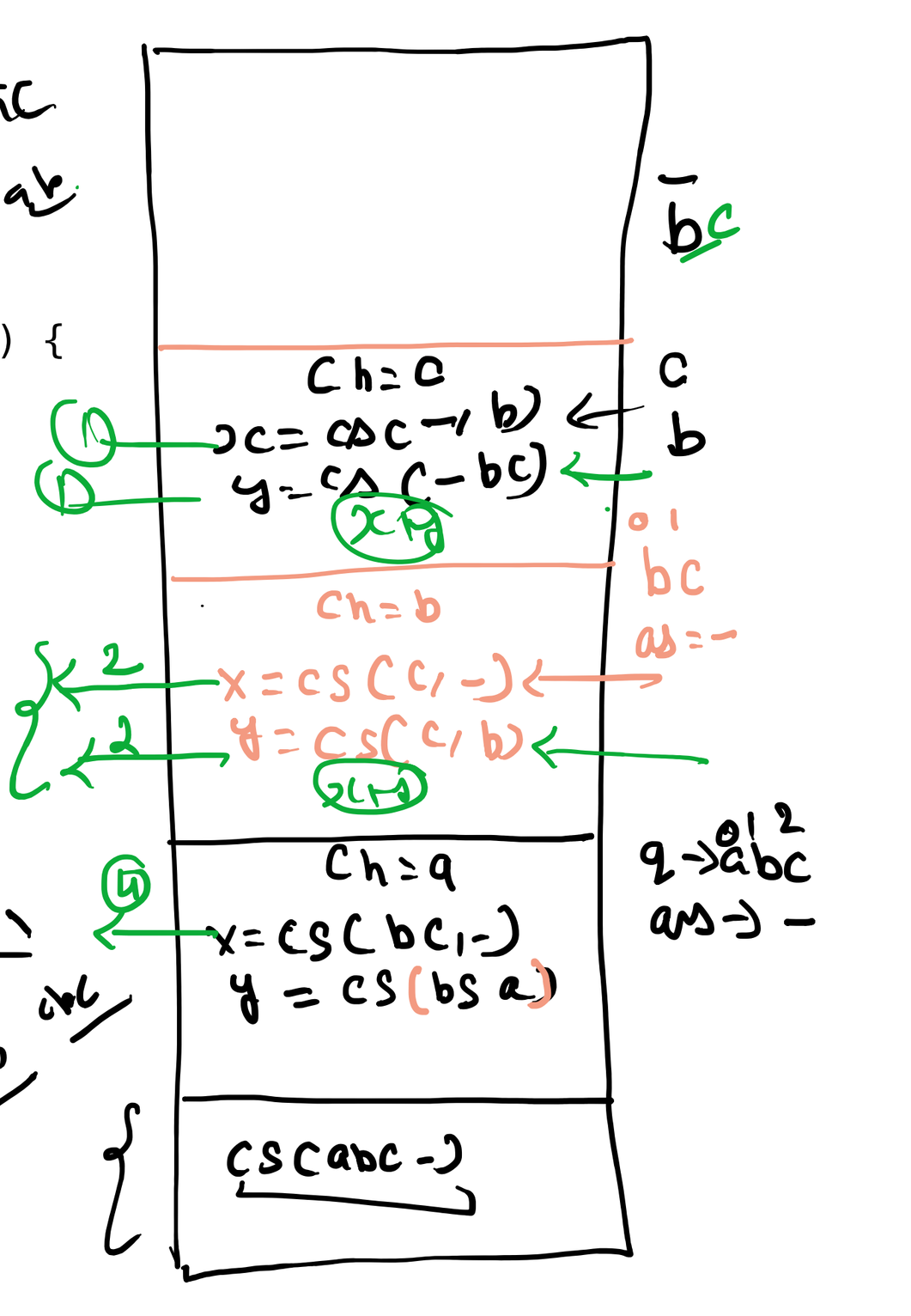
```java
public static void Parentheses(int n, int open, int close, String ans) {
    if (open == n && close == n) {
        System.out.print(ans + " ");
        return;
    }
    if (open > n || close > open) {
        return;
    }
    Parentheses(n, open + 1, close, ans + "(");
    Parentheses(n, open, close + 1, ans + ")");

}
```

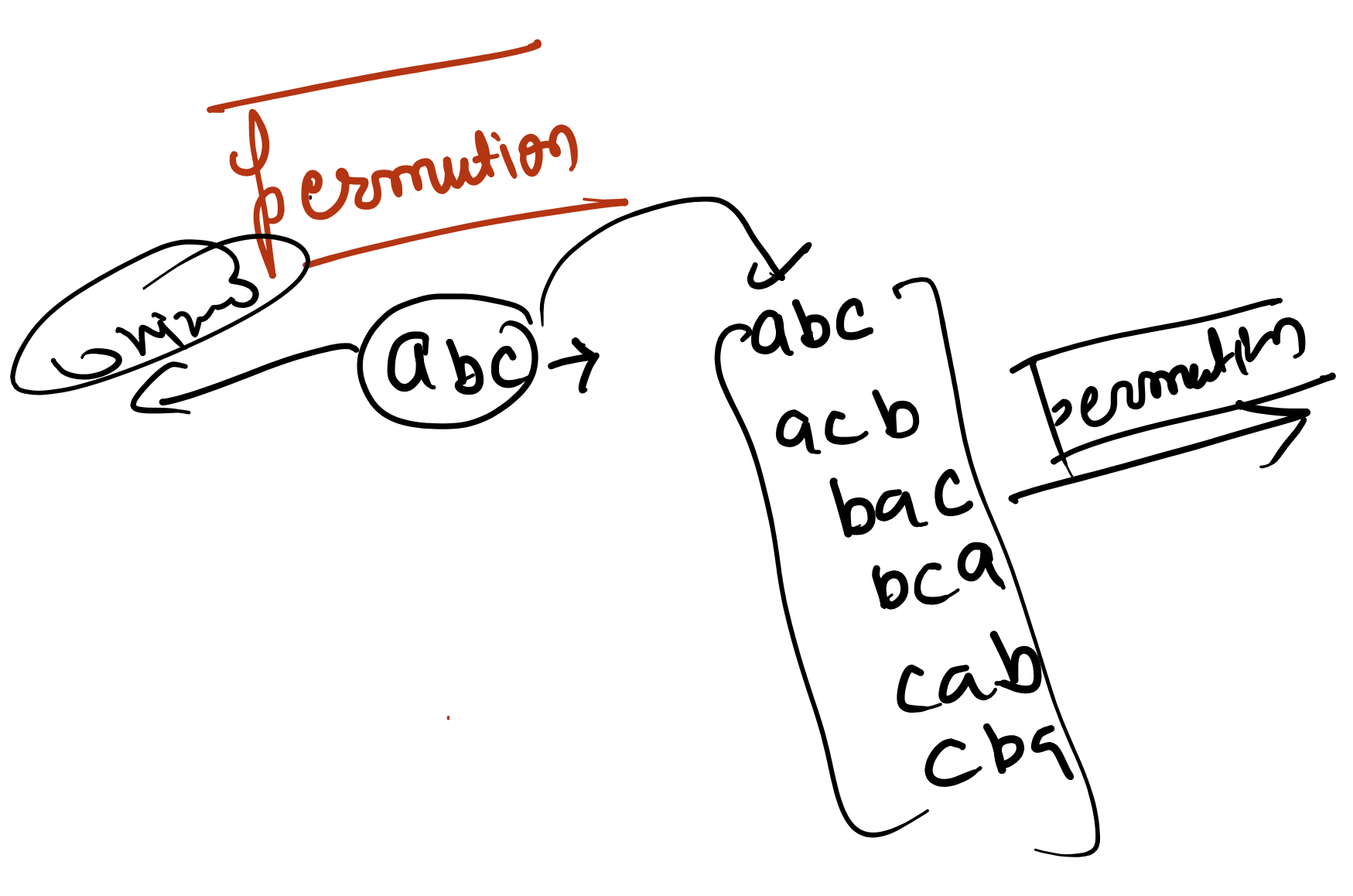


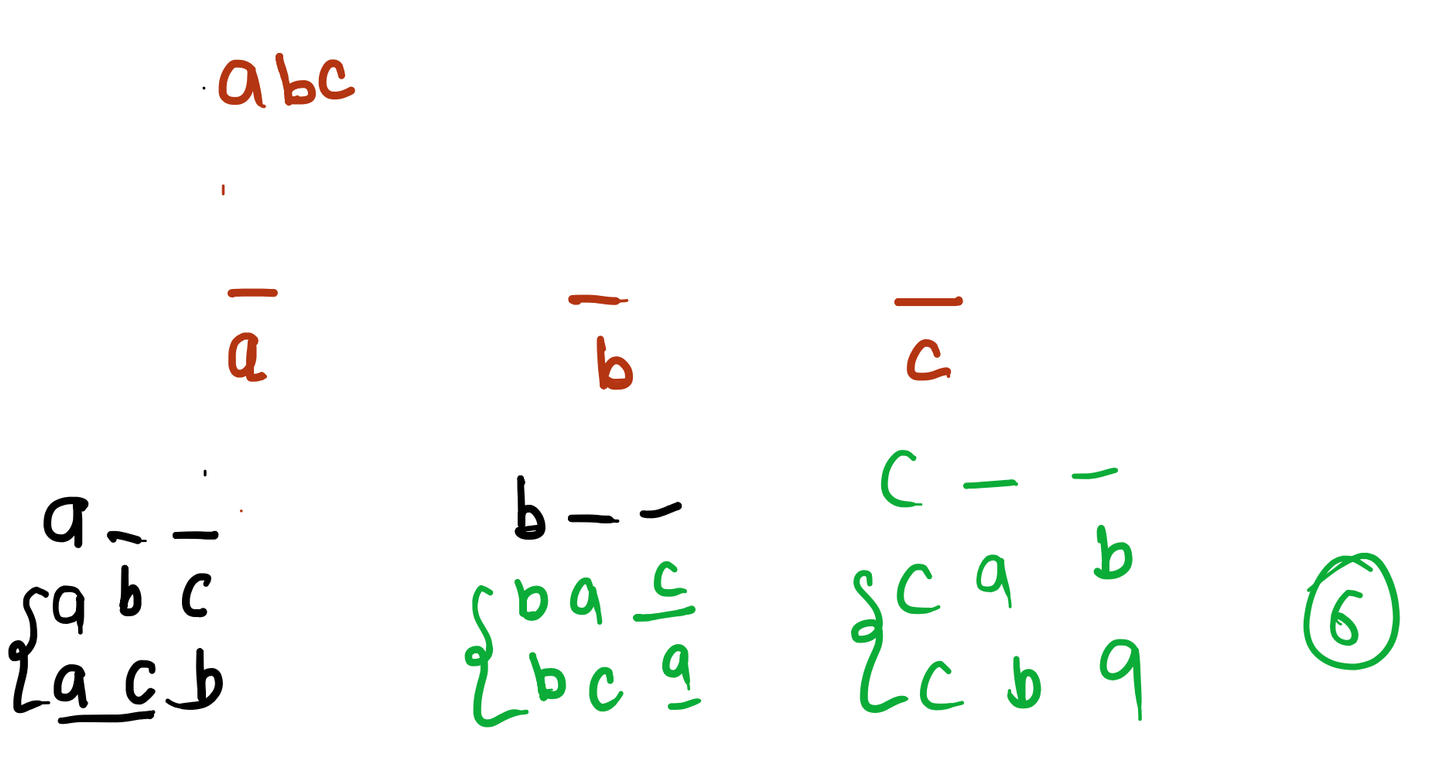


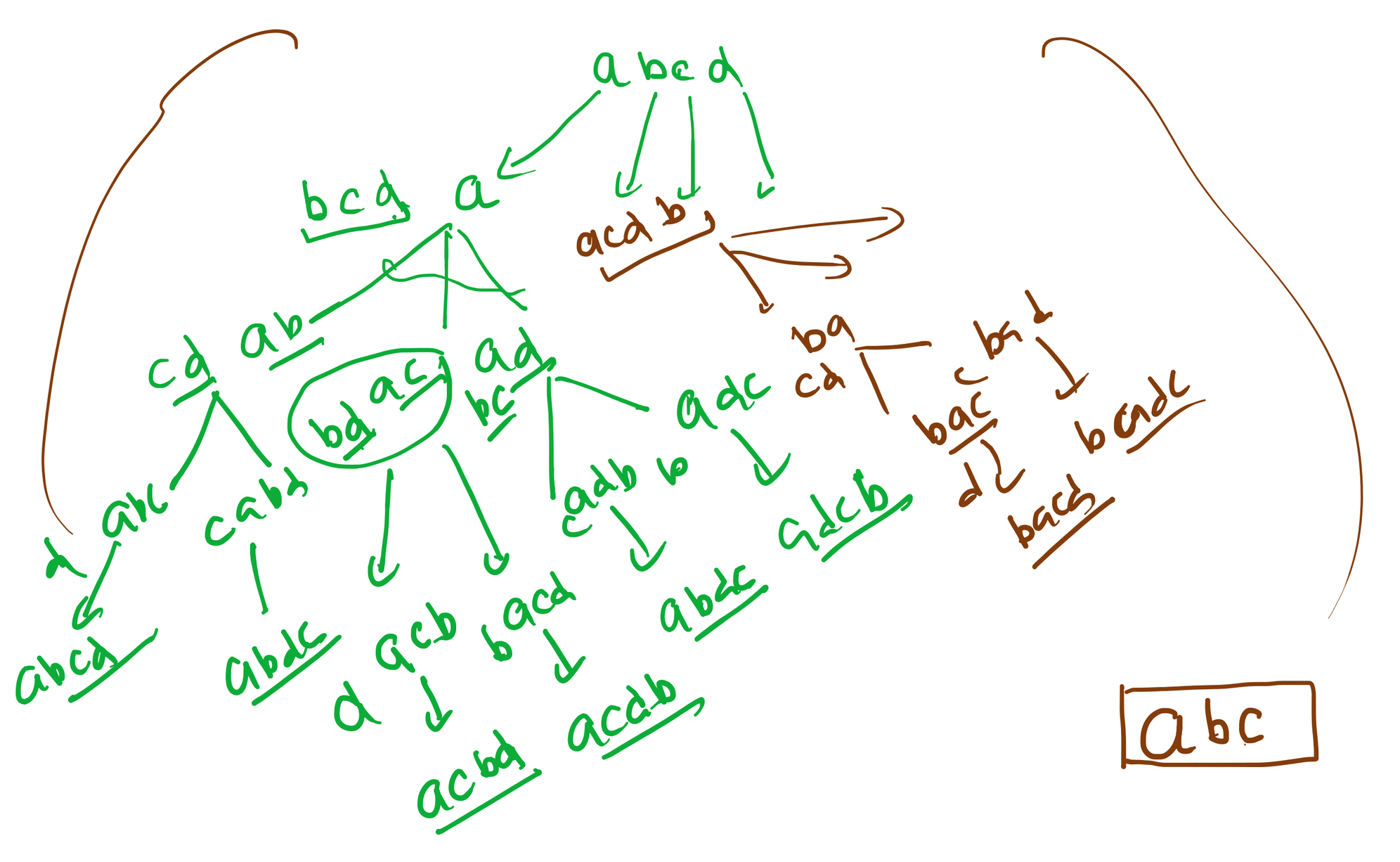
```java
public static void main(String[] args) {
        // TODO Auto-generated method stub
        String s = "abc";
        System.out.println("\n"+CountSubsence(s, ""));
    }

    public static int CountSubsence(String ques, String ans) {
        if (ques.length() == 0) {
            System.out.print(ans + " ");
            return 1;
        }
        char ch = ques.charAt(0);
        int x = CountSubsence(ques.substring(1), ans);
        int y = CountSubsence(ques.substring(1), ans + ch);
        return x + y;

    }
```

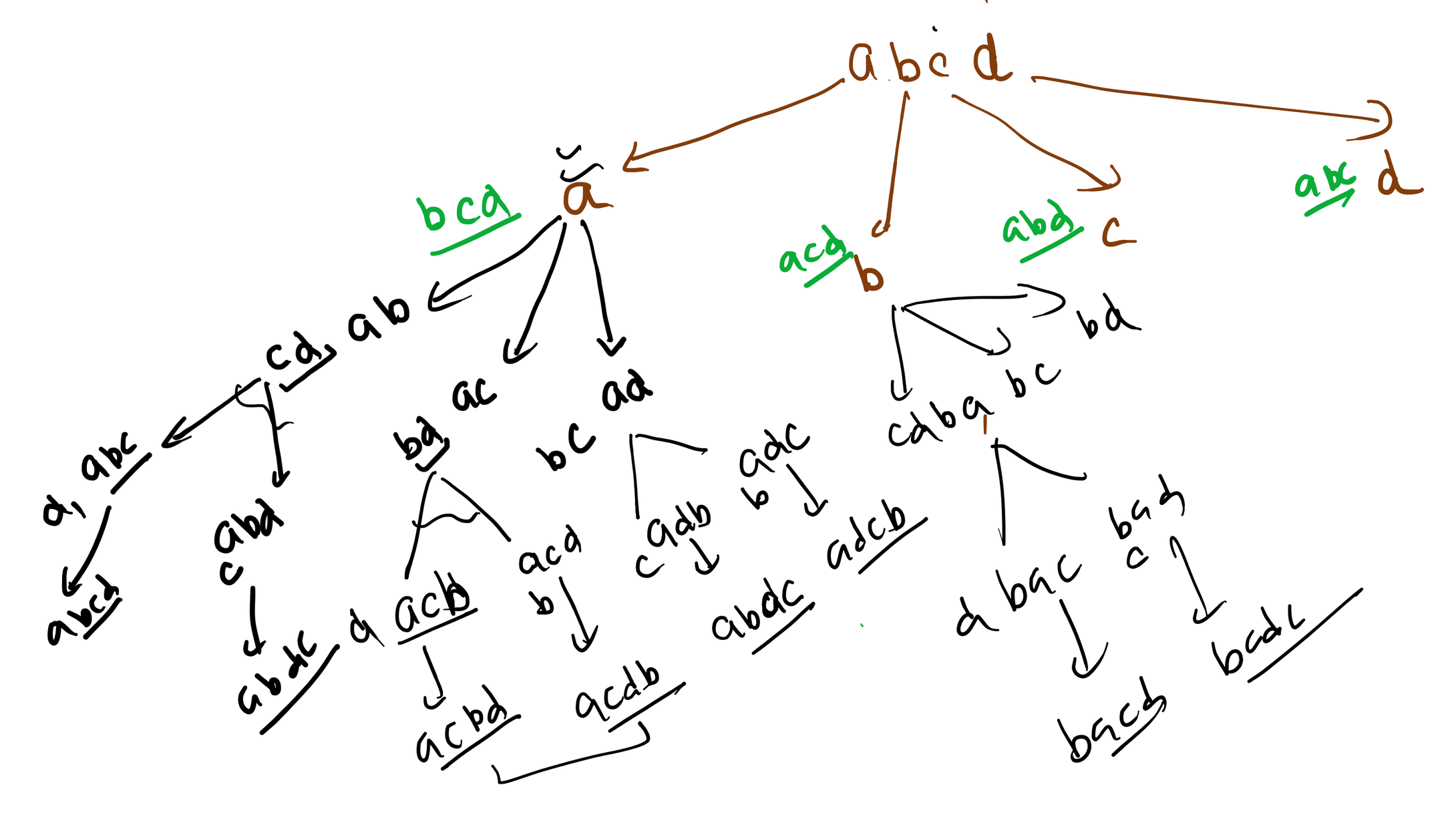


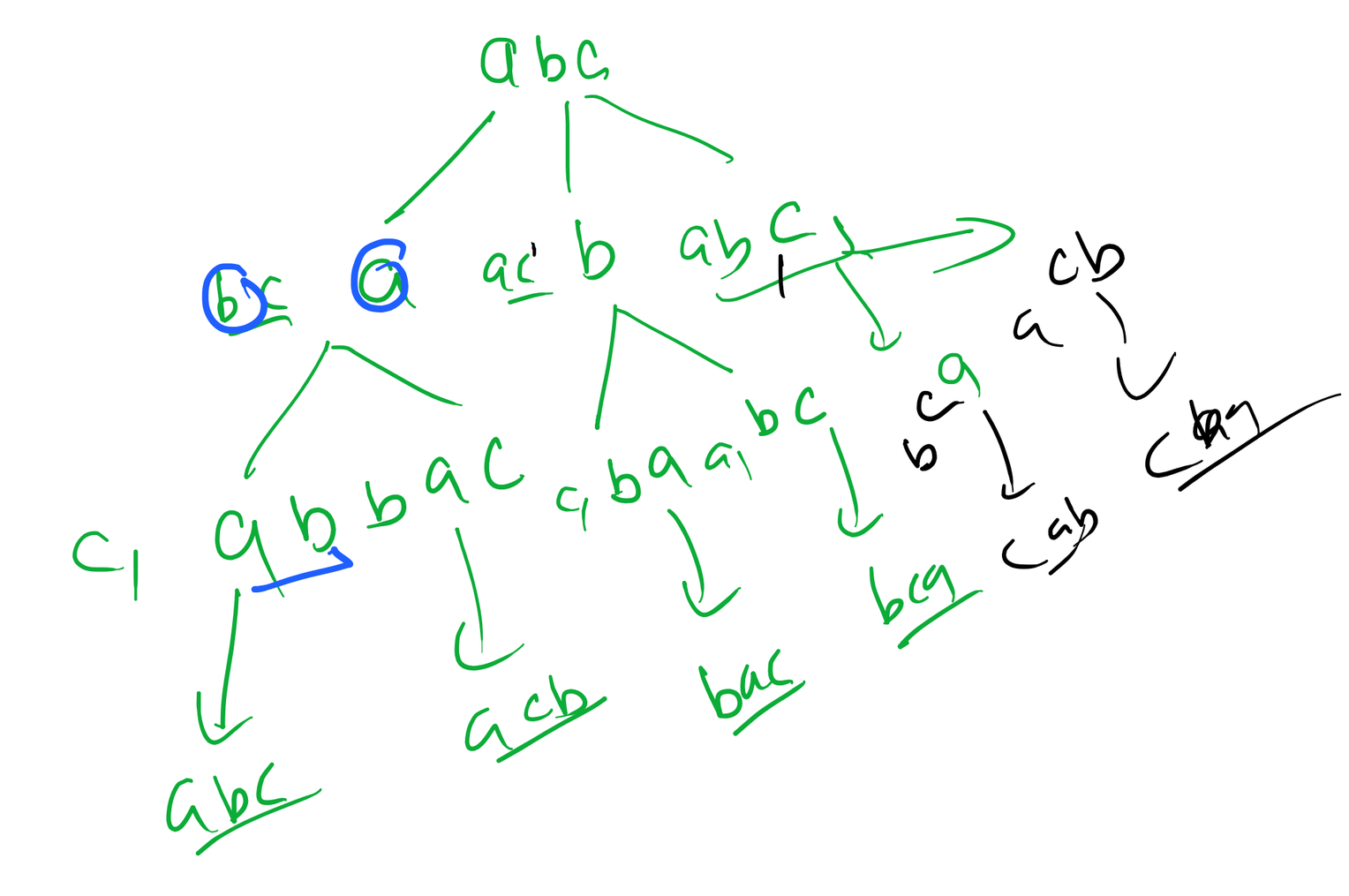












```java
public static void Print(String ques, String ans) {
        if (ques.length() == 0) {
            System.out.println(ans);
            return;
        }
        for (int i = 0; i < ques.length(); i++) {
            char ch = ques.charAt(i);
            String s1 = ques.substring(0, i);
            String s2 = ques.substring(i + 1);
            Print(s1 + s2, ans + ch);
        }

    }
```