Tree formation logic:

1. Fixing position at each level.
2. Both position and input are considered as option at each level.

How many positions are available for next level on a particular brach of tree ?

Positions ahead of position fixed in previous level. Becuase we are following combinatorics, so need to avoid duplicate generation.

How many inputs available for next level on a particular branch of tree ?

Chars in input string ahead of position fixed in previous level. Becuase we are following combinatorics, so need to avoid duplicate generation.

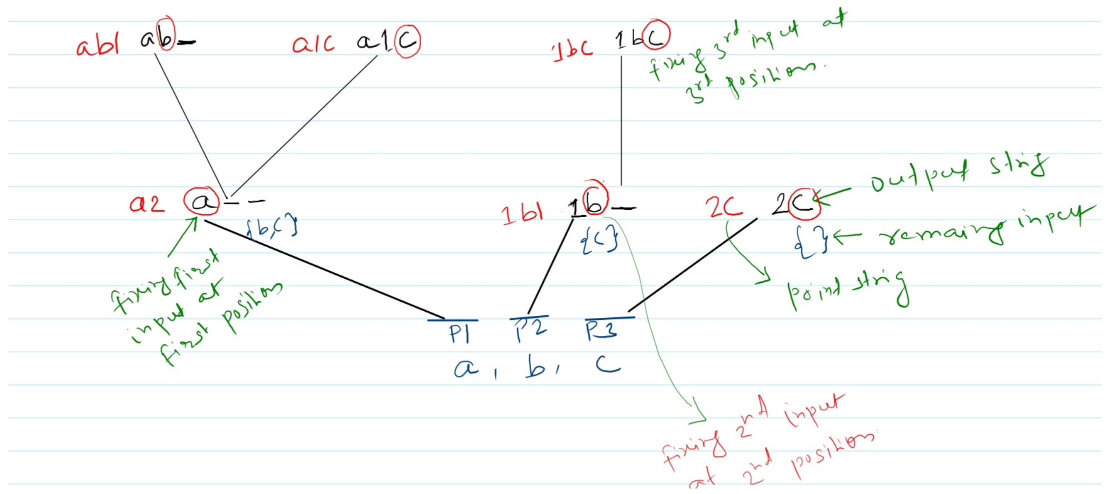
Empty_space calculation logic: It is happening with respect to current position to fix.

1. before_empty_space count = (position_tried_as_option - curentPosToFix)
2. end_empty_space count = (end_input_index - position_tried_as_option)

output String : previous_level_output + before_empty_count + char_at_position_tried_as_option

print String : output + end_empty_count

```java
private void printAbbreviationUsingPIE(char[] input, String output, int posToFix) {

    System.out.println((input.length - posToFix) == 0 ? output : output + (input.length - posToFix));

    int currentPosToFix = posToFix;
    // core logic
    while (posToFix < input.length) {

        String newOutput = output;
        if (posToFix - currentPosToFix != 0) {
            newOutput = newOutput + (posToFix - currentPosToFix);
        }
        newOutput = newOutput + input[posToFix];

        printAbbreviationUsingPIE(input, newOutput, ++posToFix);

    }

}
```



ab1
ab_
a1c
a1c
1bc
1bc
fixing 3rd input at 3rd position.
a2
a_ _
{b,c}
fixing first input at first position
1b1
1b_
{c}
2c
2c
output string
{}
remaing input
print string
P1
P2
P3
a, b, c
fixing 2nd input at 2nd position.