⇒ Find all permutations in Sorted (lexicographic) order in a string. (can be duplicate char).

eg: AABC

$\frac{4!}{2!} = 12$

1. AABC
2. AACB
3. ABAC
4. ABCA
5. ACAB
6. ACBA
7. BAAC
8. BACA
9. BCAA
10. CAAB
11. CABA
12. CBAA

all permutation in Sorted. (lexicographic) order

AABBBC

$= \frac{6!}{2!3!} = 60$

⇒ Solve :-

AABC ⟶

A B C → String array → Should be sorted

2 1 1 → count array

↓

for this initially stored in TreeMap

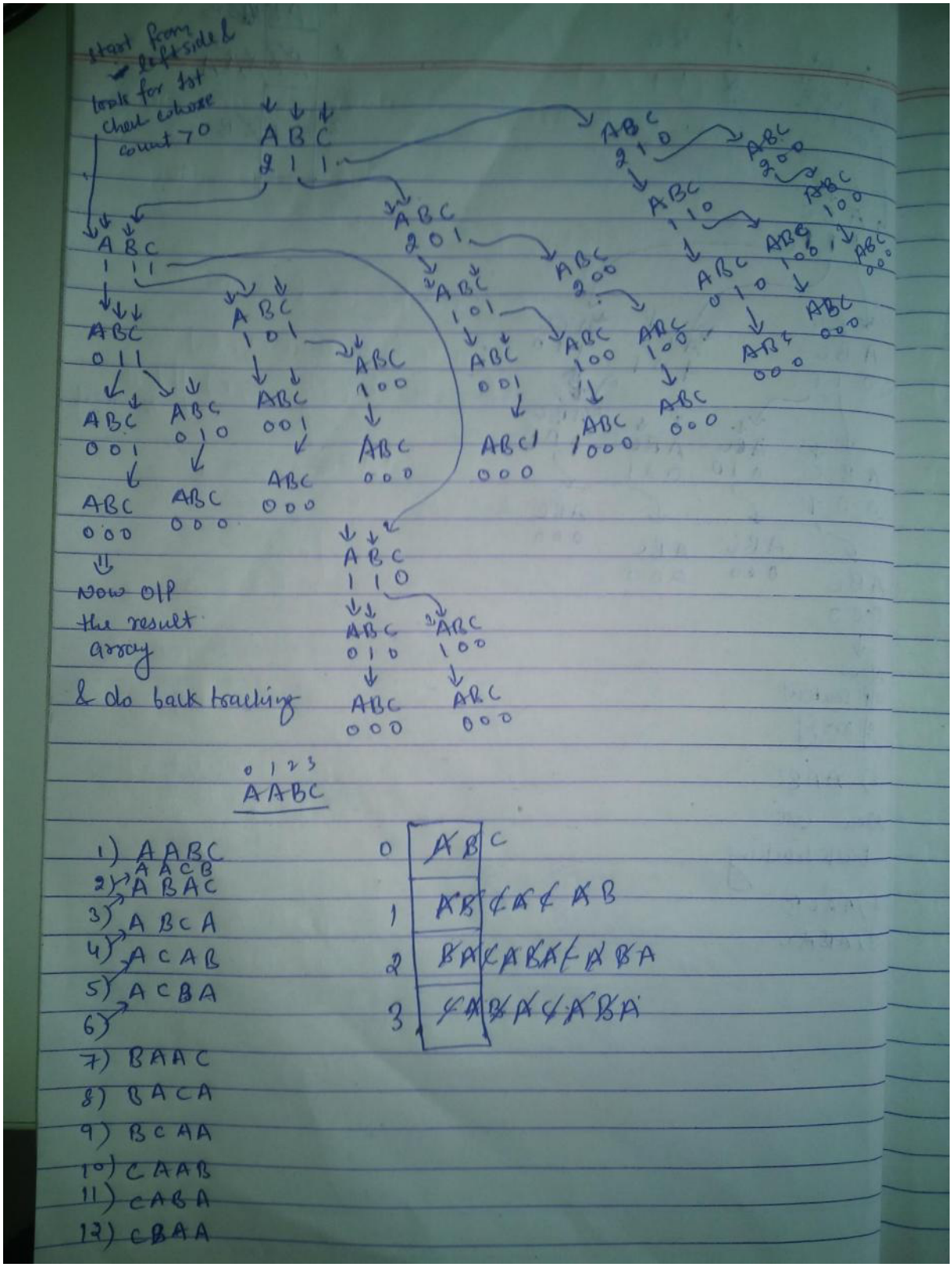
↓

(char, value) pair

↓

then in String & count

Take an result list


Start from left side & look for 1st char colour count 7 0
Now OP the result array
& do back tracking
0 1 2 3
AABC
1) AABC
2) AACB
3) ABAC
4) ABCA
5) ACAB
6) ACBA
7) BAAC
8) BACA
9) BCAA
10) CAAB
11) CABA
12) CBAA
0
1
2
3

(1) Take a Count array which store Count of character.

(2) Take result array of size length of String.

(3) Idea behind Alg. :- here Str length = 4

- Go from left to right look for a character whose Count is greater than Zero
- Then put character into its corresponding position in the result array by looking at the depth of the recursion. So (3)
- So, if have third level in recursion we will put at the third position in the result array.
- Then we gonna go into the recursion again by decrementing the count :: we use this char then we came backs from the recursion we restore the Count back to its original value & then ~~going~~ look for next available character on the right side of this particular char. & keep repeating this process