PROBLEM STATEMENT:

We are given a string str, say "abc", we have to return all the permutations of the string in lexicographically increasing order.

A string str1 is lexicographically less than a string str2 if:

1. str1 ≠ str2
2. str1 is a prefix of str2

OR

2. There exists some i $(0 \le i \le \min(|str1|, |str2|))$ such that str1[i] < str2[i] (ASCII values) and for all $0 \le j < i$, str1[j] = str2[j].

Example: str1 = "abcdefg"
(indices 0 1 2 3 4 5 6)
str2 = "abcxyz"
(indices 0 1 2 3 4 5)

Equal — First Difference where str1[i] < str2[i]

⇒ str1 < str2 lexicographically

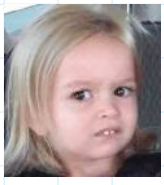
APPROACH: Say our string is "abc"

We want to generate the following permutations:

abc, acb, bac, bca, cab, cba

So, we want to swap elements throughout the string to generate the permutations.

Confused? No worries.



EXAMPLE: str = "abc"

Starting with i = 0, abc (i = 0, indices 0 1 2)

① Swap a with a, increment i. → a b c (i = 1)

② Swap a with b, increment i. → b a c (i = 1)

③ Swap a with c, increment i. → c b a (i = 1)

Call the function again for i = 1 for each output in steps ① to ③.

④ a b c (i = 1) ⇒ Swap b with b ⇒ a b c (i = 2) ①

⑤ a b c (i = 1) ⇒ Swap b with c ⇒ a c b (i = 2) ②

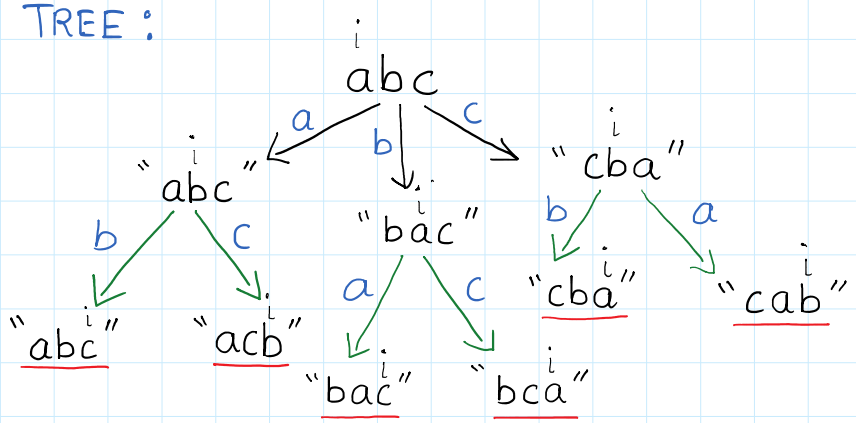
⑥ b a c (i = 1) ⇒ Swap a with a ⇒ b a c (i = 2) ③

⑦ b a c (i = 1) ⇒ Swap a with c ⇒ b c a (i = 2) ④

⑧ c b a (i = 1) ⇒ Swap b with b ⇒ c b a (i = 2) ⑤

⑨ c b a (i = 1) ⇒ Swap b with a ⇒ c a b (i = 2) ⑥

RECURSION TREE:



Code :

```
class Solution {
private:
    void solve(vector<int> nums, vector<vector<int>>& ans, int i) {
        if(i >= nums.size()) {
            ans.push_back(nums);
            return;
        }
        for(int j=i;j<nums.size();j++) {
            swap(nums[i], nums[j]);
            solve(nums, ans, i+1);
            swap(nums[i], nums[j]);
        }
    }
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;
        int index = 0;
        solve(nums, ans, index);
        return ans;
    }
};
```

When you do a swap & generate all permutations for the given index, you need to swap them again to bring the array back to its original state.

EXAMPLE :