13. LC 46 Print all Permutations of a String/Array | Recursion | Approach - 2

46. Permutations

Medium 11464 204 Add to List Share

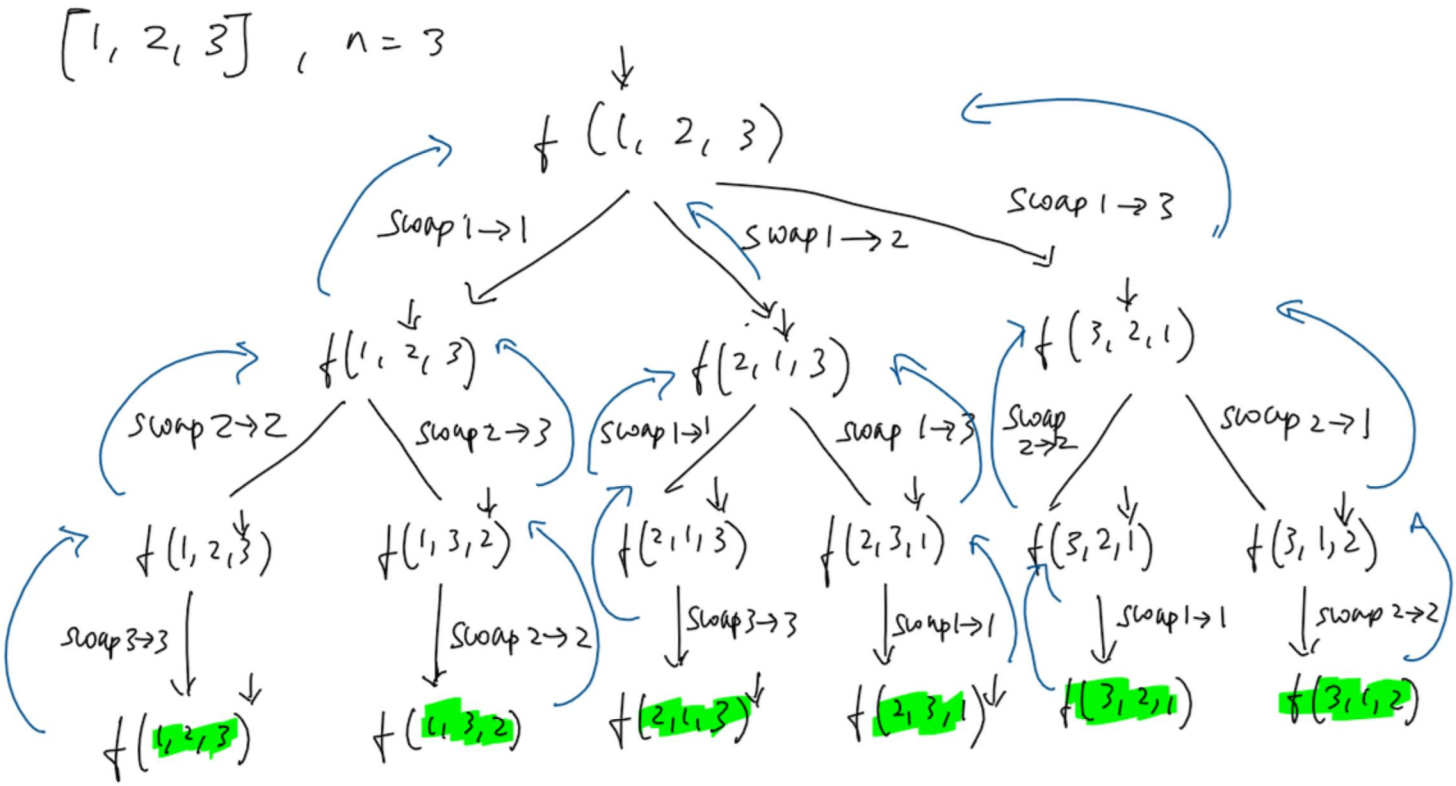
Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Example 1:

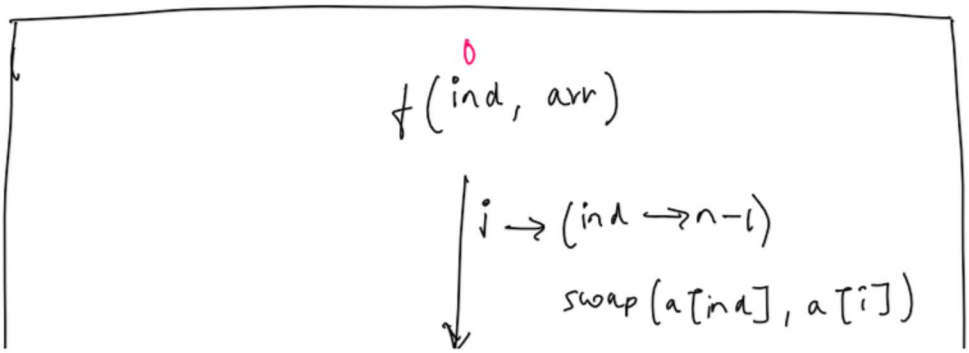
```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

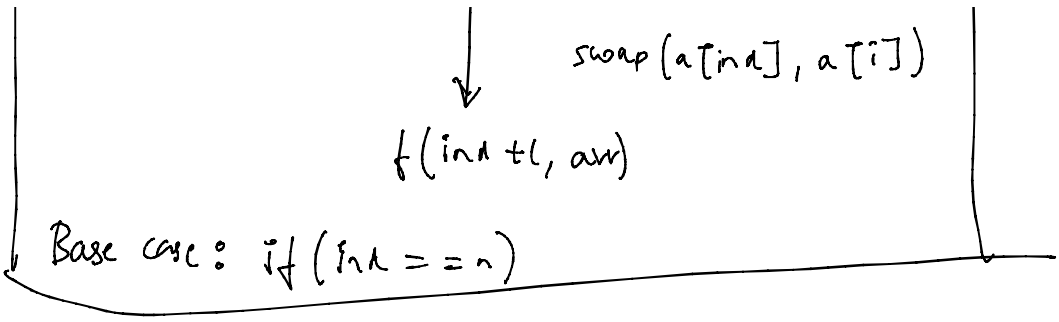
Example 2:

```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```



o/p ⇒ 123, 132, 213, 231, 321, 312





Intitution behind swapping : that every number should be at particular index.

$$T.C \Rightarrow O(n! \times n)$$

$$S.C \Rightarrow O(1)$$

STEPS:

* Given nums array, So declare an ans vector of vector that store all the permutations.
* Call an recursive functions that start with zero, nums array and ans array.

Recursion:

* Go from index to n-1 and swap. Once the swap has done call recursion for the next step. After coming back from recursion make sure you re-swap it because for the next element the swap will not take place.

Java • Autocomplete

```java
// Approach - 2
class Solution {
    private void rPermute(int ind, int[] nums, List<List<Integer>> ans){
        // base case
        if(ind == nums.length){
            // copy data struct to ans
            List<Integer> ds = new ArrayList<>();
            for(int i = 0; i < nums.length; i++){
                ds.add(nums[i]);
            }
            ans.add(new ArrayList<>(ds));
            return;
        }
        for(int i = ind; i < nums.length; i++){
            swap(i, ind, nums);
            rPermute(ind + 1, nums, ans);
            swap(i, ind, nums); // reswap it when you come back from Recursion
        }
    }

    private void swap(int i, int j, int[] nums){
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }

    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        rPermute(0, nums, ans);
        return ans;
    }
}
```

C++ • Autocomplete

```cpp
// Approach - 2 (using swap)
class Solution {
private:
    void rPermute(int ind, vector<int> &nums, vector<vector<int>> &ans){
```

C++ • Autocomplete

```cpp
// Approach - 2 (using swap)
class Solution {
private:
    void rPermute(int ind, vector<int> &nums, vector<vector<int>> &ans){
        // base case
        if(ind == nums.size()){
            ans.push_back(nums);
            return;
        }
        for(int i = ind; i < nums.size(); i++){
            swap(nums[ind], nums[i]);
            rPermute(ind + 1, nums, ans);
            swap(nums[ind], nums[i]);
        }
    }
};
```