46. Permutations

Medium | Topics | Companies

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:

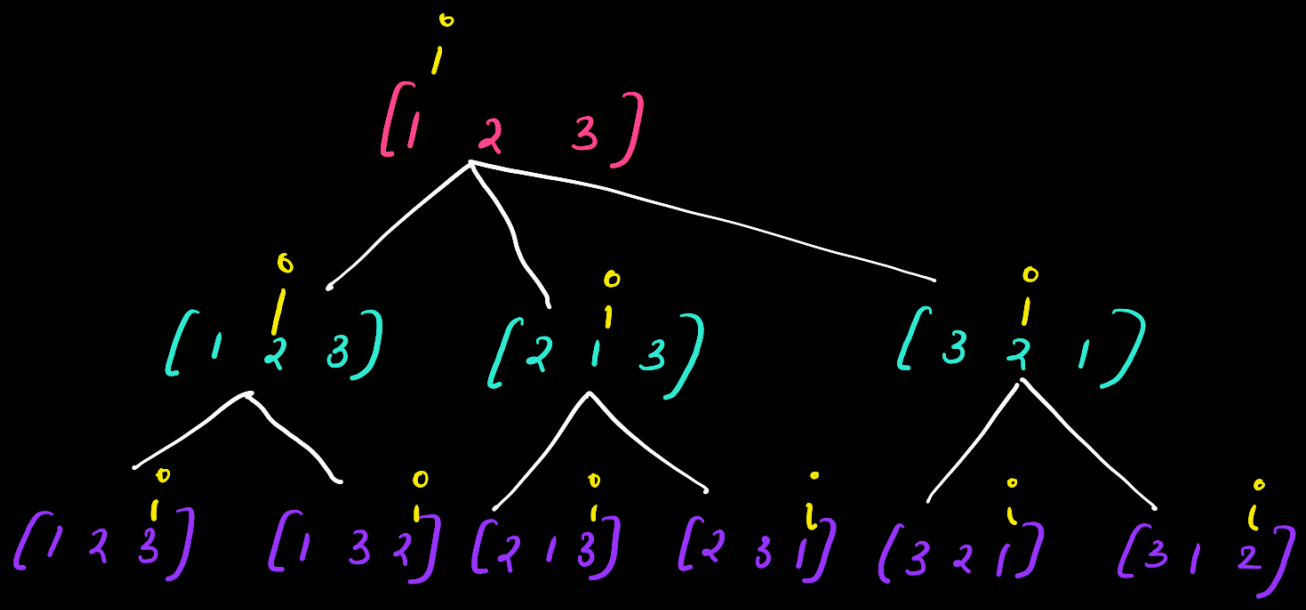
```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

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

Constraints:

- `1 <= nums.length <= 6`
- `-10 <= nums[i] <= 10`
- All the integers of `nums` are **unique**.




```cpp
class Solution {
public:
    void find(vector<int> nums,int i,vector<vector<int>> &ans){

        if(i == nums.size()-1){
            ans.push_back(nums);
            return;
        }
```

copy is created

```cpp
class Solution {
public:
    void find(vector<int> &nums,int i,vector<vector<int>> &ans){

        if(i == nums.size()-1){
            ans.push_back(nums);
            return;
        }
```

just reference

```
        for(int j=i;j<nums.size();j++){
            swap(nums[i],nums[j]);
            find(nums,i+1,ans);
        }
    }

    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;

        find(nums,0,ans);
        return ans;
    }
};
```

```
        for(int j=i;j<nums.size();j++){
            swap(nums[i],nums[j]);
            find(nums,i+1,ans);
            swap(nums[i],nums[j]);
        }
    }

    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;

        find(nums,0,ans);
        return ans;
    }
};
```

faster

$T(n) : O(n!)$ i.e. dominated by last level function calls in recursion tree

$S(n)$: no xtra space

Approach a: Using an xtra data structure and a visited array

```
class Solution {
public:
    void find(vector<int> &nums,vector<int> &curr,vector<bool> &isThere,vector<vector<int>> &ans){
        if(curr.size() == nums.size()){
            ans.push_back(curr);
            return;
        }

        for(int i=0;i<nums.size();i++){
            if(!isThere[i]){
                curr.push_back(nums[i]);
                isThere[i] = true;
                find(nums,curr,isThere,ans);
                curr.pop_back();
                isThere[i] = false;
            }
        }
    }

    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;

        vector<int> curr;
        vector<bool> isThere(nums.size(),false);

        find(nums,curr,isThere,ans);
        return ans;
    }
};
```

$T(n) : O(n!)$

$S(n) : O(n)^0 + O(n)$