# **Leetcode 46 – Permutations**

## Problem Understanding

Given an array nums of **distinct integers**, return **all possible permutations**.  
Each permutation must contain all elements, and **no element should repeat** in the same permutation.

### Example:

Input: [1, 2, 3]

Output: [

[1,2,3], [1,3,2],

[2,1,3], [2,3,1],

[3,1,2], [3,2,1]

]

## Optimized Java Solution (In-place Swapping)

class Solution {

public List<List<Integer>> permute(int[] nums) {

List<List<Integer>> res = new ArrayList<>();

backtrack(nums, 0, res);

return res;

}

public void backtrack(int[] nums, int start, List<List<Integer>> res) {

if (start == nums.length) {

List<Integer> perm = new ArrayList<>();

for (int val : nums) perm.add(val); // Convert array to list

res.add(perm);

return;

}

for (int i = start; i < nums.length; i++) {

swap(nums, i, start); // Choose

backtrack(nums, start + 1, res); // Explore

swap(nums, i, start); // Un-choose (backtrack)

}

}

public void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

## Dry Run Using Table (nums = [1, 2, 3])

|  |  |  |  |
| --- | --- | --- | --- |
| Call Level | start | nums[] | Action |
| 0 | 0 | [1, 2, 3] | fix index 0 |
| 1 | 1 | [1, 2, 3] | fix index 1 |
| 2 | 2 | [1, 2, 3] | fix index 2 |
| 3 | 3 | [1, 2, 3] | ✅ added [1, 2, 3] |
| backtrack | 2 | [1, 3, 2] | ✅ added [1, 3, 2] |
| backtrack | 1 | [2, 1, 3] | ✅ added [2, 1, 3] |
| backtrack | 2 | [2, 3, 1] | ✅ added [2, 3, 1] |
| backtrack | 1 | [3, 2, 1] | ✅ added [3, 2, 1] |
| backtrack | 2 | [3, 1, 2] | ✅ added [3, 1, 2] |

Final Output: [[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,2,1], [3,1,2]]

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(n!) |
| Space | O(n!) for result + O(n) recursion stack |

* Total number of permutations: n!
* Backtracking uses depth = n

## Alternate Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time | Space | Notes |
| In-place Swapping | O(n!) | O(n) | Space-efficient, no extra flags |
| Backtracking w/ used[] | O(n!) | O(n) | More readable for beginners |
| Iterative | O(n!) | O(n!) | Less intuitive, harder to scale |