Permutations

For this problem we need to generate all possible permutations of a list of distinct integers; the approach is backtracking.

Key Idea:

- Build permutation incrementally.
- Use a used array (or in-place swapping) to track visited elements.
- When current permutation length == nums.size(), add to result.

Algorithm:

1. Initialize an empty result vector.
2. Use recursion (backtrack):
   - If path.size() == nums.size(), push path to result.
   - Otherwise, iterate over nums:
     - Skip already used elements.
     - Add current number mark as used.
     - Recurse.
     - Backtrack: remove last number, mark unused.

Alternative: In-Place Swapping:

- At each recursion level, swap current index with i and recurse.

Complexity:

- Time: $O(n \times n!)$ – total permutation count is $n!$
- Space: $O(n)$ recursion stack.