46. Permutations

Medium

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

```
class Solution:
class Solution:
   def permute(self, nums: List[any]) -> List[List[any]]:
                                                                       def permute(self, nums: List[int]) -> List[List[int]]:
       result = []
                                                                           result = []
       permutation = []
                                                                           def backtrack(i):
        def backtrack():
                                                                               if i == len(nums):
           if len(permutation) == len(nums):
                                                                                   result.append(nums[:])
                result.append(permutation.copy())
                                                                                   return
                                                                               for idx in range(i, len(nums)):
            for num in nums:
                                                                                   nums[i], nums[idx] = nums[idx], nums[i]
                if num not in permutation:
                                                                                   backtrack(i + 1)
                                                                                   nums[i], nums[idx] = nums[idx], nums[i]
                    permutation.append(num)
                    backtrack()
                                                                           backtrack(0)
                    permutation.pop()
                                                                           return result
        backtrack()
                                                           Nums = [1,2,3]
        return result
                                                                         backtrack(i):
                                                                                                        result. append[1,2,
                                                                         if i == len(nums):
                                                                             result.append(nums[:]
                                                                             return
def backtrack():
                                                                         for idx in range(i, len(nums)):
    if len(permutation) == len(nums):
                                                                            nums[i], nums[idx] = nums[idx], nums[i]
        result.append(permutation.copy())
                                                                             backtrack(i + 1)
        return
                                                                             nums[i], nums[idx] = nums[idx], nums[i]
    for num in nums:
                                                                               1=0.idx=0
        if num not in permutation:
                                                                                  nums [ o], nums [ o] = hums [ o], nums
            permutation.append(num)
            backtrack()
                                                                               backtrack (0 m)
            permutation.pop()
                                                                               backgrack (1+1
                                                                                 1:2, 1/2:2
                                                                                backtrack (2+1)
```

```
class Solution:
   def permute(self, nums: List[any]) -> List[List[any]]:
        result = []
       def backtrack(nums list, current permutation):
            if not nums list:
                result.append(current permutation[:])
                return
            for i in range(len(nums list)):
                n = nums list[i]
                current permutation.append(n)
                backtrack(nums list[:i] + nums list[i+1:], current permutation)
                current permutation.pop()
        backtrack(nums, [])
        return result
                         rif nums. list is empty
    def backtrack(nums list, current permutation):
        if not nums list◆
            result.append(current permutation[:])
            return
        for i in range(len(nums list)):
            n = nums list[i]
            current permutation.append(n)
            backtrack(nums list[:i] + nums list[i+1:], current permutation)
            current permutation.pop()
```