0047. 全排列 II

ITCharge 大约 1 分钟

- 标签：数组、回溯
- 难度：中等

题目链接

- 0047. 全排列 II - 力扣

题目大意

描述：给定一个可包含重复数字的序列 `nums` 。

要求：按任意顺序返回所有不重复的全排列。

说明：

- $1 \leq nums.length \leq 8$。
- $-10 \leq nums[i] \leq 10$。

示例：

- 示例 1：

```py
输入：nums = [1,1,2]
输出：[[1,1,2],[1,2,1],[2,1,1]]
```

- 示例 2：

```py
输入：nums = [1,2,3]
输出：[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

解题思路

思路 1：回溯算法

这道题跟「0046. 全排列 」不一样的地方在于增加了序列中的元素可重复这一条件。这就涉及到了如何去重。

我们可以先对数组 `nums` 进行排序，然后使用一个数组 `visited` 标记该元素在当前排列中是否被访问过。

如果未被访问过则将其加入排列中，并在访问后将该元素变为未访问状态。

然后再递归遍历下一层元素之前，增加一句语句进行判重： `if i > 0 and nums[i] == nums[i - 1] and not visited[i - 1]: continue` 。

然后再进行回溯遍历。

思路 1：代码

```py
class Solution:
    res = []
    path = []
    def backtrack(self, nums: List[int], visited: List[bool]):
        if len(self.path) == len(nums):
            self.res.append(self.path[:])
            return
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i - 1] and not visited[i - 1]:
                continue

            if not visited[i]:
                visited[i] = True
                self.path.append(nums[i])
                self.backtrack(nums, visited)
                self.path.pop()
                visited[i] = False

    def permuteUnique(self, nums: List[int]) -> List[List[int]]:
```

```
        self.res.clear()
        self.path.clear()
        nums.sort()
        visited = [False for _ in range(len(nums))]
        self.backtrack(nums, visited)
        return self.res
```

思路 1：复杂度分析

- **时间复杂度**：$O(n \times n!)$，其中 n 为数组 `nums` 的元素个数。
- **空间复杂度**：$O(n)$。