

LintCode领扣题解 (/problem) / 全排列 · Permutations

全排列 · Permutations

中文

[微软 \(/problem/?tags=microsoft\)](/problem/?tags=microsoft)[脸书 \(/problem/?tags=facebook\)](/problem/?tags=facebook)[领英 \(/problem/?tags=linkedin\)](/problem/?tags=linkedin)[递归 \(/problem/?tags=recursion\)](/problem/?tags=recursion)

描述

给定一个数字列表, 返回其所有可能的排列。

❗ 你可以假设没有重复数字。

样例

样例 1:

```
输入: [1]
输出:
[
  [1]
]
```

样例 2:

```
输入: [1,2,3]
输出:
[
  [1,2,3],
  [1,3,2],
  [2,1,3],
  [2,3,1],
  [3,1,2],
  [3,2,1]
]
```

挑战

使用递归和非递归分别解决。

在线评测地址: <https://www.lintcode.com/problem/permutations/> (<https://www.lintcode.com/problem/permutations/>)

收起题目描述 ^

语言类型

[ALL \(37\)](#)[python \(17\)](#)[cpp \(10\)](#)[java \(8\)](#)[javascript \(2\)](#)[上传题解](#)

令狐冲

更新于 11/5/2020, 8:40:31 AM

使用深度优先搜索算法。使用 visited 数组记录某个数是否被放到 permutation 里了。

[java](#)[python](#)

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: nums: A list of integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        if not nums:
            return [[]]

        permutations = []
        self.dfs(nums, [], set(), permutations)
        return permutations

    def dfs(self, nums, permutation, visited, permutations):
        if len(nums) == len(permutation):
            permutations.append(list(permutation))
            return

        for num in nums:
            if num in visited:
                continue
            permutation.append(num)
            visited.add(num)
            self.dfs(nums, permutation, visited, permutations)
            visited.remove(num)
            permutation.pop()
```

👍 获赞 25

💬 7 条评论

你的口袋题库

2000+ 算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 6/9/2020, 7:03:58 AM

Given a collection of numbers, return all possible permutations.

For example, 1,2,3 () have the following permutations: 1,2,3 (), 1,3,2 (), 2,1,3 (), 2,3,1 (), 3,1,2 (), and 3,2,1 ().

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
// version 2: Recursion
class Solution {
```

```

public:
    /**
     * @param nums: A list of integers.
     * @return: A list of permutations.
     */
    vector<vector<int> > permute(vector<int> nums) {

        vector<vector<int> > res;
        int n = nums.size();
        if (n == 0) {
            res.push_back(vector<int>());
            return res;
        }

        helper(res, nums, n - 1);

        return res;
    }

    void helper(vector<vector<int> > &res, vector<int> nums, int n){

        if(n == 0){
            res.push_back(nums);
        }

        for(int i = 0 ; i <= n; i++){
            swap(nums[i], nums[n]);
            helper(res, nums, n - 1);
            swap(nums[i], nums[n]);
        }
    }
};

// version 1: Non-Recursion
class Solution {
public:
    /**
     * @param nums: A list of integers.
     * @return: A list of permutations.
     */
    vector<vector<int> > permute(vector<int> nums) {
        vector<vector<int> > permutations;
        if (nums.size() == 0) {
            permutations.push_back(vector<int>());
            return permutations;
        }

        int n = nums.size();
        vector<int> stack;
        bool inStack[n];
        for (int i = 0; i < n; i++) {
            inStack[i] = false;
        }

        stack.push_back(-1);
        while (stack.size() != 0) {
            // pop the last
            int last = stack[stack.size() - 1];
            stack.pop_back();
            if (last != -1) {
                inStack[last] = false;
            }

            // increase the last, find the next bigger & available number
            int next = -1;
            for (int i = last + 1; i < n; i++) {
                if (inStack[i] == false) {

```

```

        next = i;
        break;
    }
}
if (next == -1) {
    continue;
}

// generate the next permutation
stack.push_back(next);
inStack[next] = true;
for (int i = 0; i < n; i++) {
    if (!inStack[i]) {
        stack.push_back(i);
        inStack[i] = true;
    }
}

// generate real permutation from index
vector<int> permutation;
for (int i = 0; i < n; i++) {
    permutation.push_back(nums[stack[i]]);
}
permutations.push_back(permutation);
}

return permutations;
}
};

```

👍 获赞 1

💬 添加评论



令狐冲

更新于 6/9/2020, 7:03:58 AM

Given a collection of numbers, return all possible permutations.

For example, 1,2,3 () have the following permutations: 1,2,3 (), 1,3,2 (), 2,1,3 (), 2,3,1 (), 3,1,2 (), and 3,2,1 ().

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
# Recursion
class Solution:
    """
    @param nums: A list of Integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        # write your code here
        def _permute(result, temp, nums):
            if nums == []:
                result += [temp]
            else:
                for i in range(len(nums)):
                    _permute(result, temp + [nums[i]], nums[:i] + nums[i+1:])

        if nums is None:

```

```
        return []

    if nums is []:
        return [[]]

    result = []
    _permute(result, [], sorted(nums))
    return result

# Non-Recursion
class Solution:
    """
    @param nums: A list of Integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        if nums is None:
            return []
        if nums == []:
            return [[]]
        nums = sorted(nums)
        permutation = []
        stack = [-1]
        permutations = []
        while len(stack):
            index = stack.pop()
            index += 1
            while index < len(nums):
                if nums[index] not in permutation:
                    break
                index += 1
            else:
                if len(permutation):
                    permutation.pop()
                continue

            stack.append(index)
            stack.append(-1)
            permutation.append(nums[index])
            if len(permutation) == len(nums):
                permutations.append(list(permutation))
        return permutations
```

👍 获赞 1

💬 6 条评论



九章-小原

更新于 6/9/2020, 7:03:51 AM

算法

DFS (回溯法)

算法分析

对于全排列，比如 [1,2,3] 的全排列，我们按顺序穷举，步骤如下： 1. 首先会固定第一位为 1，然后第二位如果取2，最后一位只能是3；再可以把第二位变成3，第三位就只能2； 2. 接下来变化第一位变成 2，然后再像上述的过程穷举后两位分别是1,3； 3. 最后第一位是3，再穷举后两位分别是1,2，整个过程就完成了。

以上的过程，很适合使用DFS (回溯法) 进行实现。

算法步骤

1. 根据题目要求，首先需要定义并初始化一个布尔类型used数组记录每个数字是否已被使用，一个数组current记录当前遍历的排列，一个二维数组results记录所有

结果；

2. 对输入的数字列表nums进行深度优先搜索，传入的参数包括：数字列表nums、used、current、results；
 - 边界条件：当current中的元素个数和nums的相同，代表一次遍历完毕，将当前的current加入results并return；
 - 遍历nums，如果nums中当前这个元素未被使用，则在used中标记其已使用，将其加入current，并递归调用dfs；
 - 调用完完后需要回退，即在used中标记其未使用；

复杂度

- 时间复杂度： $O(\sum_{m=1}^n P_n^m)$
 - 对于每一位，可以从n个元素中选择k个来放置，共有n位。
- 空间复杂度： $O(N!)$
 - 共有N!个全排列，故需要保存N!个解

[java](#)[c++](#)[python](#)

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param nums: A list of Integers.
    @return: A list of permutations.
    """

    def permute(self, nums):
        results = []

        # 如果数组为空直接返回空
        if nums is None:
            return []

        # dfs
        used = [0] * len(nums)
        self.dfs(nums, used, [], results)
        return results

    def dfs(self, nums, used, current, results):

        # 找到一组排列, 已到达边界条件
        if len(nums) == len(current):
            # 因为地址传递, 在最终回溯后current为空导致results中均为空列表
            # 所以不能写成results.append(current)
            results.append(current[:])
            return

        for i in range(len(nums)):
            # i位置这个元素已经被用过
            if used[i]:
                continue

            # 继续递归
            current.append(nums[i])
            used[i] = 1
            self.dfs(nums, used, current, results)
            used[i] = 0
            current.pop()

```

👍 获赞 1 💬 添加评论



令狐冲

更新于 6/9/2020, 7:04:31 AM

Given a collection of numbers, return all possible permutations.

For example, 1,2,3 () have the following permutations: 1,2,3 (), 1,3,2 (), 2,1,3 (), 2,3,1 (), 3,1,2 (), and 3,2,1 ().

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD

```

```
* - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
* - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
*/
public class Solution {
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> results = new ArrayList<>();
        if (nums == null) {
            return results;
        }

        if (nums.length == 0) {
            results.add(new ArrayList<Integer>());
            return results;
        }

        List<Integer> permutation = new ArrayList<Integer>();
        Set<Integer> set = new HashSet<>();
        helper(nums, permutation, set, results);

        return results;
    }

    // 1. 找到所有以permutation 开头的排列
    public void helper(int[] nums,
                      List<Integer> permutation,
                      Set<Integer> set,
                      List<List<Integer>> results) {
        // 3. 递归的出口
        if (permutation.size() == nums.length) {
            results.add(new ArrayList<Integer>(permutation));
            return;
        }

        // [3] => [3,1], [3,2], [3,4] ...
        for (int i = 0; i < nums.length; i++) {
            if (set.contains(nums[i])) {
                continue;
            }

            permutation.add(nums[i]);
            set.add(nums[i]);
            helper(nums, permutation, set, results);
            set.remove(nums[i]);
            permutation.remove(permutation.size() - 1);
        }
    }
}

// Non-Recursion
class Solution {
    /**
     * @param nums: A list of integers.
     * @return: A list of permutations.
     */
    public List<List<Integer>> permute(int[] nums) {
        ArrayList<List<Integer>> permutations
            = new ArrayList<List<Integer>>();
        if (nums == null) {
            return permutations;
        }

        if (nums.length == 0) {
            permutations.add(new ArrayList<Integer>());
            return permutations;
        }
    }
}
```



```
int n = nums.length;
ArrayList<Integer> stack = new ArrayList<Integer>();

stack.add(-1);
while (stack.size() != 0) {
    Integer last = stack.get(stack.size() - 1);
    stack.remove(stack.size() - 1);

    // increase the last number
    int next = -1;
    for (int i = last + 1; i < n; i++) {
        if (!stack.contains(i)) {
            next = i;
            break;
        }
    }
    if (next == -1) {
        continue;
    }

    // generate the next permutation
    stack.add(next);
    for (int i = 0; i < n; i++) {
        if (!stack.contains(i)) {
            stack.add(i);
        }
    }

    // copy to permutations set
    ArrayList<Integer> permutation = new ArrayList<Integer>();
    for (int i = 0; i < n; i++) {
        permutation.add(nums[stack.get(i)]);
    }
    permutations.add(permutation);
}

return permutations;
}
```

👍 获赞 0

💬 1 条评论



令狐冲

更新于 6/9/2020, 7:04:00 AM

DFS

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param: nums: A list of integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        if not nums:
            return [[]]

        permutations = []
        self.dfs(nums, [], set(), permutations)
        return permutations

    def dfs(self, nums, permutation, visited, permutations):
        if len(nums) == len(permutation):
            permutations.append(list(permutation))
            return

        for num in nums:
            if num in visited:
                continue
            permutation.append(num)
            visited.add(num)
            self.dfs(nums, permutation, visited, permutations)
            visited.remove(num)
            permutation.pop()

```

👍 获赞 0

💬 添加评论



九章用户A2PXQ8

更新于 6/9/2020, 7:03:45 AM

solution is the same as JAVA version

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: nums: A list of integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        # write your code here
        self.results = []
        self.dfs(nums, [])
        return self.results

    def dfs(self, nums, temp):

        if len(temp) == len(nums):
            self.results.append(temp[:])
            return

        for i in range(0, len(nums)):
            if nums[i] not in temp:
                temp.append(nums[i])
                self.dfs(nums, temp)
                temp.pop()
```

👍 获赞 9

💬 添加评论



Di

更新于 6/9/2020, 7:03:46 AM

一个非递归的解法, 利用stack。BFS, 找到一个就放一个到result里。



invitation/sh:



```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution(object):
    def permute(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        if not nums:
            return [[]]
        result = []
        stack = [[i] for i in nums]
        while stack:
            last = stack.pop()
            if len(last) == len(nums):
                result.append(last)
                continue
            for n in nums:
                if n not in last:
                    stack.append(last + [n])
        return result
```

 获赞 5 添加评论**Tin**


更新于 6/9/2020, 7:03:47 AM

最易懂的DFS实现, 也最正宗的Python。要点是直接用"for each"循环, 不要用"range() for"循环。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: nums: A list of integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        output = []
        self.dfs(nums, [], output)
        return output

    def dfs(self, nums, workingSet, output):
        if len(nums) == len(workingSet):
            output.append(list(workingSet))
            return

        for num in nums:
            if num not in workingSet:
                workingSet.append(num)
                self.dfs(nums, workingSet, output)
                workingSet.pop()
```

 获赞 4 2 条评论

芋米

更新于 6/9/2020, 7:03:49 AM

Backtrack解法, 每次循环都要回溯。

时间复杂度: $O(2^n)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> res = new ArrayList();
        helper(res, new ArrayList(), nums);
        return res;
    }

    public void helper(List<List<Integer>> res, List<Integer> row, int[] nums) {
        if (row.size() == nums.length) {
            res.add(new ArrayList(row));
            return;
        }
        for (int i = 0; i < nums.length; i++) {
            if (row.contains(nums[i])) {
                continue;
            }
            row.add(nums[i]);
            helper(res, row, nums);
            row.remove(row.size() - 1);
        }
    }
}
```

👍 获赞 2

💬 3 条评论



九章用户IMCU1D

更新于 6/9/2020, 7:03:49 AM

不用递归不用栈, 由前N-1个元素的全排列, 加入第N个元素。

每次看nums中的一个元素, 对之前结果集的每一个permutation, 在所有可能的插入点 (N个) 插入当前元素。

比如nums是 [1, 2, 3], 最初结果集是 [[]]。看到1, 结果集变为 [[1]], 因为空list只有一个位置可以插入。看到2, 扩展为 [[1, 2], [2, 1]], 因为 [1] 有两个位置可以插入。看到3, 扩展为 [[3, 1, 2], [1, 3, 2], [1, 2, 3], [3, 2, 1], [2, 3, 1], [2, 1, 3]], 因为上面的两种排列各有3个插入点。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: nums: A list of integers.
    @return: A list of permutations.
    """
    def permute(self, nums):
        results = [[]]
        for i, n in enumerate(nums):
            tmp = []
            for r in results:
                for j in range(0, i + 1):
                    tmp.append(r[:j] + [n] + r[j:])
            results = tmp
        return results
```

👍 获赞 2 💬 添加评论



九章用户YJTVWU

更新于 6/9/2020, 7:03:57 AM

与令狐冲大师兄的Subsets题解法 <https://www.jiuzhang.com/solution/subsets> (<https://www.jiuzhang.com/solution/subsets>) 非常相似。

缺点大概是比较耗费空间一点, 虽然空间复杂度木有提升。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param nums: A list of integers.
     * @return: A list of permutations.
     */
    vector<vector<int>> permute(vector<int> &nums) {
        vector<vector<int>> results;
        vector<int> temp;
        sort(nums.begin(), nums.end());
        allPerm(results, temp, nums);
        return results;
    }

private:
    void allPerm(vector<vector<int>> &results, vector<int> temp, vector<int> nums) {
        if (nums.empty()) {
            results.push_back(temp);
        }

        for (int i = 0; i < nums.size(); i++) {
            int numi = nums[i];
            temp.push_back(numi);
            nums.erase(nums.begin() + i);
            allPerm(results, temp, nums);
            temp.pop_back();
            nums.insert(nums.begin() + i, numi);
        }
    }
};
```

👍 获赞 1

💬 添加评论

[加载更多题解](#)

进阶课程

直播+互动	直播+互动	直播+互动	互动课
九章算法班 2021 版 8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22....	系统架构设计 System Design 2021 版 成为百万架构师必上。30 课时带你快速掌握 18 大系统架构设计知识点与面...	九章算法面试高频题冲刺班 每期更新 15% 题目，考前押题，一举拿下 FLAG & BATJ Offer	面向对象设计 OOD 应届生及亚马逊面试必考，IT 求职必备基础