1. 题目
2. 题干

给定一个**没有重复** 数字的序列，返回其所有可能的全排列。

1. 示例

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]

]

二．题解

（一）思路

同样是DFS的问题，通过递归，回溯来解决。

因此，考虑该问题的结构组成：递归何时结束？满足什么条件进入下一个递归？递归在回溯时如何回到上一个递归的状态？

1. 递归何时结束？

本问题中，我们设置了一个临时的数组temp存储当前排列的情况，当该数组temp的长度与原序列nums的长度相同时，说明该数组是正确的排列，可以存到最终输出数组中。此时也就是递归结束的位置。

1. 满足什么条件进入下一个递归中？

进入下一个递归，也就是说明当前的数可以放入临时的存储数组temp中，如何判断当前的数是否可以存到temp中呢？本题的要求是对没有重复数字的序列进行全排列，那么temp中当然不能有重复的数字。因此，我们可以用一个标识符数组flag来标识某个数是否被加到了temp数组中，若该数已经被加入temp数组中，flag中该数为true，否则为false。

1. 回溯时如何回到上一个递归的状态？

在递归中，当既不能满足结束递归的条件，也不能满足进入下一个递归的条件时，就要回溯到上一个递归的状态了。那么，我们想想从上一个递归到这一个递归我们干了什么？给temp数组添加了当前的数字，为flag数组修改了当前数字的标识。因此，我们只需要将temp数组弹出尾部数字，修改flag数组对应的标识即可。

1. 代码实现

C++：

class Solution {

public:

    int n;

    vector<vector<int>> ans;

    vector<int> temp;

    vector<bool> flag;

    void backtrace(vector<int>& nums){

        if(temp.size() ==  n){

            ans.push\_back(temp);

            return;

        }

        for(int i = 0; i < n; ++i){

            if(!flag[i]){

                temp.push\_back(nums[i]);

                flag[i] = true;

                backtrace(nums);

                temp.pop\_back();

                flag[i] = false;

            }

        }

    }

    vector<vector<int>> permute(vector<int>& nums) {

        n = nums.size();

        flag.resize(n , false);

        backtrace(nums);

        return ans;

    }

};