15. 3Sum (Medium)

Given an integer array `nums`, return all the triplets `[nums[i], nums[j], nums[k]]` where `nums[i] + nums[j] + nums[k] == 0`, and the indices `i`, `j` and `k` are all distinct.

The output should *not* contain any duplicate triplets. You may return the output and the triplets in any order.

Example 1:

```
Input: nums = [-1,0,1,2,-1,-4]

Output: [[-1,-1,2],[-1,0,1]]
```

Explanation:

`nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.`

`nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.`

`nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.`

The distinct triplets are `[-1,0,1]` and `[-1,-1,2]`.

Example 2:

```
Input: nums = [0,1,1]

Output: []
```

Explanation: The only possible triplet does not sum up to 0.

Example 3:

```
Input: nums = [0,0,0]

Output: [[0,0,0]]
```

Explanation: The only possible triplet sums up to 0.

Constraints:

- `3 <= nums.length <= 1000`
- `10^5 <= nums[i] <= 10^5`

▼ 思路:

作法一：two point比對後以hash map確認是否有重複答案

T:$O(n^2)$, S:$O(n)$

作法一精簡：two point比對前先避免重複相同數值計算

T:$O(n^2)$, S:$O(1)$

作法一

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        vector<vector<int>> ans;
        unordered_map <string,int> check;
        for(int i=0;i<nums.size()-2;i++){
            int j=i+1,k=nums.size()-1;
            while(j<k){
                if(nums[i]+nums[j]+nums[k]>0) k--;
                else if(nums[i]+nums[j]+nums[k]<0) j++;
                else {
                    string s = to_string(nums[i]) + to_string(nums[j]) + to_string(nums[k])
                    if(check.find(s)!=check.end()){
                        j++;
                        continue;
                    }
                    check[s]=1;
                    ans.push_back({nums[i],nums[j++],nums[k]});
                }
            }
        }
        return ans;
```

```
    }
};
```

作法一精簡

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        vector<vector<int>> ans;
        for(int i=0;i<nums.size()-2;i++){
            int j=i+1,k=nums.size()-1;
            if(nums[i]>0) break;
            if(i>0 && nums[i] == nums[i-1]) continue;
            while(j<k){
                if(nums[i]+nums[j]+nums[k]>0) k--;
                else if(nums[i]+nums[j]+nums[k]<0) j++;
                else {
                    ans.push_back({nums[i],nums[j++],nums[k--]});
                    while(nums[j] == nums[j-1] && j<k) j++;
                }
            }
        }
        return ans;
    }
};
```