我开始用的哈希表

有重复问题，案例不能通过。

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {

        int N = nums.size();
        vector<vector<int>> result;

        // 使用哈希表

        // 定义哈希表
        unordered_map<int, int> hashTable;

        for (int i = 0; i < N; i++) {

            for (int j = i + 1; j < N; j++) {
                int sumOfTwoNumber = nums[i] + nums[j];

                int b = -sumOfTwoNumber;

                auto it = hashTable.find(b);

                // 说明找到了
                if (it =≠ hashTable.end()) {
                    vector<int> temp;
                    temp.push_back(it→first);
                    temp.push_back(nums[i]);
                    temp.push_back(nums[j]);
                    result.push_back(temp);
                }
            }

            hashTable[nums[i]] = i;
        }

        return result;
    }
};```
```

去重复后的哈希表

```C++

class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {

        int N = nums.size();
        vector<vector<int>> result;

        // 使用哈希表
        unordered_set<string> seen;

        // 用于存储已找到的三元组，避免重复
        unordered_map<int, int> hashTable;

        for (int i = 0; i < N; i++) {
            for (int j = i + 1; j < N; j++) {
                int sumOfTwoNumber = nums[i] + nums[j];
                int b = -sumOfTwoNumber;

                auto it = hashTable.find(b);
                // 说明找到了
                if (it != hashTable.end() && it->second < i) {
                    // 创建三元组并排序以避免重复
                    vector<int> temp = {it->first, nums[i], nums[j]};
                    sort(temp.begin(), temp.end());

                    // 使用字符串作为唯一标识
                    string key = to_string(temp[0]) + "," + to_string(temp[1]) +
                                 "," + to_string(temp[2]);
                    if (seen.find(key) == seen.end()) {
                        result.push_back(temp);
                        seen.insert(key);
                    }
                }
            }
            // 记录当前数字的位置
            hashTable[nums[i]] = i;
        }

```

```
        return result;
    }
};
```

标准题解（双指针）

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        // 待返回的三元组
        vector<vector<int>> triples;

        for (int i = 0; i < nums.size(); ++i) {
            // 检测重复的 nums[i]
            if (i > 0 && nums[i] == nums[i - 1]) continue;

            int l = i + 1;
            int r = nums.size() - 1;

            // 不同处，因为每个指令执行之前均会考虑 while (l < r)，所以不用考虑越界或者错位
            while (l < r) {
                // 检测重复的 nums[l]
                if (l > i + 1 && nums[l] == nums[l - 1]) {
                    ++l;
                }
                // 检测重复的 nums[r]
                else if (r < nums.size() - 1 && nums[r] == nums[r + 1]) {
                    --r;
                }
                // 均不重复再按照两数之和的思路
                else if (nums[i] + nums[l] + nums[r] > 0) {
                    --r;
                }
                else if (nums[i] + nums[l] + nums[r] < 0) {
                    ++l;
                } else {
                    triples.push_back({nums[i], nums[l], nums[r]});
                    ++l; --r;
                }
            }
        }
```

```
37
38          return triples;
39      }
40  };
```