1. twoSum

```
给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个整数，并返回他们的数组下标。
你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。
示例:
给定 nums = [2, 7, 11, 15], target = 9
因为 nums[0] + nums[1] = 2 + 7 = 9
所以返回 [0, 1
```

Analysis

1.暴力枚举–双循环，时间复杂度高(O(n^2))

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        for(int i = 0; i < nums.size() - 1; i++)
        {
             for(int j = i + 1; j < nums.size(); j++)
            {
                if(nums[i] + nums[j] == target){
                    return {i, j};
                }
            }
        }
        return {};
    }
};
```

2.使用map保存以前遍历结果–时间复杂度为(O(n))–推荐

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        std::unordered_map<int, int> selectedMap;
        for(int i = 0; i<nums.size(); i++)
        {
            auto it = selectedMap.find(target - nums[i]);
            if(it != selectedMap.end())
            {
                return{it->second, i};
```

```
            }
            selectedMap[nums[i]] = i;
        }
        return {};
    }
};
```

Code

```
#include <iostream>
#include <vector>
#include <unordered_map>
using namespace std;
//violent solution O(n^2)
// vector<int> twoSum(vector<int>& nums, int target) {
//     for (int i = 0; i < (int)nums.size() - 1; i++) {
//         for (int j = i+1; j < (int)nums.size(); j++) {
//             if (nums[i] + nums[j] == target) {
//                 return{i, j};
//             }
//         }
//     }
//     return {};
// }
//using map O(n)
vector<int> twoSum(vector<int>& nums, int target) {
    unordered_map<int, int> traversedMap;
    for (int i = 0; i < (int)nums.size(); i++) {
        auto it = traversedMap.find(target - nums[i]);
        if (it != traversedMap.end()) {
            return {it->second, i};
        }
        traversedMap[nums[i]] = i;
    }
    return {};
}
int main() {
    int nums[] = {2,7,11,15};
    std::vector<int> array(nums, nums + 4);
    int target = 9;
    vector<int> result = twoSum(array, target);
    //print
    std::cout << "[" ;
    for (int i = 0; i < (int)result.size(); i++) {
        std::cout << result[i];
        if (i != (int)result.size() - 1) {
            std::cout << ",";
        }
    }
```

```
    std::cout << "]" <<std::endl;
    return 0;
}
```