

1. twoSum

```
1 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。
2 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。
5 示例：
8 给定 nums = [2, 7, 11, 15], target = 9
10 因为 nums[0] + nums[1] = 2 + 7 = 9
11 所以返回 [0, 1]
```

Analysis

1.暴力枚举–双循环，时间复杂度高($O(n^2)$)

```
1  class Solution {
2  public:
3      vector<int> twoSum(vector<int>& nums, int target) {
4          for(int i = 0; i < nums.size() - 1; i++)
5              {
6                  for(int j = i + 1; j < nums.size(); j++)
7                      {
8                          if(nums[i] + nums[j] == target){
9                              return {i, j};
10                         }
11                     }
12             }
13         return {};
14     }
15 };
```

2.使用map保存以前遍历结果–时间复杂度为($O(n)$)–推荐

```
1  class Solution {
2  public:
3      vector<int> twoSum(vector<int>& nums, int target) {
4          std::unordered_map<int, int> selectedMap;
5          for(int i = 0; i < nums.size(); i++)
6              {
7                  auto it = selectedMap.find(target - nums[i]);
8                  if(it != selectedMap.end())
9                      {
10                         return{it->second, i};
11                     }
12             }
13     }
```

```

11         }
12         selectedMap[nums[i]] = i;
13     }
14     return {};
15 }
16 };

```

Code

```

1  #include <iostream>
2  #include <vector>
3  #include <unordered_map>
4  using namespace std;
5  //violent solution O(n^2)
6  // vector<int> twoSum(vector<int>& nums, int target) {
7  //     for (int i = 0; i < (int)nums.size() - 1; i++) {
8  //         for (int j = i+1; j < (int)nums.size(); j++) {
9  //             if (nums[i] + nums[j] == target) {
10 //                 return {i, j};
11 //             }
12 //         }
13 //     }
14 //     return {};
15 // }
16 //using map O(n)
17 vector<int> twoSum(vector<int>& nums, int target) {
18     unordered_map<int, int> traversedMap;
19     for (int i = 0; i < (int)nums.size(); i++) {
20         auto it = traversedMap.find(target - nums[i]);
21         if (it != traversedMap.end()) {
22             return {it->second, i};
23         }
24         traversedMap[nums[i]] = i;
25     }
26     return {};
27 }
28 int main() {
29     int nums[] = {2,7,11,15};
30     std::vector<int> array(nums, nums + 4);
31     int target = 9;
32     vector<int> result = twoSum(array, target);
33     //print
34     std::cout << "[" ;
35     for (int i = 0; i < (int)result.size(); i++) {
36         std::cout << result[i];
37         if (i != (int)result.size() - 1) {
38             std::cout << ",";
39         }
40     }
41     }

```

```
53     std::cout << "]" <<std::endl;  
55     return 0;  
56 }
```