Two Sum

> 给定数组中是否存在两数和为指定target的元素，存在输出其索引位置。

```
class Solution {
public:
  std::vector<int> twoSum(std::vector<int>& nums, int target) {
    std::vector<int>             res;
    std::unordered_map<int, int> map;

    for (int i = 0; i < nums.size(); i++) {
      if (map.count(target - nums[i])) {
        res.push_back(i);
        res.push_back(map[target - nums[i]]);
        return res;
      }

      map[nums[i]] = i;
    }

    return res;
  }
};
```

扩展：

> nums 中可能有多对儿元素之和都等于 target，请你的算法返回所有和为 target 的元素对儿，其中不能出现重复。

```
std::vector<std::vector<int>> twoSumTarget(std::vector<int>& nums, int target) {
  // 数组排序
  sort(nums.begin(), nums.end());
  std::vector<std::vector<int>> res;

  int left = 0, right = nums.size() - 1;
  while(left < right) {
    int tmp = nums[left] + nums[right];
    int tleft = nums[left];
    int tright = nums[right];
    if(tmp == right) {
      res.push_back({nums[left], nums[right]});
      while(left < right && nums[left] == tleft) {
        left++;
      }
```

```
        while(left < right && nums[right]) == tright){
          right--;
        }
      }else if(tmp < target) {
        while(left < right && tleft == nums[left]){
          left++;
        }
      }else {
        while(left < right && tright == nums[right]){
          right--;
        }
      }
    }

    return res;
  }
```

3sum和

```
给定一个数据，求是否存在三个元素的和为0.
```

```
class Solution {
public:
  std::vector<std::vector<int>> threeSum(std::vector<int> &nums) {
    std::vector<std::vector<int>> res;
    threeSum(nums, res, 0);
    return res;
  }

private:
  void threeSum(std::vector<int> &              nums,
                std::vector<std::vector<int>> &res,
                int                             target) {
    std::sort(nums.begin(), nums.end());
    for (int i = 0; i < nums.size(); i++) {
      int left = i + 1, right = nums.size() - 1;
      int t = target - nums[i];
      while (left < right) {
        int sum = nums[left] + nums[right];
        int t1 = nums[left], t2 = nums[right];
        if (sum == t) {
          std::vector<int> tmp;
          tmp.push_back(nums[i]);
          tmp.push_back(nums[left]);
          tmp.push_back(nums[right]);
          res.push_back(tmp);

          while (left < right && nums[left] == t1) {
            left++;
          }

          while (left < right && t2 == nums[right]) {
            right--;
          }
        } else if (sum < t) {
          while (left < right && t1 == nums[left]) {
            left++;
          }
        } else {
          while (left < right && nums[right] == t2) {
            right--;
          }
        }
      }
      while (i + 1 < nums.size() - 1 && nums[i] == nums[i + 1]) {
        i++;
      }
    }
  }
};
```

4Sum

> 寻找一个数据中所有满足`a+b+c+d=target`的元素集合

```
class Solution {
public:
  std::vector<std::vector<int>> fourSum(std::vector<int>& nums, int
target) {
    std::sort(nums.begin(), nums.end());
    std::vector<std::vector<int>> res;

    fourSum(nums, res, target);
    return res;
  }

private:
  void fourSum(std::vector<int>&              nums,
               std::vector<std::vector<int>>& res,
               int                            target) {
    std::sort(nums.begin(), nums.end());
    for (int i = 0; i < nums.size() - 1; i++) {
      int t = target - nums[i];
      for (int j = i + 1; j < nums.size(); j++) {
        int a    = t - nums[j];
        int left = j + 1, right = nums.size() - 1;
        while (left < right) {
          int sum = nums[left] + nums[right];
          int t1 = nums[left], t2 = nums[right];
          if (sum == a) {
            std::vector<int> tmp;
            tmp.push_back(nums[j]);
            tmp.push_back(nums[i]);
            tmp.push_back(nums[left]);
            tmp.push_back(nums[right]);
            res.push_back(tmp);

            while (left < right && nums[left] == t1) {
              left++;
            }

            while (left < right && t2 == nums[right]) {
              right--;
            }
          } else if (sum < a) {
            while (left < right && t1 == nums[left]) {
              left++;
            }
          } else {
            while (left < right && nums[right] == t2) {
              right--;
            }
          }
        }

        while (j + 1 < nums.size() - 1 && nums[j] == nums[j + 1]) {
          j++;
        }
      }
      while (i + 1 < nums.size() - 2 && nums[i] == nums[i + 1]) {
        i++;
      }
    }
  }
};
```