

# Array & Sorting

## 217. Contains Duplicate

🕒 Created	@January 29, 2022 3:02 PM
▼ Difficulty	Easy
≡ LC Url	<a href="https://leetcode.com/problems/contains-duplicate/">https://leetcode.com/problems/contains-duplicate/</a>
▼ Importance	
:≡ Tag	
≡ Video	<a href="https://www.youtube.com/watch?v=xEXqi7mwPCI&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=54">https://www.youtube.com/watch?v=xEXqi7mwPCI&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=54</a>

Given an integer array `nums`, return `true` if any value appears **at least twice** in the array, and return `false` if every element is distinct.

### Example 1:

```
Input: nums = [1,2,3,1]
Output: true
```

### Example 2:

```
Input: nums = [1,2,3,4]
Output: false
```

### Example 3:

```
Input: nums = [1,1,1,3,3,4,3,2,4,2]
Output: true
```

### Constraints:

- `1 <= nums.length <= 105`
- `-109 <= nums[i] <= 109`

## Solution

```
class Solution:
    def containsDuplicate(self, nums: List[int]) -> bool:
        nums.sort()
        for i in range(len(nums)-1):
            if nums[i] == nums[i+1]:
                return True
        return False
```

```
class Solution:
    def containsDuplicate(self, nums: List[int]) -> bool:
        hashset = set()

        for n in nums:
            if n in hashset:
                return True
            hashset.add(n)
        return False
```

# 242. Valid Anagram

🕒 Created	@January 29, 2022 2:31 PM
▼ Difficulty	Easy
☰ LC Url	<a href="https://leetcode.com/problems/valid-anagram/">https://leetcode.com/problems/valid-anagram/</a>
▼ Importance	
☰ Tag	Array&Sorting
☰ Video	<a href="https://www.youtube.com/watch?v=wScXoa8pN6o&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=51">https://www.youtube.com/watch?v=wScXoa8pN6o&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=51</a>

Given two strings `s` and `t`, return `true` if `t` is an anagram of `s`, and `false` otherwise.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

## Example 1:

```
Input: s = "anagram", t = "nagaram"
Output: true
```

## Example 2:

```
Input: s = "rat", t = "car"
Output: false
```

## Constraints:

- `1 <= s.length, t.length <= 5 * 104`
- `s` and `t` consist of lowercase English letters.

**Follow up:** What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

## Solution

```

class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        '''
        时间空间复杂度都是O(n)
        '''
        if len(s) != len(t):
            return False

        lookup = {}

        for i in s:
            if i not in lookup:
                lookup[i] = 1
            else:
                lookup[i] += 1

        for j in t:
            if j not in lookup:
                return False
            else:
                lookup[j] -= 1

        for k in lookup:
            if lookup[k] != 0:
                return False

        return True

```

```

class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        if len(s) != len(t):
            return False

        countS, countT = {}, {}

        for i in range(len(s)):
            countS[s[i]] = 1 + countS.get(s[i], 0)
            countT[t[i]] = 1 + countT.get(t[i], 0)
        return countS == countT

```

# 1. Two Sum

🕒 Created	@July 10, 2020 6:08 AM
▼ Difficulty	Easy
≡ LC Url	<a href="https://leetcode.com/problems/two-sum/">https://leetcode.com/problems/two-sum/</a>
▼ Importance	
≡ Tag	Array&Sorting Hashmap NEET Two pointers
≡ Video	<a href="https://maxming0.github.io/2021/08/20/Two-Sum/">https://maxming0.github.io/2021/08/20/Two-Sum/</a>

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have **exactly one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

## Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

## Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

## Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

## Constraints:

- `2 <= nums.length <= 104`
- `-109 <= nums[i] <= 109`
- `-109 <= target <= 109`
- Only one valid answer exists.

## Follow-up:

Can you come up with an algorithm that is less than

$O(n^2)$

time complexity?

## Solution

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        prevMap = {} # val -> index

        for i, n in enumerate(nums):
            diff = target - n
            if diff in prevMap:
                return [prevMap[diff], i]
            prevMap[n] = i
```

### 复杂度分析

- 时间复杂度： $O(N)$ ，其中  $N$  是数组中的元素数量。对于每一个元素  $x$ ，我们可以  $O(1)$  地寻找  $target - x$ 。
- 空间复杂度： $O(N)$ ，其中  $N$  是数组中的元素数量。主要为哈希表的开销。

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        temp = nums.copy()
        temp.sort()

        start, end = 0, len(nums) - 1

        while start < end:
            if (temp[start] + temp[end]) > target:
                end -= 1
            elif (temp[start] + temp[end]) < target:
                start += 1
            else:
                break

        index1 = nums.index(temp[start])
        # nums.pop(index1)
        index2 = nums.index(temp[end])
        # if index2 >= index1:
        #     index2 += 1
        return [index1, index2]
```

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        # 2. 双指针
        # 作者: yun-yu-chen
```

```
# 链接: https://leetcode.cn/problems/two-sum/solution/san-chong-fang-fa-bao-li-shuang-zhi-zhen-ha-xi-san/

temp = nums.copy()
temp.sort()

start, end = 0, len(nums) - 1

while start < end:
    if (temp[start] + temp[end]) > target:
        end -= 1
    elif (temp[start] + temp[end]) < target:
        start += 1
    else:
        break
index1 = nums.index(temp[start])
nums.pop(index1)

index2 = nums.index(temp[end])
if index2 >= index1:
    index2 += 1
return [index1, index2]
```

# 49. Group Anagrams

🕒 Created	@July 20, 2020 3:14 AM
▼ Difficulty	Medium
≡ LC Url	<a href="https://leetcode.com/problems/group-anagrams/">https://leetcode.com/problems/group-anagrams/</a>
▼ Importance	
≡ Tag	Array&Sorting Hashmap NEET
≡ Video	字母异位词分组 - 字母异位词分组 - 力扣 (LeetCode) (leetcode-cn.com), <a href="https://maxming0.github.io/2020/04/26/Group-Anagrams/">https://maxming0.github.io/2020/04/26/Group-Anagrams/</a>

Given an array of strings `strs`, group **the anagrams** together. You can return the answer in **any order**.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

## Example 1:

```
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
```

## Example 2:

```
Input: strs = [""]
Output: [[""]]
```

## Example 3:

```
Input: strs = ["a"]
Output: [["a"]]
```

## Constraints:

- `1 <= strs.length <= 10 4`
- `0 <= strs[i].length <= 100`
- `strs[i]` consists of lowercase English letters.

## Solution



```

class Solution:
    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:
        # d = defaultdict(list)
        # for s in strs:
        #     d[''.join(sorted(s))].append(s)
        # return d.values()

        # https://leetcode-cn.com/problems/group-anagrams/solution/zi-mu-yi-wei-ci-fen-zu-by-leetcode-solut-gyoc/
        d = defaultdict(list)
        for s in strs:
            counts = [0] * 26
            for ch in s:
                counts[ord(ch) - ord('a')] += 1
            # 需要将 list 转换成 tuple 才能进行哈希
            d[tuple(counts)].append(s)
        return list(d.values())

```

# 347. Top K Frequent Elements

🕒 Created	@April 10, 2022 4:33 PM
▼ Difficulty	Medium
≡ LC Url	<a href="https://leetcode.com/problems/top-k-frequent-elements/">https://leetcode.com/problems/top-k-frequent-elements/</a>
▼ Importance	
≡ Tag	Array&Sorting NEET Queue
≡ Video	<a href="https://maxming0.github.io/2020/07/17/Top-K-Frequent-Elements/">https://maxming0.github.io/2020/07/17/Top-K-Frequent-Elements/</a>

Given an integer array `nums` and an integer `k`, return the `k` most frequent elements. You may return the answer in **any order**.

## Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

## Example 2:

```
Input: nums = [1], k = 1
Output: [1]
```

## Constraints:

- `1 <= nums.length <= 105`
- `k` is in the range `[1, the number of unique elements in the array]`.
- It is **guaranteed** that the answer is **unique**.

**Follow up:** Your algorithm's time complexity must be better than  `$O(n \log n)$` , where `n` is the array's size.


## Solution

```
class Solution:
    def topKFrequent(self, nums: List[int], k: int) -> List[int]:
        # ct = Counter(nums)
        # return heapq.nlargest(k, ct.keys(), key=ct.get)

        q = []
        for num, freq in Counter(nums).items():
            if len(q) == k:
                heapq.heappushpop(q, (freq, num))
            else:
                heapq.heappush(q, (freq, num))
        return [x[1] for x in q]
```

### Python应用--优先队列与heapq

本文始发于个人公众号：TechFlow，求个关注 今天的文章来介绍 Python当中一个蛮有用的库-- heapq。heapq的全写是heap queue，是堆队列的意思。这里的 堆和队列 都是数据结构，在后序

 <https://zhuanlan.zhihu.com/p/106170247>



# 238. Product of Array Except Self

🕒 Created	@December 26, 2021 6:53 PM
▼ Difficulty	Medium
☰ LC Url	<a href="https://leetcode.com/problems/product-of-array-except-self/">https://leetcode.com/problems/product-of-array-except-self/</a>
▼ Importance	
☰ Tag	Array&Sorting NEET
☰ Video	

Given an integer array `nums`, return an array `answer` such that `answer[i]` is equal to the product of all the elements of `nums` except `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in  $O(n)$  time and without using the division operation.

## Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

## Example 2:

```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```

## Constraints:

- $2 \leq \text{nums.length} \leq 10^5$
- $-30 \leq \text{nums}[i] \leq 30$
- The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

**Follow up:** Can you solve the problem in  $O(1)$  extra space complexity? (The output array **does not** count as extra space for space complexity analysis.)

## Solution

```
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        res = [1] * (len(nums))

        prefix = 1
        for i in range(len(nums)):
            res[i] = prefix
            prefix *= nums[i]
        postfix = 1
        for i in range(len(nums) - 1, -1, -1):
            res[i] *= postfix
            postfix *= nums[i]
        return res
```

# 242. Valid Anagram

🕒 Created	@January 29, 2022 2:31 PM
▼ Difficulty	Easy
☰ LC Url	<a href="https://leetcode.com/problems/valid-anagram/">https://leetcode.com/problems/valid-anagram/</a>
▼ Importance	
☰ Tag	Array&Sorting
☰ Video	<a href="https://www.youtube.com/watch?v=wScXoa8pN6o&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=51">https://www.youtube.com/watch?v=wScXoa8pN6o&amp;list=PL2rWx9cCzU85RX9NeRMVUV_kgI4YGKURD&amp;index=51</a>

Given two strings `s` and `t`, return `true` if `t` is an anagram of `s`, and `false` otherwise.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

## Example 1:

```
Input: s = "anagram", t = "nagaram"
Output: true
```

## Example 2:

```
Input: s = "rat", t = "car"
Output: false
```

## Constraints:

- `1 <= s.length, t.length <= 5 * 104`
- `s` and `t` consist of lowercase English letters.

**Follow up:** What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

## Solution

```

class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        '''
        时间空间复杂度都是O(n)
        '''
        if len(s) != len(t):
            return False

        lookup = {}

        for i in s:
            if i not in lookup:
                lookup[i] = 1
            else:
                lookup[i] += 1

        for j in t:
            if j not in lookup:
                return False
            else:
                lookup[j] -= 1

        for k in lookup:
            if lookup[k] != 0:
                return False

        return True

```

```

class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        if len(s) != len(t):
            return False

        countS, countT = {}, {}

        for i in range(len(s)):
            countS[s[i]] = 1 + countS.get(s[i], 0)
            countT[t[i]] = 1 + countT.get(t[i], 0)
        return countS == countT

```

# 128. Longest Consecutive Sequence

🕒 Created	@April 4, 2022 11:48 PM
▼ Difficulty	Medium
☰ LC Url	<a href="https://leetcode.com/problems/longest-consecutive-sequence/">https://leetcode.com/problems/longest-consecutive-sequence/</a>
▼ Importance	
☰ Tag	Array&Sorting Hashmap NEET
☰ Video	<a href="https://www.youtube.com/watch?v=P6RZZMu_maU">https://www.youtube.com/watch?v=P6RZZMu_maU</a>

Given an unsorted array of integers `nums`, return *the length of the longest consecutive elements sequence*.

You must write an algorithm that runs in  $O(n)$  time.

## Example 1:

```
Input: nums = [100,4,200,1,3,2]
Output: 4
Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.
```

## Example 2:

```
Input: nums = [0,3,7,2,5,8,4,6,0,1]
Output: 9
```

## Constraints:

- $0 \leq \text{nums.length} \leq 10^5$
- $-10^9 \leq \text{nums}[i] \leq 10^9$



# Solution

```
class Solution:
    def longestConsecutive(self, nums: List[int]) -> int:
        numSet = set(nums)
        longest = 0

        for n in nums:
            if (n - 1) not in numSet:
                length = 0
                while (n + length) in numSet:
                    length += 1
                longest = max(length, longest)
        return longest
```