# 双指针

双指针技巧在 Python 编程中非常有用，尤其是在处理数组、链表和字符串等问题时。以下是一些常见的双指针应用场景及其示例：

## 1. 两数之和（Two Sum）

**问题描述**：在一个有序数组中，找到两个数，使它们的和等于目标值。

**示例代码**：

```python
def two_sum(nums, target):
    left, right = 0, len(nums) - 1
    while left < right:
        current_sum = nums[left] + nums[right]
        if current_sum == target:
            return [left, right]
        elif current_sum < target:
            left += 1
        else:
            right -= 1
    return []
```

## 2. 移除元素（Remove Element）

**问题描述**：从数组中移除所有指定的值，并返回新数组的长度。

**示例代码**：

```python
def remove_element(nums, val):
    slow = 0
    for fast in range(len(nums)):
        if nums[fast] != val:
            nums[slow] = nums[fast]
            slow += 1
    return slow
```

## 3. 合并两个有序数组（Merge Two Sorted Arrays）

**问题描述**：将两个有序数组合并成一个有序数组。

**示例代码**：

```python
def merge(nums1, m, nums2, n):
    p1, p2, p = m - 1, n - 1, m + n - 1
    while p1 >= 0 and p2 >= 0:
        if nums1[p1] > nums2[p2]:
            nums1[p] = nums1[p1]
            p1 -= 1
        else:
            nums1[p] = nums2[p2]
            p2 -= 1
        p -= 1
    nums1[:p2 + 1] = nums2[:p2 + 1]
```

## 4. 回文子串（Palindrome Substrings）

**问题描述**：在字符串中找到所有回文子串。

**示例代码**：

```python
def count_substrings(s):
    def expand_around_center(left, right):
        count = 0
        while left >= 0 and right < len(s) and s[left] == s[right]:
            count += 1
            left -= 1
            right += 1
        return count

    total_count = 0
    for i in range(len(s)):
        total_count += expand_around_center(i, i)       # 奇数长度的回文
        total_count += expand_around_center(i, i + 1)   # 偶数长度的回文
    return total_count
```

## 5. 三数之和（Three Sum）

**问题描述**：在一个数组中找到所有和为零的三元组。

**示例代码**：

```python
def three_sum(nums):
    nums.sort()
    result = []
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        left, right = i + 1, len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total < 0:
                left += 1
```

```
            elif total > 0:
                right -= 1
            else:
                result.append([nums[i], nums[left], nums[right]])
                while left < right and nums[left] == nums[left + 1]:
                    left += 1
                while left < right and nums[right] == nums[right - 1]:
                    right -= 1
                left += 1
                right -= 1
    return result
```

## 6. 滑动窗口（Sliding Window）

**问题描述**：在一个数组中找到满足某些条件的最长子数组。

**示例代码**：

```
def longest_subarray_with_sum_k(nums, k):
    left = 0
    current_sum = 0
    max_length = 0
    for right in range(len(nums)):
        current_sum += nums[right]
        while current_sum > k:
            current_sum -= nums[left]
            left += 1
        if current_sum == k:
            max_length = max(max_length, right - left + 1)
    return max_length
```

## 7. 反转链表（Reverse Linked List）

**问题描述**：反转一个单链表。

**示例代码**：

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def reverse_list(head):
    prev = None
    current = head
    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node
    return prev
```

这些示例展示了双指针技巧在不同问题中的应用。通过合理使用双指针，可以有效提高算法的效率和可读性。希望这些示例对你有所帮助！如果有更多具体问题或需要进一步解释，请随时告诉我。