1.Odd String Difference

```
def findOddString(words):
    def get_difference_array(word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]
    difference_arrays = [get_difference_array(word) for word in words]
    from collections import defaultdict
    difference_count = defaultdict(int)
    for diff in difference_arrays:
        difference_count[tuple(diff)] += 1
    odd_difference = None
    for diff, count in difference_count.items():
        if count == 1:
            odd_difference = diff
            break
    for word in words:
        if get_difference_array(word) == list(odd_difference):
            return word
words = ["adc", "wzy", "abc"]
print(findOddString(words))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Odd String Difference.py
abc
>>>
```

2.Words within two Edits of Dictionary

```
def two_edits_away(queries, dictionary):
    def is_within_two_edits(word1, word2):
        count = 0
        for c1, c2 in zip(word1, word2):
            if c1 != c2:
                count += 1
                if count > 2:
                    return False
        return count <= 2
    result = []
    for query in queries:
        if any(is_within_two_edits(query, dict_word) for dict_word in dictionary):
            result.append(query)
    return result
queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
print(two_edits_away(queries, dictionary))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Words within Two Edits of Dictionary.py
['word', 'note', 'wood']
>>>
```

3.Next Greater element IV

```
def secondGreater(nums):
    n = len(nums)
    answer = [-1] * n
    first_greater_stack = []
    second_greater_stack = []
    for i, num in enumerate(nums):
        while second_greater_stack and nums[second_greater_stack[-1]] < num:
            answer[second_greater_stack.pop()] = num
        temp_stack = []
        while first_greater_stack and nums[first_greater_stack[-1]] < num:
            temp_stack.append(first_greater_stack.pop())

        while temp_stack:
            second_greater_stack.append(temp_stack.pop())
        first_greater_stack.append(i)
    return answer
nums = [2,4,0,9,6]
print(secondGreater(nums))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Next Greater Element IV.py
[9, 6, 6, -1, -1]
>>>
```

4.Minimum Addition to Make Integer Beautiful

```
def digit_sum(num):
    return sum(int(digit) for digit in str(num))
def min_addition_to_make_beautiful(n, target):
    x = 0
    while True:
        if digit_sum(n + x) <= target:
            return x
        x += 1
n = 467
target = 6
print(min_addition_to_make_beautiful(n, target))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Minimum Addition to Make Integer Beautiful.py
33
>>>
```

5.Sort Array by Moving Items to Empty Space

```
def min_operations_to_sort(nums):
    n = len(nums)
    target1 = list(range(n))
    target2 = list(range(1, n)) + [0]
    def count_operations(arr, target):
        arr = arr[:]
        position = {num: i for i, num in enumerate(arr)}
        operations = 0
        for i in range(n):
            if arr[i] != target[i]:
                target_index = position[target[i]]
                arr[i], arr[target_index] = arr[target_index], arr[i]
                position[arr[target_index]] = target_index
                position[arr[i]] = i
                operations += 1
        return operations
    return min(count_operations(nums, target1), count_operations(nums, target2))
nums = [1,2,3,4,0]
print(min_operations_to_sort(nums))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Sort Array By Moving Items to Empty Space.py
0
>>>
```