Problem 1: Odd String Difference

```
def oddString(words):
    def get_difference(word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]

    differences = [get_difference(word) for word in words]
    for i in range(len(differences)):
        if differences.count(differences[i]) == 1:
            return words[i]

words = ["adc","wzy","abc"]
print(oddString(words))
```

Output: "abc"

Time complexity :- O(m2·n).

Problem 2: Words Within Two Edits of Dictionary

```
def withinTwoEdits(queries, dictionary):
    def is_within_two_edits(query, word):
        if len(query) != len(word):
            return False
        diff_count = sum(1 for a, b in zip(query, word) if a != b)
        return diff_count <= 2
     result = []
    for query in queries:
        if any(is_within_two_edits(query, word) for word in dictionary):
            result.append(query)

    return result
queries = ["word","note","ants","wood"]
dictionary = ["wood","joke","moat"]
print(withinTwoEdits(queries, dictionary))
```

Output: ["word","note","wood"]

Time complexity :- O(q·d·n)

Problem 3: Next Greater Element IV

```
def nextGreaterElementIV(nums):
    n = len(nums)
    res = [-1] * n
    stack1, stack2 = [], []
    for i in range(n):
        while stack2 and nums[stack2[-1]] < nums[i]:
            res[stack2.pop()] = nums[i]
        while stack1 and nums[stack1[-1]] < nums[i]:
            stack2.append(stack1.pop())
        stack1.append(i)

    return res
nums = [2,4,0,9,6]
print(nextGreaterElementIV(nums))
```

Output: [9,6,6,-1,-1]

Time complexity :- O(n)

Problem 4: Minimum Addition to Make Integer Beautiful

```
def makeIntegerBeautiful(n, target):
    def digit_sum(x):
        return sum(int(digit) for digit in str(x))

    x = 0
    while digit_sum(n + x) > target:
        x += 1
   return x
n = 16
target = 6
print(makeIntegerBeautiful(n, target))
```

Output: 4

Time complexity :- O(n·log n)

Problem 5: Sort Array by Moving Items to Empty Space

```
def sortByMovingToEmpty(nums):
    n = len(nums)
    sorted_nums = sorted(nums)
    sorted_positions = {val: idx for idx, val in enumerate(sorted_nums)}

    moves = 0
    zero_index = nums.index(0)

    for i in range(n):
        while nums[i] != sorted_nums[i]:
            target_index = sorted_positions[nums[i]]
            nums[zero_index], nums[target_index] = nums[target_index], nums[zero_index]
            zero_index = target_index
            moves += 1

    return moves

nums = [4,2,0,3,1]
print(sortByMovingToEmpty(nums))
```

Output: 3

Time complexity :- O(nlogn).