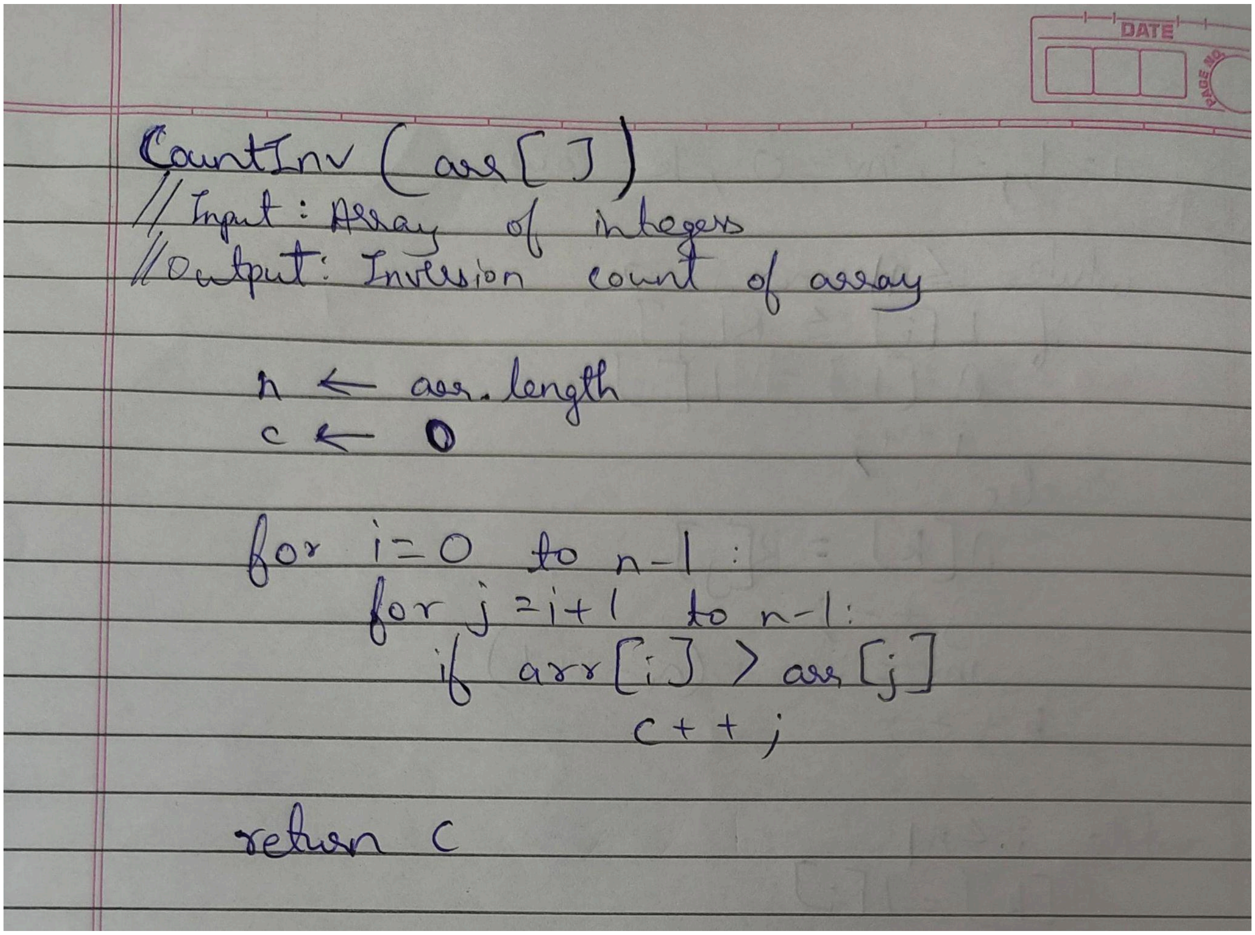
DAA Assignment 4

Ishaan Shaikh
231070063
SY CE

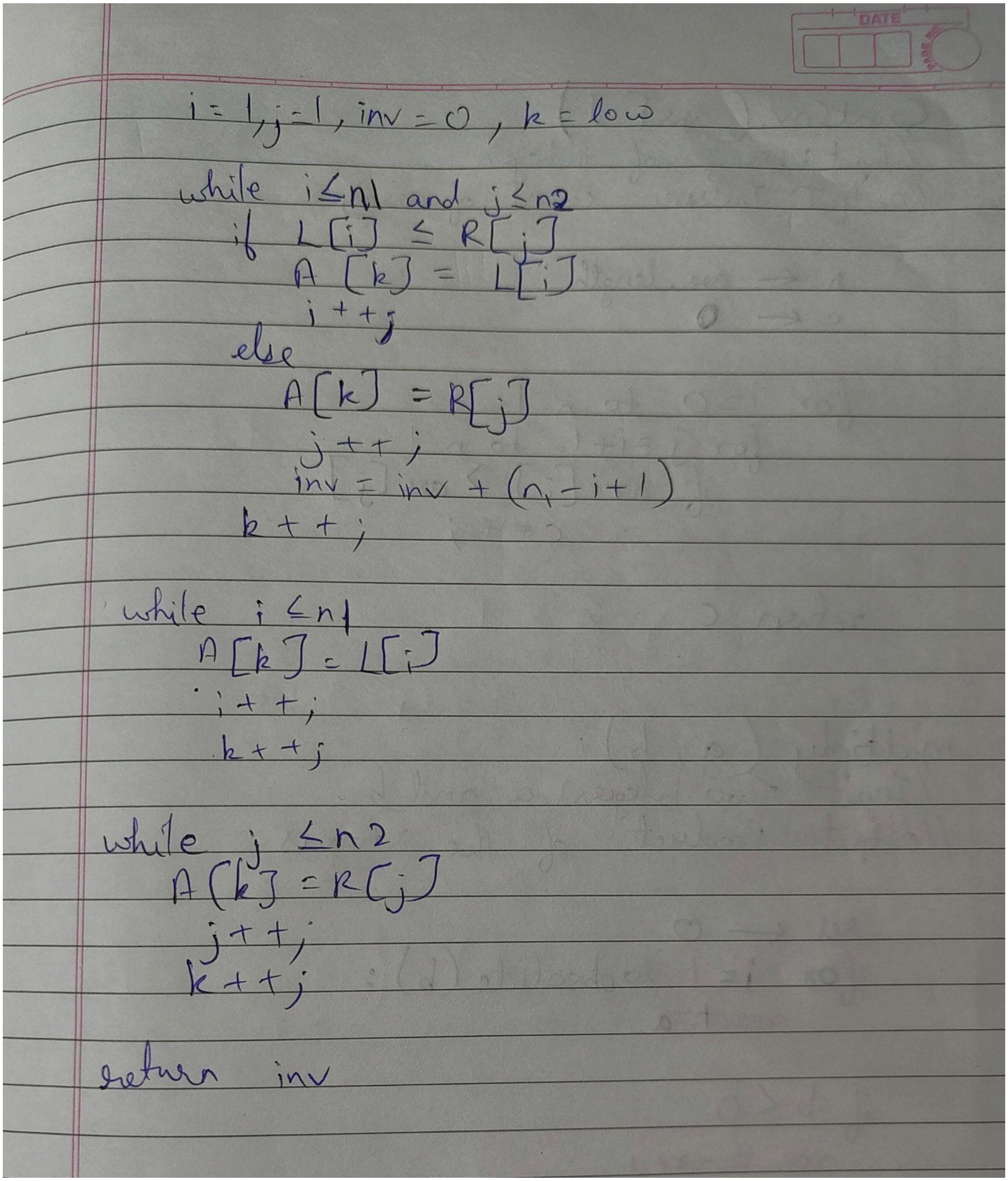
Q. Consider first/second year course-code choices of 100 students.
Find the inversion count of these choices.
Find students with zero, one, two, three inversion counts and comment on your result.

Algorithm:



```
CountInv (arr[ ])
//Input: Array of integers
//Output: Inversion count of array

    n ← arr.length
    c ← 0

    for i=0 to n-1 :
        for j =i+1 to n-1:
            if arr[i] > arr[j]
                c++;

    return c
```

```
CountInv (arr [], l, r):
// Input : Array of integers and left and right of array
// Output : Inversion count of array

    if arr.length <= 1 or arr.length == 0
        return 0

    count = 0
    mid = (l+r)//2

    LeftInv = CountInv(arr, l, mid);
    RightInv = CountInv(arr, mid+1, r);
    SplitInv = CountMergeInv(arr, l, mid, r)

    return LeftInv + RightInv + SplitInv

CountMergeInv (arr [], l, mid, r):
// Input : Array of integers, left and right and middle position
// Output : Inversion count while merging.

    n1 = mid - low + 1
    n2 = high - mid

    L[1 ... $n_1$] = [ ]
    R[1 ... $n_2$] = [ ]

    for i=1 to n1
        L[i] = A[low + i - 1]

    for j=1 to n2
        R[j] = A[mid + j]
```



```
i = 1, j = 1, inv = 0 , k = low

while i ≤ n1 and j ≤ n2
    if L[i] ≤ R[j]
        A [k] = L[i]
        i ++;
    else
        A[k] = R[j]
        j ++;
        inv = inv + (n1 - i + 1)
    k ++;

while i ≤ n1
    A [k] = L[i]
    i ++;
    k ++;

while j ≤ n2
    A [k] = R[j]
    j ++;
    k ++;

return inv
```

Test cases:

Test cases:

1) Input: [23491, 23571, 23497, 23321, 23499,
23731, 23892, 23554, 23901, 23956]
Output: Numbers of inversions : 8

2) Input: [23231, 23321, 23345, 23452, 23552,
23567, 23681, 23790, 23888, 23991]
Output: Numbers of inversions : 0

3) Input: [23590, 23791, 23214, 23413, 23521,
23771, 23839, 23415, 23115, 23557]
Output: Number of inversions : 24

4) Input: [23390, 23591, 23431]

Output: The array must have atleast 10 elements

5) Input: [ ]

Output: The array is empty.

6) Input: [23590, 23791, 23214, 23413, 23521, 23771, 23839,
23415, 23115, "23557"]
Output: All elements of the array must be numbers

Time Complexity:

Time complexity:

Linear counting inversion:

Input: array of size n

Output: Inversion count

Basic operation: check whether elements are greater to the right

Let Cworst(n) denote the time required to execute the for loops

$$\therefore C_{worst}(n) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} n-\cancel{1}-(j+1)+\cancel{1}$$

$$= \sum_{i=0}^{n-1} (n-i-1)$$

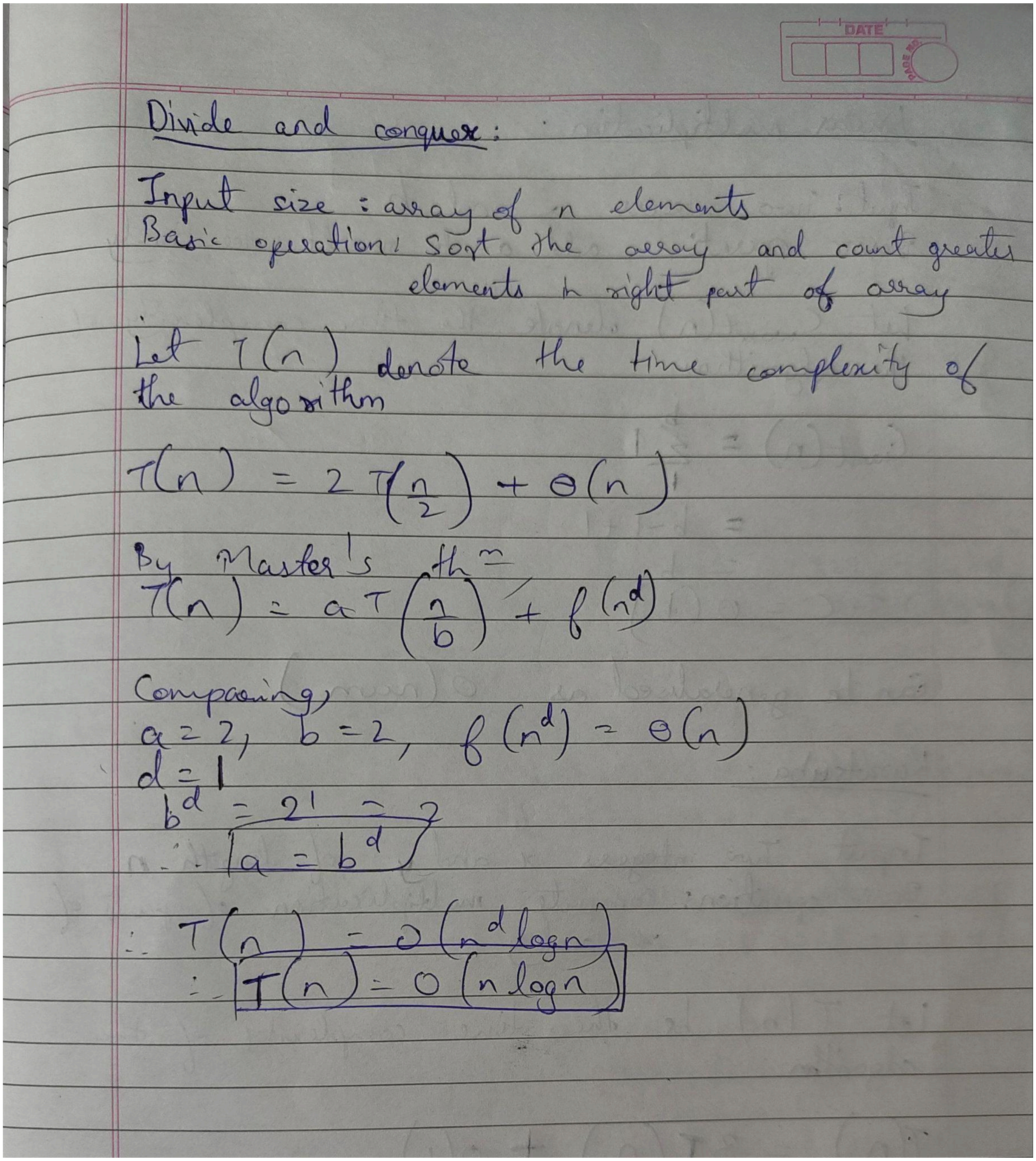
$$= n\sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i - \sum_{i=0}^{n-1} 1$$

$$= n(n-\cancel{1}+\cancel{1}) - (0+1+2 \dots n-\cancel{1}+\cancel{1}) - (n-1+1$$

$$= n \times n - \frac{n(n+1)}{2} - n$$

$$= n^2 - \frac{n(n+1)}{2} - n$$

$$\therefore \boxed{C_{worst}(n) = O(n^2)}$$



Divide and conquer:

Input size : array of n elements
Basic operation: Sort the array and count greater elements in right part of array

Let T(n) denote the time complexity of the algorithm

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

By Master's th^m

$$T(n) = aT\left(\frac{n}{b}\right) + f(n^d)$$

Comparing,

$a = 2$, $b = 2$, $f(n^d) = \Theta(n)$

$d = 1$

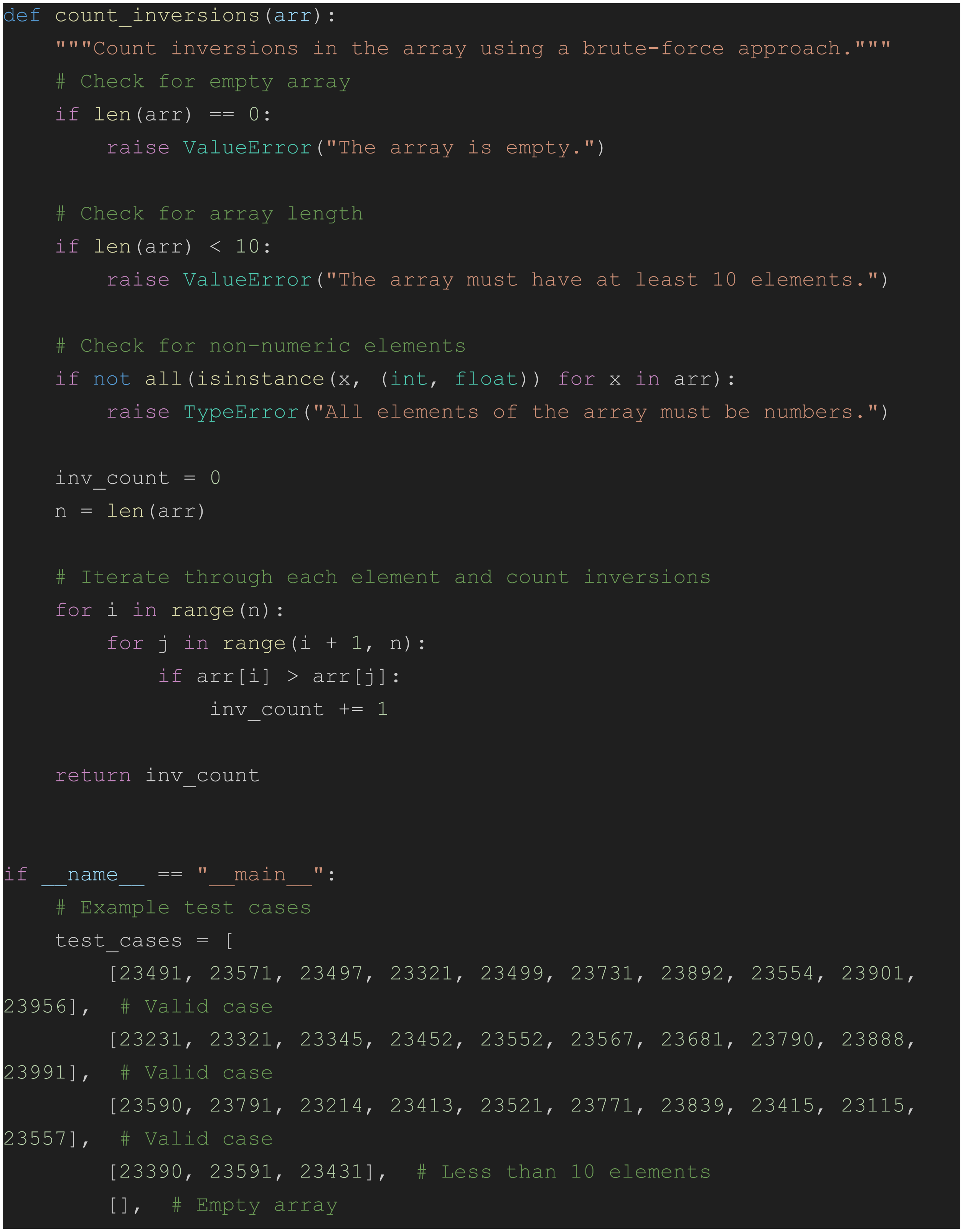
$b^d = 2^1 = 2$

$\therefore \boxed{a = b^d}$

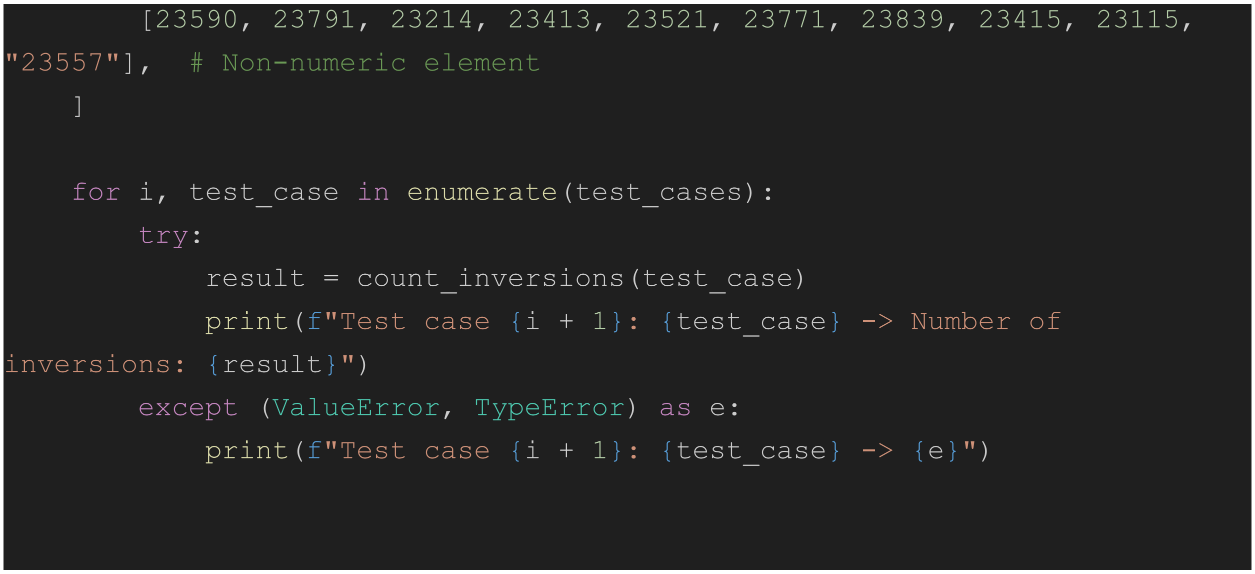
$\therefore T(n) = O(n^d \log n)$

$\therefore \boxed{T(n) = O(n \log n)}$

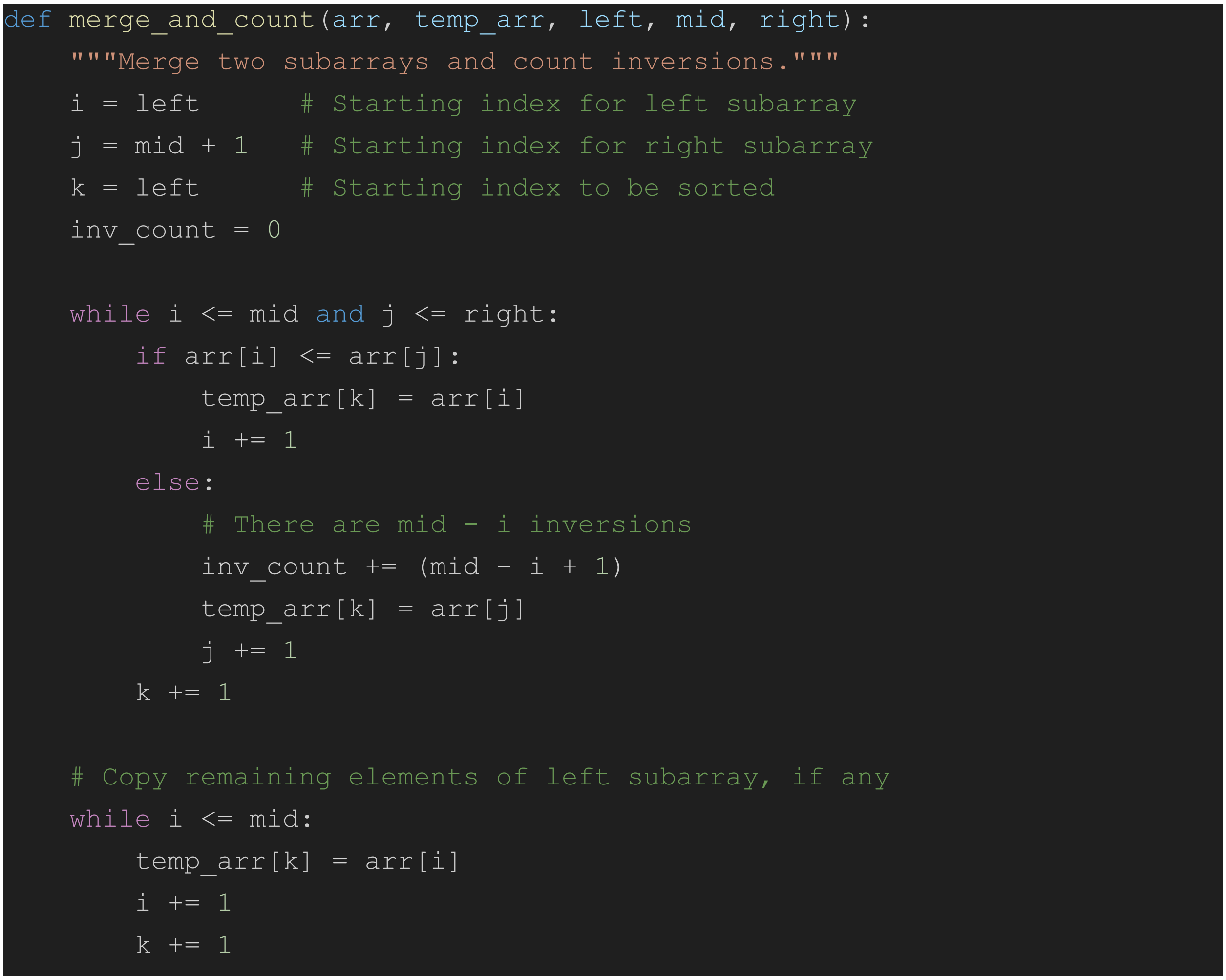
Program: **PEP 08 Coding style for python is used**



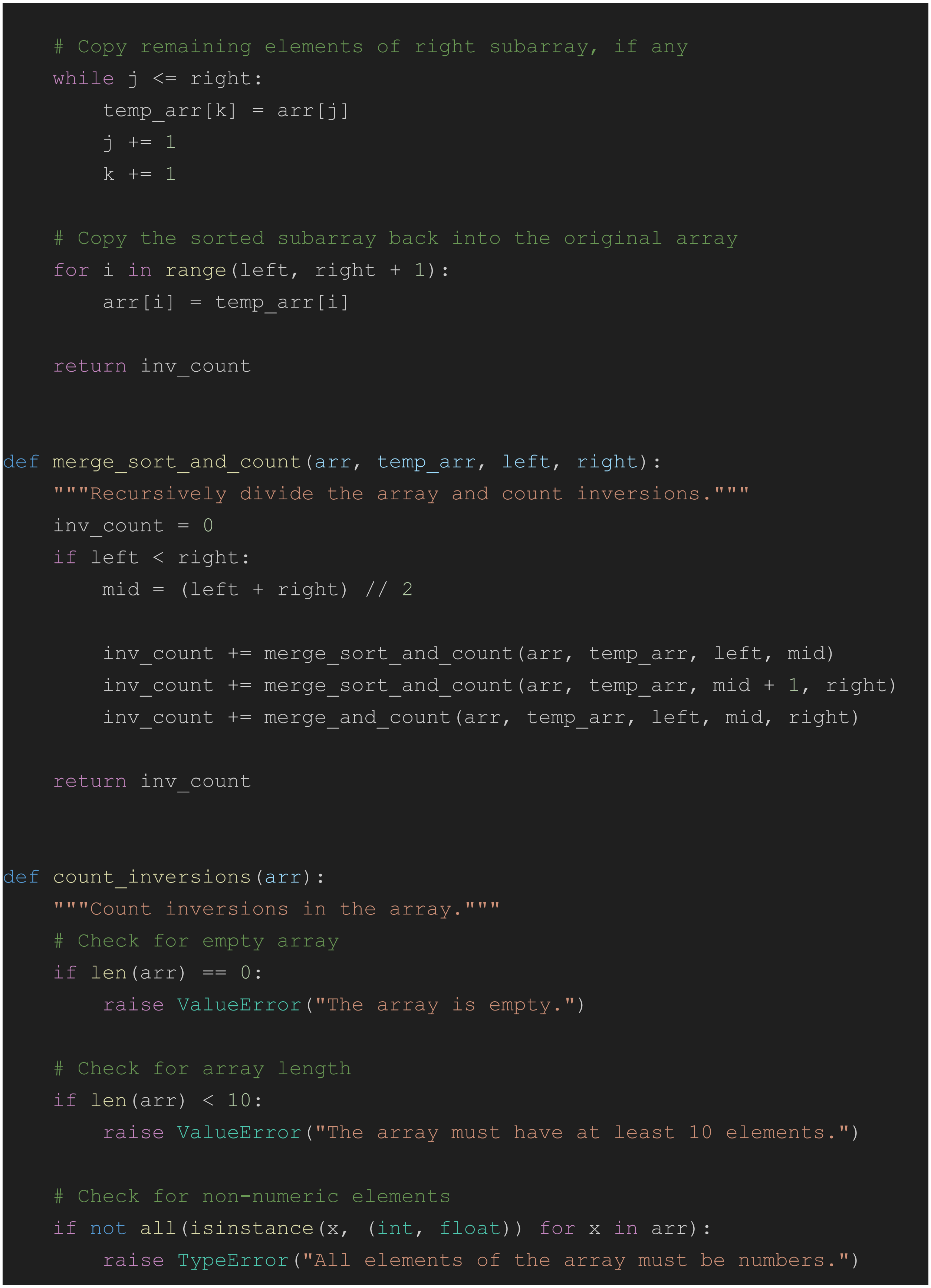
```python
def count_inversions(arr):
    """Count inversions in the array using a brute-force approach."""
    # Check for empty array
    if len(arr) == 0:
        raise ValueError("The array is empty.")

    # Check for array length
    if len(arr) < 10:
        raise ValueError("The array must have at least 10 elements.")

    # Check for non-numeric elements
    if not all(isinstance(x, (int, float)) for x in arr):
        raise TypeError("All elements of the array must be numbers.")

    inv_count = 0
    n = len(arr)

    # Iterate through each element and count inversions
    for i in range(n):
        for j in range(i + 1, n):
            if arr[i] > arr[j]:
                inv_count += 1

    return inv_count


if __name__ == "__main__":
    # Example test cases
    test_cases = [
        [23491, 23571, 23497, 23321, 23499, 23731, 23892, 23554, 23901,
23956],  # Valid case
        [23231, 23321, 23345, 23452, 23552, 23567, 23681, 23790, 23888,
23991],  # Valid case
        [23590, 23791, 23214, 23413, 23521, 23771, 23839, 23415, 23115,
23557],  # Valid case
        [23390, 23591, 23431],  # Less than 10 elements
        [],  # Empty array
```



```
        [23590, 23791, 23214, 23413, 23521, 23771, 23839, 23415, 23115,
"23557"],  # Non-numeric element
    ]

    for i, test_case in enumerate(test_cases):
        try:
            result = count_inversions(test_case)
            print(f"Test case {i + 1}: {test_case} -> Number of
inversions: {result}")
        except (ValueError, TypeError) as e:
            print(f"Test case {i + 1}: {test_case} -> {e}")
```

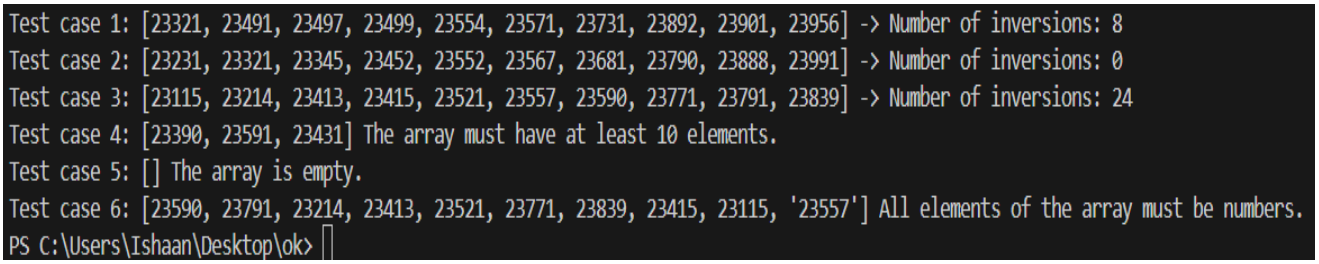


```
def merge_and_count(arr, temp_arr, left, mid, right):
    """Merge two subarrays and count inversions."""
    i = left      # Starting index for left subarray
    j = mid + 1   # Starting index for right subarray
    k = left      # Starting index to be sorted
    inv_count = 0

    while i <= mid and j <= right:
        if arr[i] <= arr[j]:
            temp_arr[k] = arr[i]
            i += 1
        else:
            # There are mid - i inversions
            inv_count += (mid - i + 1)
            temp_arr[k] = arr[j]
            j += 1
        k += 1

    # Copy remaining elements of left subarray, if any
    while i <= mid:
        temp_arr[k] = arr[i]
        i += 1
        k += 1
```



```
    # Copy remaining elements of right subarray, if any
    while j <= right:
        temp_arr[k] = arr[j]
        j += 1
        k += 1

    # Copy the sorted subarray back into the original array
    for i in range(left, right + 1):
        arr[i] = temp_arr[i]

    return inv_count


def merge_sort_and_count(arr, temp_arr, left, right):
    """Recursively divide the array and count inversions."""
    inv_count = 0
    if left < right:
        mid = (left + right) // 2

        inv_count += merge_sort_and_count(arr, temp_arr, left, mid)
        inv_count += merge_sort_and_count(arr, temp_arr, mid + 1, right)
        inv_count += merge_and_count(arr, temp_arr, left, mid, right)

    return inv_count


def count_inversions(arr):
    """Count inversions in the array."""
    # Check for empty array
    if len(arr) == 0:
        raise ValueError("The array is empty.")

    # Check for array length
    if len(arr) < 10:
        raise ValueError("The array must have at least 10 elements.")

    # Check for non-numeric elements
    if not all(isinstance(x, (int, float)) for x in arr):
        raise TypeError("All elements of the array must be numbers.")
    temp_arr = [0] * len(arr)
    return merge_sort_and_count(arr, temp_arr, 0, len(arr) - 1)


if __name__ == "__main__":
    test_cases = [
        [23491, 23571, 23497, 23321, 23499, 23731, 23892, 23554, 23901,
23956],  # Valid case
        [23231, 23321, 23345, 23452, 23552, 23567, 23681, 23790, 23888,
23991],  # Valid case
        [23590, 23791, 23214, 23413, 23521, 23771, 23839, 23415, 23115,
23557],  # Valid case
        [23390, 23591, 23431],  # Less than 10 elements
        [],  # Empty array
        [23590, 23791, 23214, 23413, 23521, 23771, 23839, 23415, 23115,
"23557"],  # Non-numeric element
    ]

    for i, test_case in enumerate(test_cases):
        try:
            result = count_inversions(test_case)
            print(f"Test case {i + 1}: {test_case} -> Number of
inversions: {result}")
        except (ValueError, TypeError) as e:
            print(f"Test case {i + 1}: {test_case} {e}")
```

Output:



```
Test case 1: [23321, 23491, 23497, 23499, 23554, 23571, 23731, 23892, 23901, 23956] -> Number of inversions: 8
Test case 2: [23231, 23321, 23345, 23452, 23552, 23567, 23681, 23790, 23888, 23991] -> Number of inversions: 0
Test case 3: [23115, 23214, 23413, 23415, 23521, 23557, 23590, 23771, 23791, 23839] -> Number of inversions: 24
Test case 4: [23390, 23591, 23431] The array must have at least 10 elements.
Test case 5: [] The array is empty.
Test case 6: [23590, 23791, 23214, 23413, 23521, 23771, 23839, 23415, 23115, '23557'] All elements of the array must be numbers.
PS C:\Users\Ishaan\Desktop\ok>
```

Conclusion: Hence, we have studied the program to count the inversions in a given array. We have implemented the program using both linear and divide and conquer algorithms. Divide and conquer is implemented using merge sort which is more efficient in this case.