Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam - 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)

UCS2403: DESIGN & ANALYSIS OF ALGORITHMS

Assignment 3

1. Given a list L of n numbers, an inversion is defined as a pair $(L[i], L[j])$ such that $i < j$ and $L[i] > L[j]$. For example, if $L = [3, 2, 8, 1]$, then $(3, 2), (8, 1), (2, 1), (3, 1)$ are the inversions in L. Consider the Python codes given in (1) and (2) below for finding the count of inversions in a list.

```
(1) def count_inversions1(nums):
        count = 0
        for i in range(1, len(nums)):
            if nums[i] < nums[i - 1]:
                count += 1
        return count

(2) def count_inversions2(nums):
        nums.sort()
        count = 0
        for i in range(1, len(nums)):
            if nums[i] < nums[i - 1]:
                count += 1
        return count
```

   Find if there are errors in these codes. If there are, find one counterexample for each incorrect code, fix the errors, and write your own (correct) code to find the count of inversions in a list. Derive the time complexity of your final algorithm(s). Note that a counterexample is an input instance to the algorithm that produces a wrong output.

2. Given a list of integers, the comparison count sorting algorithm sorts the list as follows: For each integer at index i in the list, count the number of integers that are strictly less than it. In the sorted list, place the integer at the index equal to the number of integers that are less than it. For example, there are no integers less than the minimum integer in the list, so the minimum integer is placed at index 0. Now, consider the code given below to sort a list of numbers using comparison count sort.

```
def comparison_count_sort(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
        for j in range(i + 1, len(nums)):
            if nums[i] > nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
        nums_sorted[count[i]] = nums[i]
    return nums_sorted
```

   Find if there are errors in this code. If there are, find at least one counterexample, list the lines of code that have errors, and write your own (correct) code to implement comparison count sort. Derive the time complexity of your algorithm.