COMP3121 Algorithms and Programming Tech 2022T3 assignment 1
Chen En (z5335039)

Question 3 Counting Inversions Between Arrays
[20 marks] Let A and B be two arrays of length n, each containing a random permutation of the numbers from 1 to n. An inversion between the two permutations A and B is a pair of values (x, y) where the index of x is less than the index of y in array A, but the index of x is more than the index of y in array B.
Design an algorithm which counts the total number of inversions between A and B that runs in O(n log n) time.

To find the number of inversions, firstly we create an index array of B: B_index[n]. (i.e. B_index[B[i]]=1)

Our goal is to find the number of inversions between A and B. An inversion between A and B is a pair {A[i], A[j]} ($0\leq i < j < n$) such that B_index[A[i]]> B_index[A[j]].

Check the constraint given by the question, if (x, y) A_index[x] < A_index[y] && B_index[x] > B_index[y] we increment variable n.

To find the number of inversions, we can apply merge sort to this problem. We keep splitting array A into two subarrays until there is only single element in each array. Then start the merging and sorting part, we sort the subarray by B_index using two pointers p1, p2. (p1 points to the first element in the left subarray and p2 to the first element in the right subarray.)

That is, if B_index[i] is smaller than B_index[j] then we put i before j, and n increment m-i+1. (m is the last element in the left subarray) Then p1 points to the next element in the left array, p2 remains points to the same element.

Otherwise, if the condition returns false, p2 points to the next element in the right array and variable n remains the same. (There is no inversion pairs so we check the next two pair)

We repeat the above procedure of merging and sorting until both pointers p1 and p2 reach the last element in both arrays. In the end we return the variable n.
Time complexity

Since we are applying merge sort, the time complexity is O(nlogn). (Splitting to subarray takes O(logn) and we traverse all n elements while merging which takes O(n).)