Qiyao Zhou

Z5379852

Question 3

According to the question, one can ignore the actual size rules of the numbers in the original array and instead use the array A and the sorted array X = [1, 2, ... n] by index mapping, while converting the elements of B by the resulting mapping, so that the original problem is converted into a problem of finding the number of reversals in an array, after which the elements of B can be traversed and a dichotomous search performed in array X to obtain total number of inversions.

The specific steps are as follows:

1. Create a sorted array X = [1, 2, ... n], and then create a hash map that maps the array A to the array X by index, with the keys in the hash map being the elements in A and the values being the elements in X. The total number of inversions is recorded as 0.
2. Transform the array B according to the hash map obtained in step 1 and the resulting array is called the array R.
3. Iterate through the array R in index order, with each iterated element called r_i. Perform a binary lookup in the array X based on r_i, adding the index of each lookup to the total number of inversions (the index of any lower-valued element appearing in the array R after r_i will be j > i), and then delete the element in X.
4. When the traversal is complete, the total number of inversions is obtained.

In terms of time complexity, steps 1 and 2 each require O(n), step 3 requires n loops, each binary lookup requires O(logn), value accumulation and element deletion each require 1, so the time complexity of step 3 is 2O(n) + nO(logn), so the total time complexity is 4O(n) + nO(logn) = O(4n + nlogn) = O(nlogn)