2021.03.08 Chi-Wen Teng 鄧祺文

CLOSURE REPORT FOR ALGORITHM LAB WEEK 1：SORT

TIME COMPLEXITY

Best Case Analysis

For the insertion sort, it executes two operations in the whole sorting process. First, the method scans through the list which is waiting to sort, comparing every element pair by pair. And if the values of the elements are out of order, the method will swap the element to the right position. By counting any step in the sorting process, the best case, which the waiting list is already in sorted order, insertion sort only compares ‘n’ elements in the list and does no swapping. Therefore, the time complexity of it is O(n).

Worst Case Analysis

The worst-case for insertion sort happens when the elements of the waiting list are in decreasing order. To insert the last element to the right position, we will need to compare ‘n - 1’ times and swap ‘n – 1’ times. Next, to insert the second last element to its right position, we will need to compare it for ‘n – 2’ times and swap ‘n – 2’ times in the process. With the sorting process keeps going on, we will find the total steps to be

$$(n - 1) + (n - 1) + (n - 2) + (n - 2) + \dots$$

$$= 2 * [(n - 1) + (n - 2) + (n - 3) + \dots]$$

$$= n * (n - 1) = O(n^2)$$

Average Case Analysis

In the insertion sort, once we perform a swap, the number of inversions in the non-ordered list decreases exactly one because only the adjacent pair has disappeared. That is since no inversions are left in the array, no more swaps are needed. Therefore, the number of swaps is equal to the number of inversions.

After the precondition given above, we now make some assumptions. Since the insertion sort is mainly about comparing, only relative orders within the elements matter. For intuitive observation, we assume the elements to be all unique.

Using probability, in an average situation, given a random input list with no duplicates, in the list ‘A’, index ‘i’ < index ‘j’, there will be half of the time A[i] is less than A[j] and half of the time A[i] is bigger than A[j] which are inversions. Consequently, the probability that there’s an inversion between A[i] and A[j] is 1 / 2.

So on expectation, the runtime will be $O(n^2)$ on the average case.

$$[n * (n - 1) / 2] * (1 / 2) = n * (n - 1) / 4 \sim n^2$$