**7th Offline**

**Machine Configuration**

**Processor:** Intel(R) Core (TM) i5-7200U CPU @2.50 GHz 2.71GHz

**Installed memory (RAM)**: 4 GB (3.89 GB usable)

**Operating system:** 64 bit Windows 10 pro

**Complexity analysis**

**Merge sort**

Merge Sort follows the [Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer-introduction/) algorithm. It divides the input array in two halves and then calls itself for the two halves and finally merges the two sorted halves. Most of the implementations produce a [stable sort](https://en.wikipedia.org/wiki/Sorting_algorithm#Stability), which means that the order of equal elements is the same in the input and output.

While sorting n objects, merge sort has an best, [average](https://en.wikipedia.org/wiki/Average_performance) and [worst-case performance](https://en.wikipedia.org/wiki/Worst-case_performance) of  [O](https://en.wikipedia.org/wiki/Big_O_notation) (n log n). If the running time of merge sort for a list of length n is T(n), then the recurrence T(n) = 2T(n/2) + O (n) follows from the definition of the algorithm (applying the algorithm to divide the list into two halves and add the n steps taken to merge the resulting two lists). Its division part has O (1) time complexity and combination part has O(n) time complexity. So by solving the recurrence relation, T(n) = O (nlogn).

The complexity is same for all cases such as best, worst and average case.

Time Complexity: O (nlogn)

**Quick sort**

Quick Sort is a Divide and Conquer algorithm. In this sorting an element is picked as pivot and then partitions the given array around the picked pivot. The main process of Quick Sort is partition. Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. And then x is placed in its correct order. All this should be done in linear time. Quicksort can operate [in-place](https://en.wikipedia.org/wiki/In-place_algorithm) on an array, requiring small additional amounts of [memory](https://en.wikipedia.org/wiki/Main_memory) to perform the sorting.

While sorting pivot (an element of the array) is picked. Then partition occurs and finally [recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)) apply the above step to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

For the best and average case for n number of elements T (n) = O (nlogn) but for wost case it is O (n2)

Time Complexity: O (nlogn) -> best and average case and O (n2) -> worst case

**Data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **n =** | 10 | 100 | 1000 | 10000 | 100000 | 1000000 |
| **Case** | **Sort** |
| Sorted  (time in nanoseconds) | merge | 748.88 | 5752.62 | 77103.9 | 889960 |  |  |
| quick | 399.46 | 30958.2 | 2.86336\*106 | 3.00652\*106 |  |  |
| Reversed  (time in nanoseconds) | merge | 519.12 | 6631.21 | 75786.9 | 890330 |  |  |
| quick | 239.7 | 21530.7 | 2.08976\*106 | 2.12846\*108 |  |  |
| Random  (time in nanoseconds) | merge | 479.74 | 8287.74 | 98122.3 | 1.51853\*106 | 2.7954\*107 |  |
| quick | 519.29 | 6911.02 | 95463.7 | 1.35799\*106 | 6.7904\*107 |  |