Algorithms and Data Structures Assignment 7

Yen Ling Wong

[y.wong@jacobs-university.de](mailto:y.wong@jacobs-university.de)

Problem 7.1

Implement 3 versions of the Quicksort algorithm with 3 different versions of the partition algorithm:

1. Lomoto partition as on the lecture slides

Program is included in the file a7\_p1a.cpp

1. Hoare partition (use Wikipedia or other sources).

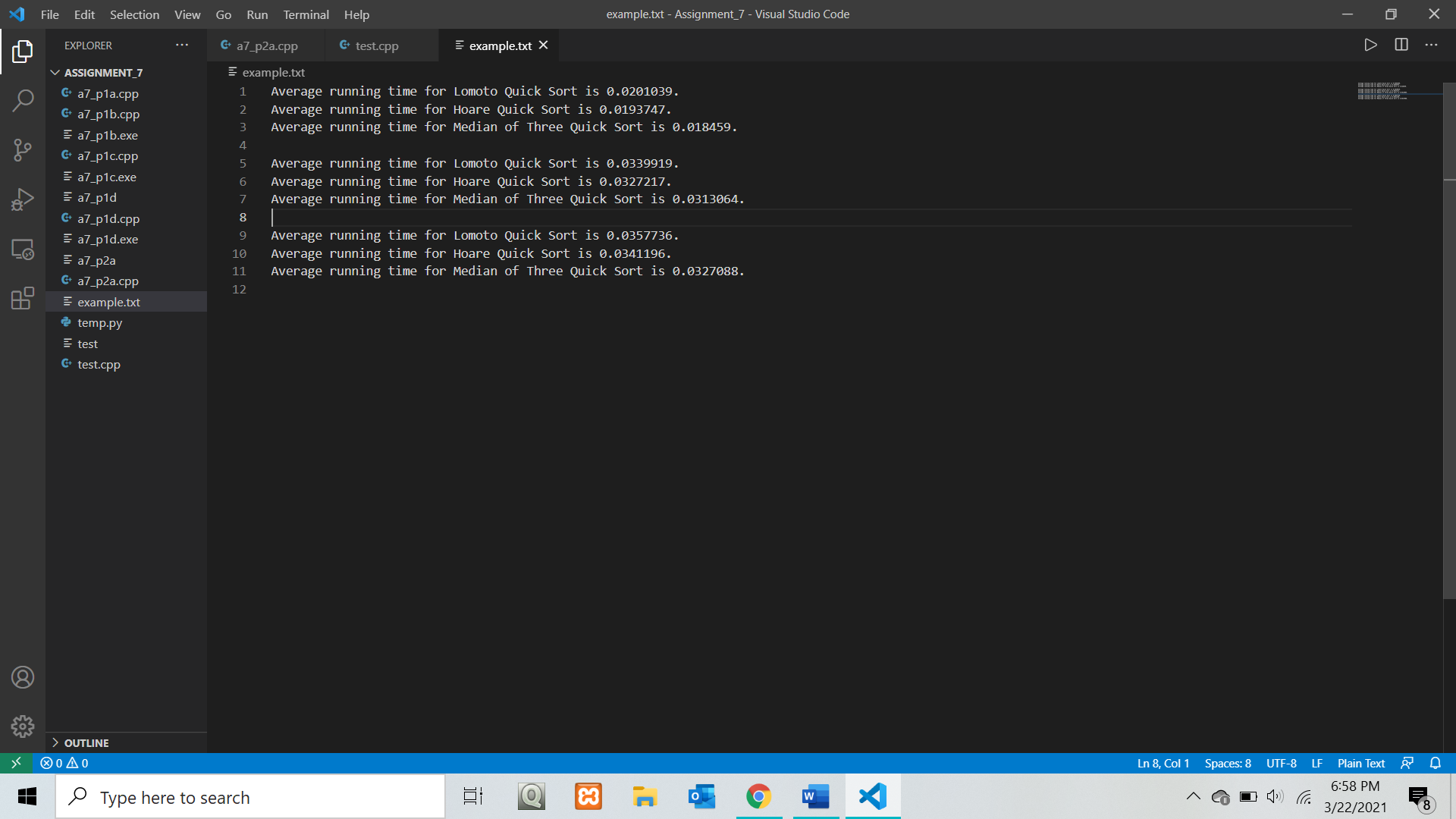
Program is included in the file a7\_p1b.cpp.

1. ”Median-of-three” partition (use Wikipedia or other sources).

Program is included in the file a7\_p1c.cpp.

1. Measure the running times of the 3 Quicksort versions from above for the same 100000 randomly generated sequences of fixed length 1000, compute the average running times for each of the 3 versions and compare them. Explain the behaviour of the 3 versions and your observations.

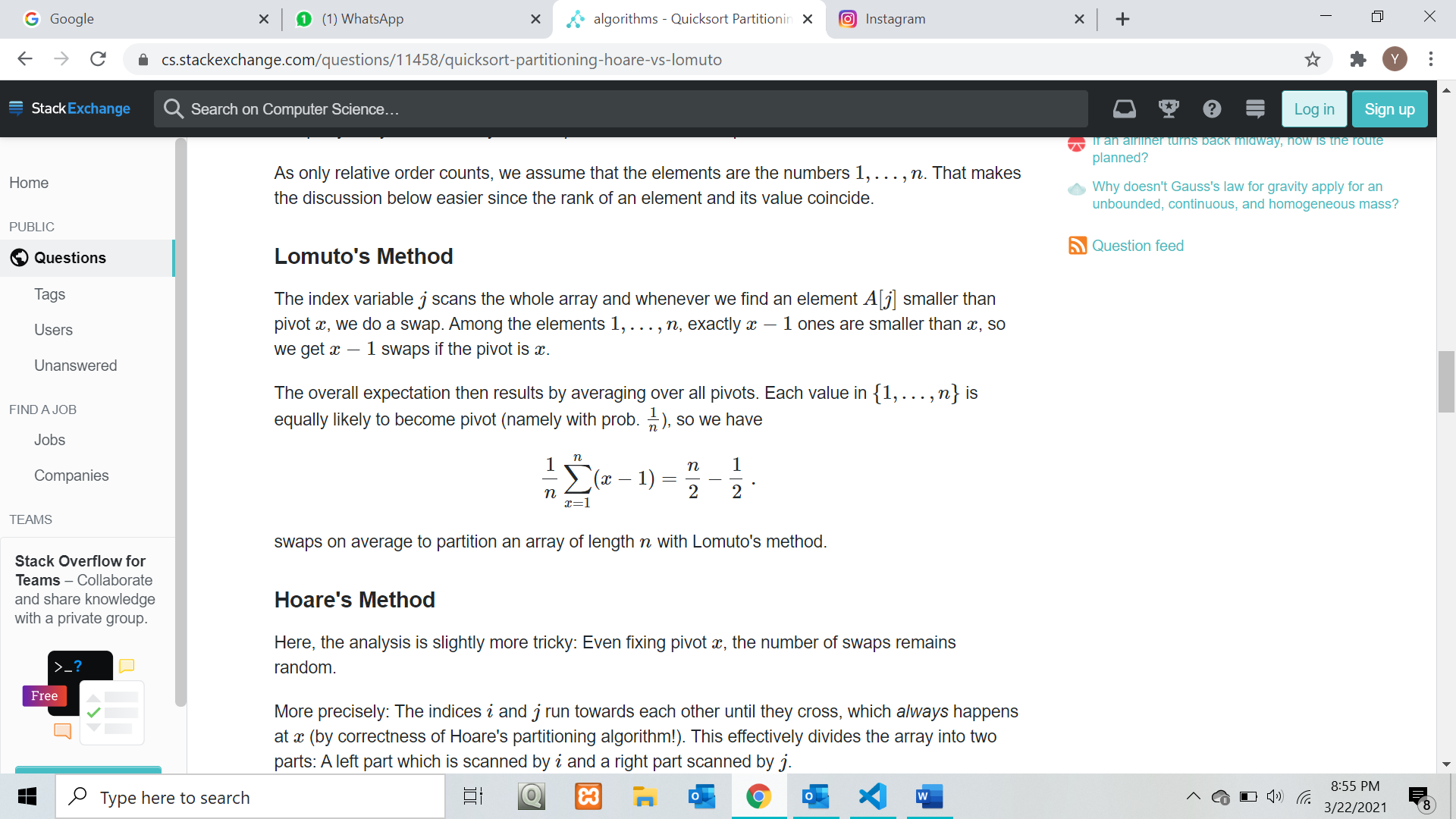
Program is included in the file a7\_p1d.cpp, where the results are written into a file called example.txt. The results:



Reference: <https://stackoverflow.com/questions/56624536/hoares-vs-lomutos-partition>

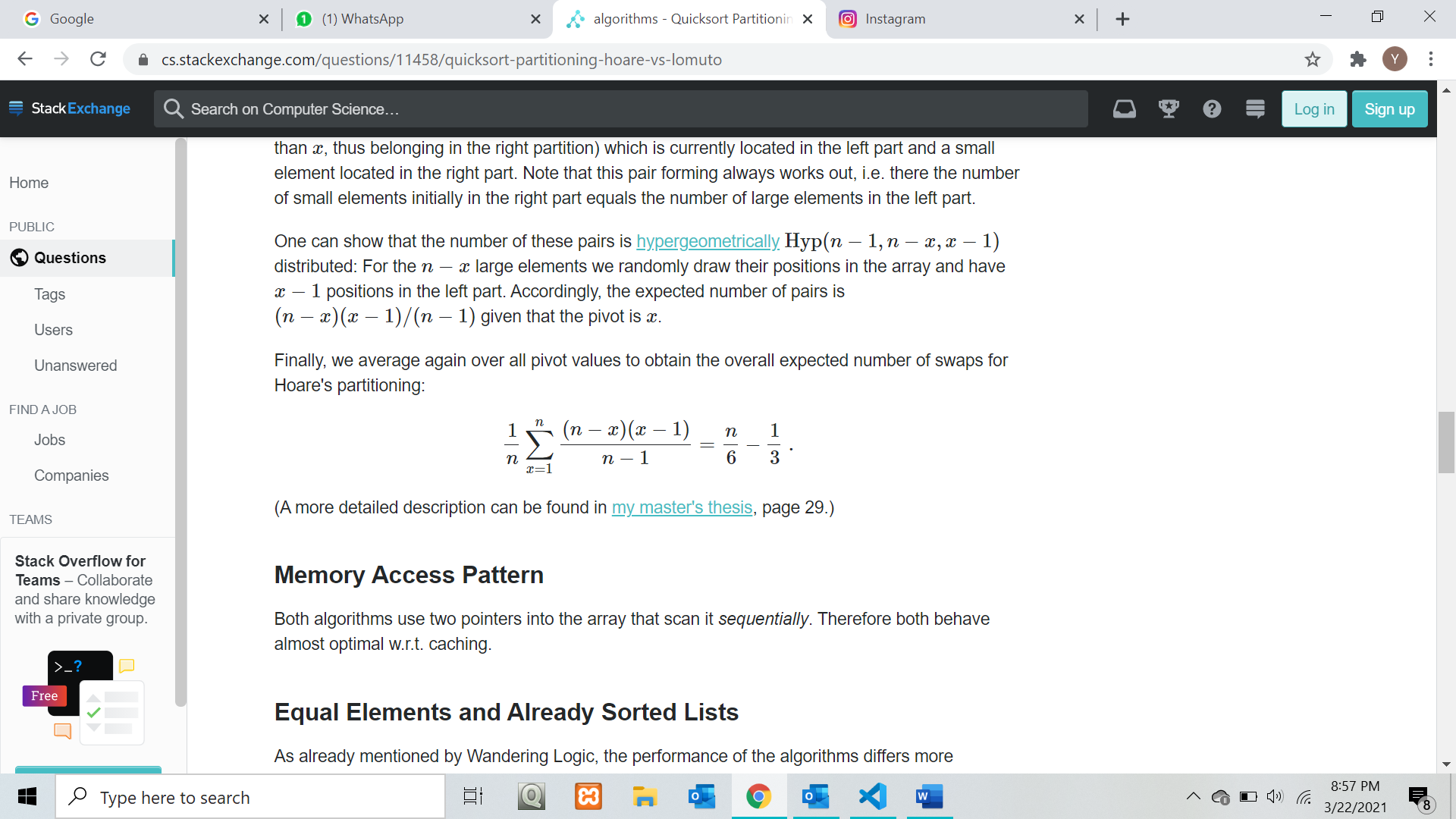
<https://stackoverflow.com/questions/7559608/median-of-three-values-strategy>

According to the results, Lomoto is the least efficient, since it takes the longest time, while Median of Three is the fastest. Lomoto is slower compared to Hoare because it takes 3 times the number of swaps, and Hoare produces efficient partitions when all values are equal. Both methods can be implemented using n – 1 comparisons to partition an array of length n. For the Lomoto partition, for every element j that is smaller than the pivot k, a swap is performed. Considering an array of length n with distinct elements, exactly k – 1 elements are smaller than x and hence k – 1 swaps are performed. On average, each element has an equal chance of becoming a pivot, so the number of swaps on average for Lomoto’s partition is



For the comparison A[j] <= x, it will always be true, so a swap is performed for every single element.

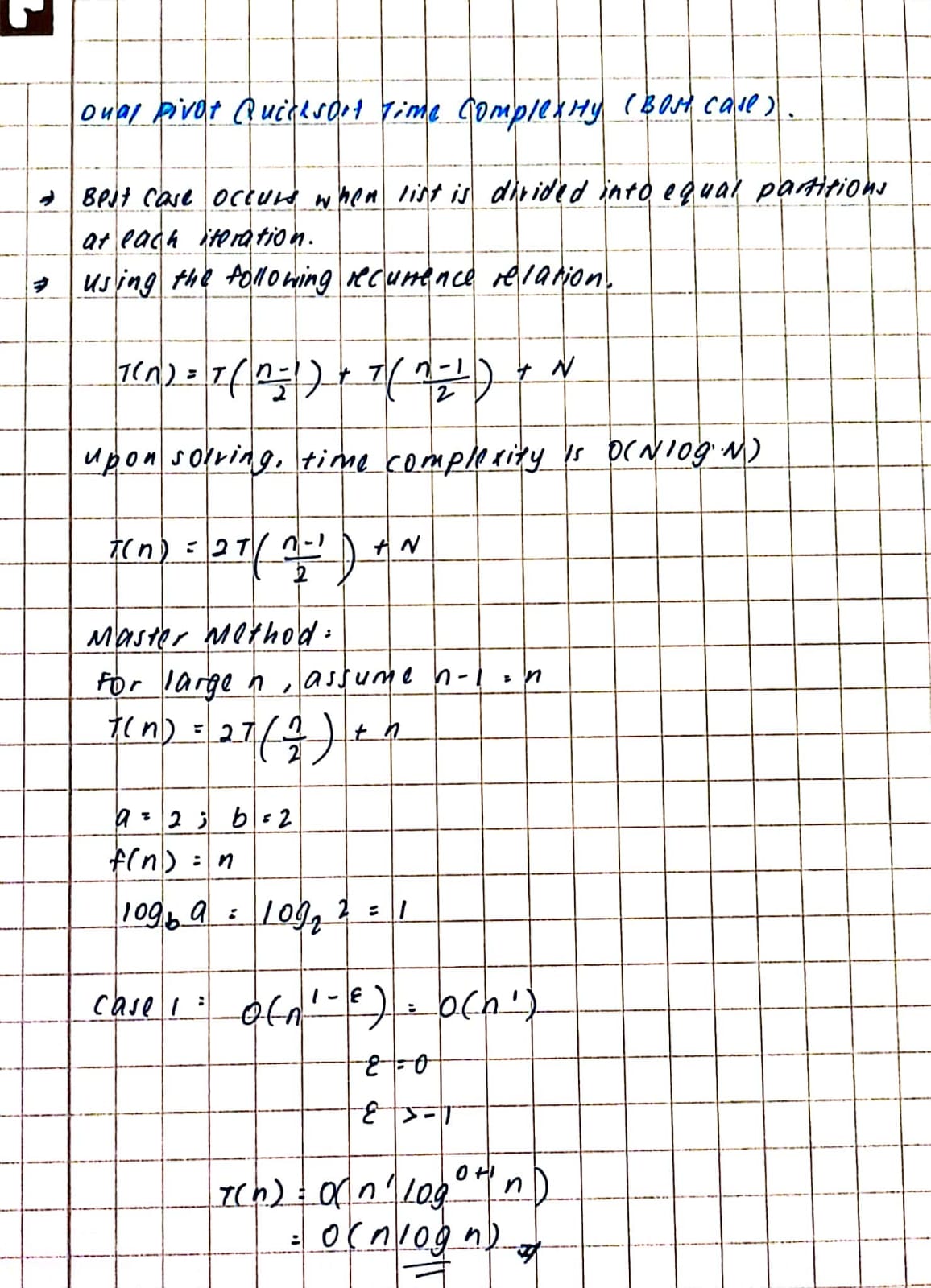
On the other hand, the average number of swaps for Hoare’s partition is

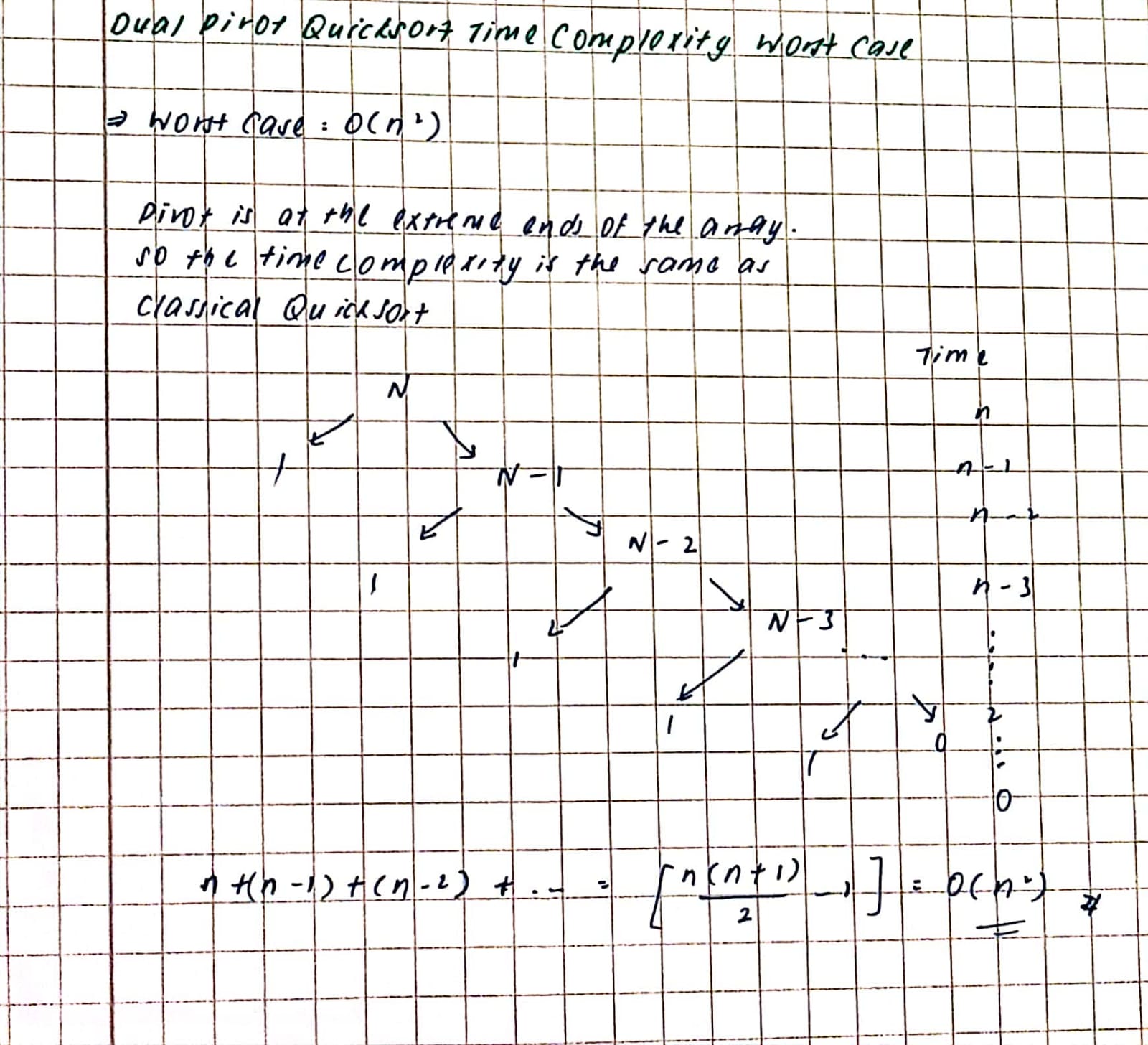


This is due to the fact that two indices i and j run towards each other until they cross, which always happens at x. This effectively divides the array into two parts: A left part which is scanned by i and a right part scanned by j. Whenever an element is not in the correct partition, for example, a large element (larger than x, thus belonging in the right partition) which is currently located in the left part and a small element located in the right part. The number of small elements initially in the right part equals the number of large elements in the left part. (Quoted from the reference link.)

For the median of three method, the first, middle and last elements of the array are chosen as pivots, and the median of those three elements as the pivot. The median approach helps speed up the sorting process because it would lead to more even partitioning in the array. This helps avoid the worst case scenario. Picking the overall median would give you the most balanced split, and hence the best run time.

Problem 7.2

1. Code is included in the file a7\_p2a.cpp
3. 



1. Code is included in a7\_p2c.cpp.