3. Mathematically derive the average runtime complexity of the non-random pivot version of quicksort.

A. In the non-random pivot version of Quicksort, we're considering the middle element of the array as the pivot. Elements less than the pivot will be in the left subarray and elements larger than the pivot will be in the right subarray following partitioning.

Let's denote the T(nl) and T(nr) as the average runtime complexities of Quicksort for the left and right subarrays, respectively. The recurrence relation for the average runtime complexity of Quicksort can be expressed as follows:

$$T(n) = T(nl) + T(nr) + O(n)$$

Where O(n) denotes the time complexity of partitioning the array.

Since we're using the middle element as the pivot, the probability of any element being chosen as the pivot is 1/n and where n is the size of the array. The expected value of p is (n+1)/2 which is the midpoint array. And the expected value of nl and nr is (n-1)/2 for each.

Substituting the above values in recurrence relation:

$$T(n) = 2T((n-1)/2) + O(n)$$

This equation indicates that T(n) is twice the value of T at (n-1)/2. This says that solution is recurrence relation is logarithm function as in each of the step n is halved equally and there are O(log n) levels in recursion tree.

Therefore, the average runtime complexity of the non-random pivot version of Quicksort is **O(nlog n).** This says times taken by the Quicksort to sort an array of size n grows logarithmically with n.