⇨ **2. For the non-random pivot version of quicksort show the following benchmarks on the same graph:**

2a) best case (generate a set of inputs that will always be the best case, repeat for multiple array input sizes "n").

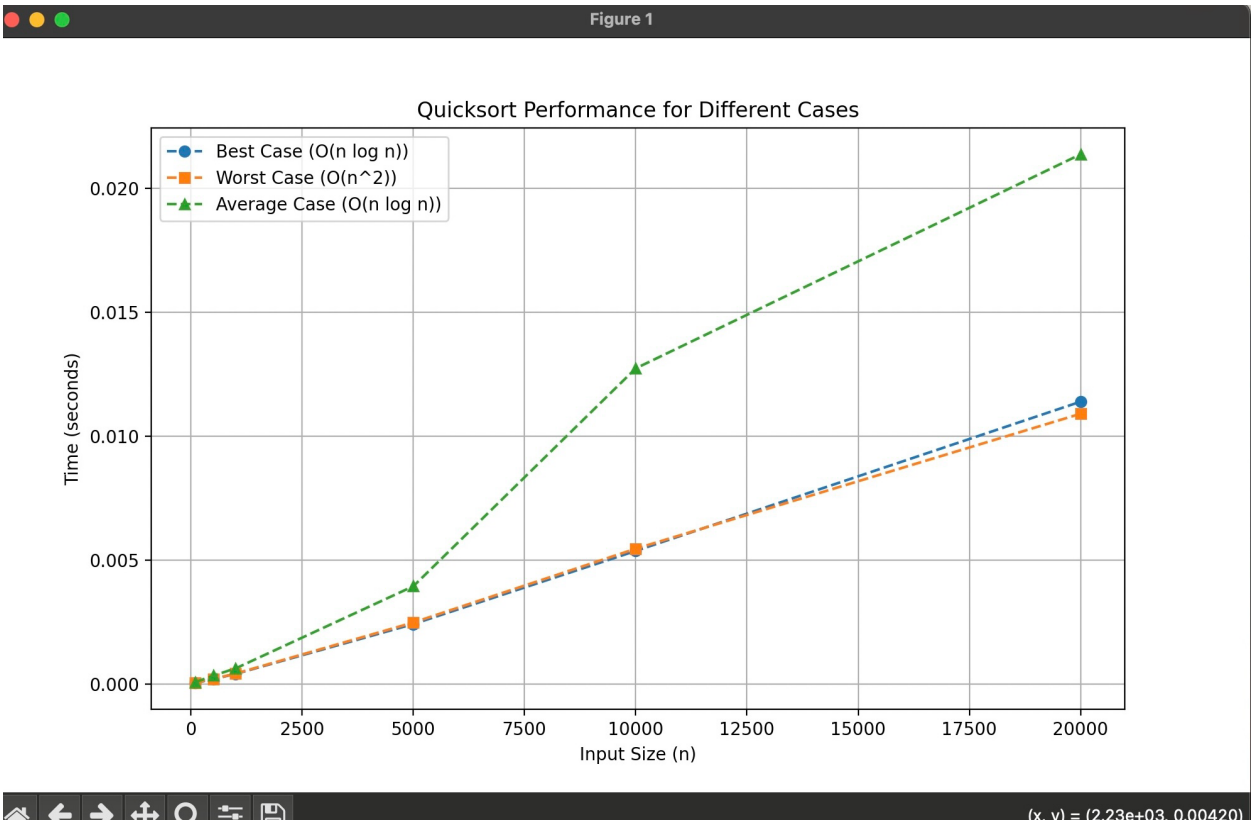
2b) worst case (generate a set of inputs that will always be the worst case, repeat for multiple array input sizes "n").

2c) average case (generate a set of inputs from a uniform distribution, repeat for multiple array input sizes "n").

⇨ Here,
Benchmarks for non-random pivot version:

Approach:

- Best Case: We use an already sorted array, which leads to an O(n log n) runtime.
- Worst Case: We use a reverse-sorted array, which leads to $O(n^2)$ runtime.
- Average Case: We generate an array with random values, leading to an expected O(n log n) runtime.



⇨ Explanation of the Benchmark:

- Best Case: Pivot always results in equal splits → O(n log n)

  Input: [0, 1, 2, ..., n-1]

- Worst Case: Pivot always results in worst splits → $O(n^2)$

  Input: [n, n-1, n-2, ..., 1]

- Average Case: Random input → Expected O(n log n)

  Input: Random shuffled array.

⇨ **Expected Graph Trends:**

- **Best case** (blue) → Should be the **fastest**, following **O(n log n)**.
- **Worst case** (red) → Should grow **much faster**, following **$O(n^2)$**.
- **Average case** (green) → Should be close to **best case** but slightly slower.