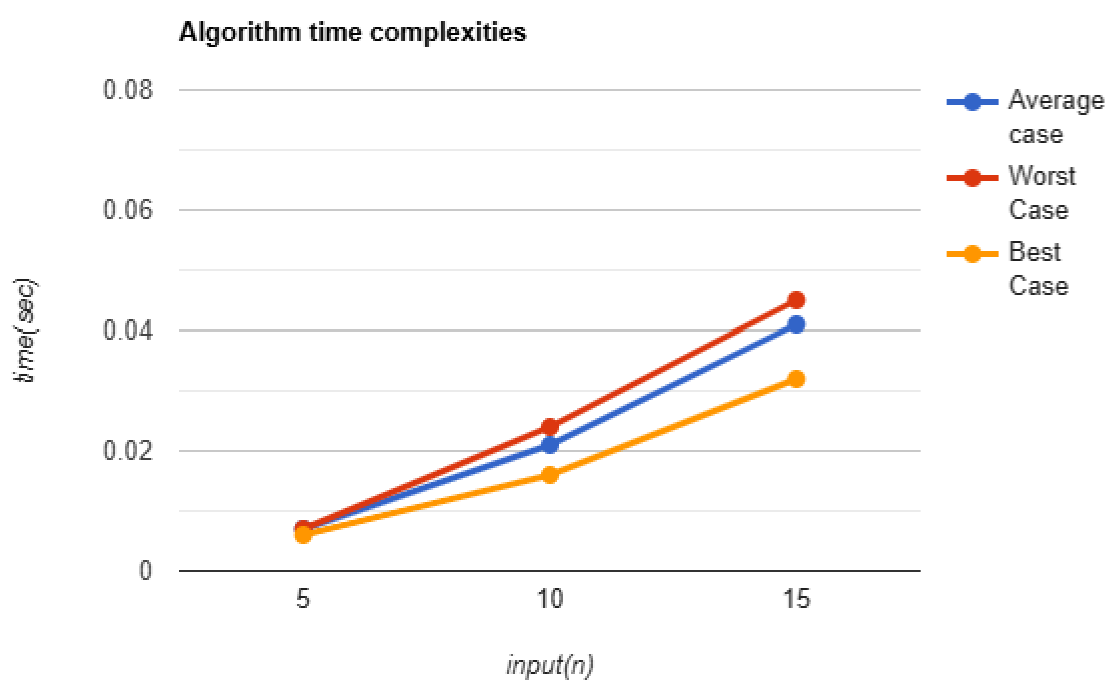
1. On GitHub.
2. Inside nonrandom_quicksort.py file. Plot:

```
nts\allHandsOn\algorithmsAllHandsOn\handsOn6> python nonrandom_quicksort.py
[31, 40, 30, 26, 80] [45, 77, 44, 71, 97, 80, 86, 96, 30, 37] [77, 30, 1, 31, 91, 8,
15, 11, 3, 20, 17, 27, 29, 67, 78]
quick Sort Time (average case): 0.007204499968793243
quick Sort Time (average case): 0.021463299985043705
quick Sort Time (average case): 0.0409170999773778
Average QuickSort Time (average case): 0.02319496664373825
[1, 2, 3, 4, 5] [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] [45, 46, 47, 48, 49, 50,
 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]
quick Sort Time (worstcase case): 0.007399099999538213
quick Sort Time (worstcase case): 0.024617400020360947
quick Sort Time (worstcase case): 0.045272600022144616
Average QuickSort Time (worstcase case): 0.025763033345962565
[1, 5, 2, 4, 3] [2, 7, 1, 9, 4, 8, 3, 10, 6, 5] [4, 12, 2, 10, 7, 14, 1, 15, 5, 13, 3
, 11, 6, 9, 8]
quick Sort Time (bestcase case): 0.006171699962578714
quick Sort Time (bestcase case): 0.016139400001698211
quick Sort Time (bestcase case): 0.03228809998836368
Average QuickSort Time (bestcase case): 0.018199733322641503
```

My laptop specification: RAM 8 GB, ROM 512 GB, CPU Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz   2.11 GHz



3. In the quicksort algorithm, we pick a pivot and the array is splitted into 2 parts, with one side having smaller numbers than the pivot and the other side having larger ones. This process is repeated for each side array until the array is fully sorted. Each time that the array is splitted, every number in the array is compared to the pivot which takes approximately n steps. Now, the next value to find is how many times can we divide n in half, which is log(n) times since we keep splitting until we reach the array of size 1.

   Therefore, we get:
   number of steps in each level * number of levels
   => n*log(n) = O(nlogn).

Algorithms were Implemented with the help of the course book given pseudocode for quicksort related functions.