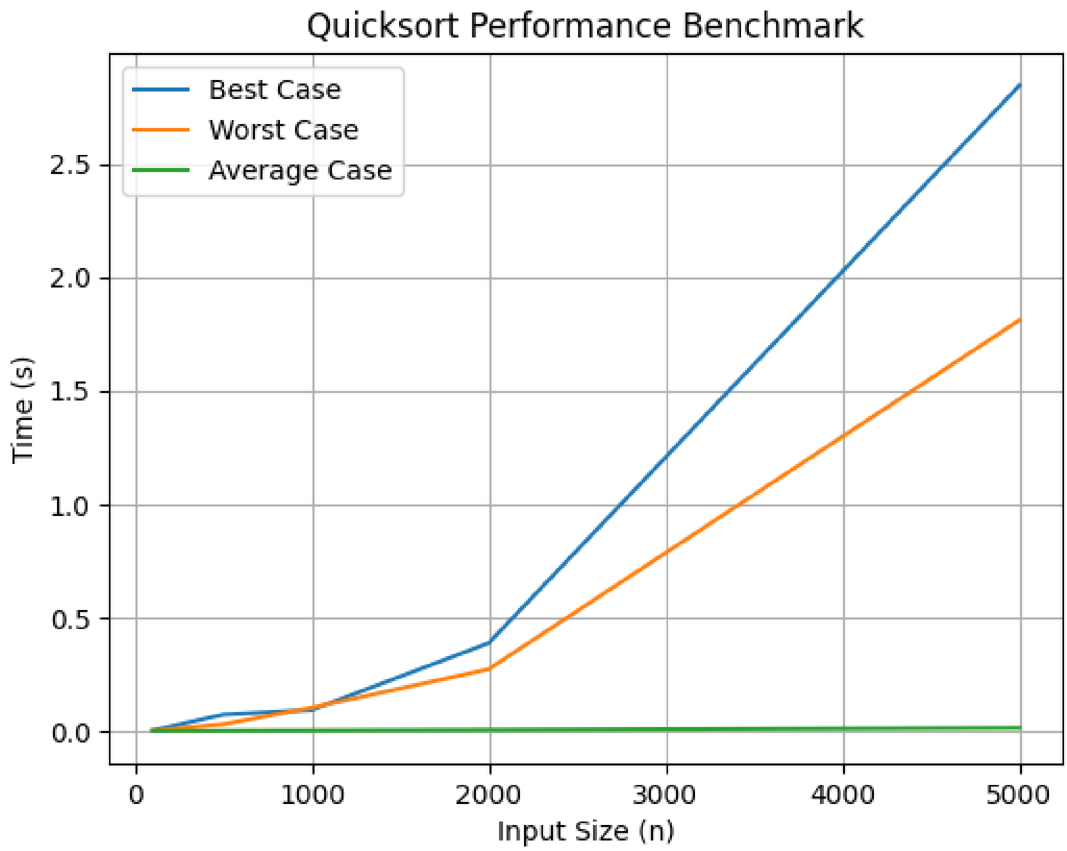
1. **Implement both versions of quicksort (random and non-random choice for the pivot) and share the GitHub repository with your source code.**

   Output:

```
Sorted array (non-random pivot): [2, 6, 8, 9, 10, 11]
Sorted array (random pivot): [1, 2, 3, 8, 17, 20]
```

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



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

<u>Best case:</u>

For the best case scenario pivot element will be the median of the given input array

$T(n) = 2T(n/2)+ cn$ where c=constant ----1
And $T(n/2)= 2T(n/4)+ c\, n/2$ ------------------2 and so on
from 1 and 2 we get
$T(n) = 4\, T(n/4)+ 2cn$
So we can say that
$T(n) = 2^k T(n/2^k) + kcn$
$2^k = n$
$K = \log n$
So $T(n) = nT(1) + n\log n$
Hence best case time complexity is $O(n\log n)$

<u>Worst Case:</u>

In worst case scenario the pivot element will be either the largest or the smallest element of the array

$T(n) = T(n – 1) + nc$ where c = constant ----1
$T(n-1)= T(n – 2) + (n – 1) c$ ---------------------2
From 1 and 2 we get
We get $T(n) = T(n-2) +(n-1)c + nc$
Similarly we get
$T(n) = T(n–k) + k * n * c – c * (k*(k – 1))/2$
By replacing k with n we get
$T(n) = T(0) + n * n * c – c * (n * (n-1)/2)$
$= n^2 – n*(n-1)/2$
$= n^2 /2 + n/2$
Hence worst case time complexity is $O(n^2)$

<u>Average case:</u>

For the average case consider the array gets divided into two parts of size k and (n-k).

$T(n) = T(n – k) + T(k)$
$= 1 / n * [\sum_{i=1}^{n-1} T(i) + \sum_{i=1}^{n-1} T(n-i)]$
Lets approximate summation of T(i) to T(n-i) we get

$T(n) = 2 / n * \sum_{i=1}^{n-1} T(i)$

$nT(n) = 2 * \sum_{i=1}^{n-1} T(i)$ ---------1

also for n-1

$(n-1)T(n-1) = 2 * \sum_{i=1}^{n-2} T(i)$ ------2

Subtracting 1 and 2 we get

$n * T(n) – (n – 1) * T(n – 1) = 2 * T(n – 1) + n^2 * c – (n – 1)^2 * c$ where c = constant

$n * T(n) = T(n – 1) * (2 + n – 1) + c + 2 * n * c – c$

$= (n + 1) * T(n – 1) + 2 * n * c$

Divide both side by n*(n-1) and we will get

$T(n) / (n + 1) = T(n – 1)/n + 2 * c / (n + 1)$ -----3

If we put n = n-1 it becomes

$T(n – 1) / n = T(n – 2)/(n – 1) + 2*c/n$

From equation 3

$T(n) / (n + 1) = T(n – 2)/(n – 1) + 2*c/(n + 1) + 2*c/n$

Similarly, we can get the value of T(n-2) by replacing n by (n-2) in the equation 3.

Finally we get

$T(n) / (n + 1) = T(1)/2 + 2*c* [1/2 + 1/3 + . . . + 1/(n – 1) + 1/n + 1/(n + 1)]$

$T(n) = 2 * c * \log n * (n + 1)$

$T(N) = \log n * (n + 1)$

So the average case time complexity is O(nlogn).