DATA-Lab7

Question 2.

The time complexity of each function is as followed:

1. quickSort(): O(nlog(n))
2. mergeSort(): O(nlog(n))
3. heapSort(): O(nlog(n))
4. insertionSort(): O(n^2)

⇒ ***Expectation:*** T_insertion > T_quick ~ T_merge ~ T_heap

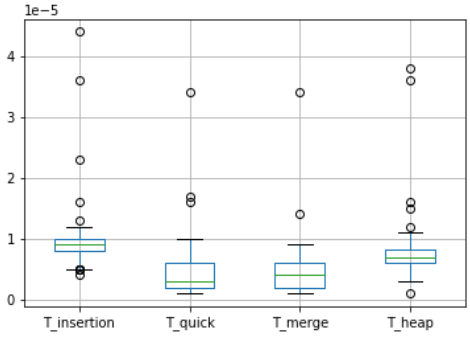
Method:

1. The running time of each function on an array of 1000 integers is collected and plotted.
2. Wilcoxon one sided or two sided tests are performed suitable with the result in the diagram.

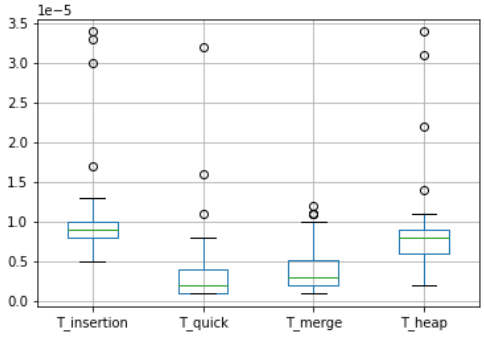
As you can see from 5 attempts below, insertionSort() is the slowest algorithm, followed by heapSort(). However, it is hard to tell whether quickSort() or mergeSort() is faster. I performed a two-sided Wilcoxon test with ***H0*** that there is no significant difference in running time of quick Sort() and mergeSort(). In 3 out of 5 attempts, the H0 was not rejected (pvalue > 0.05). Therefore, I performed 500 more Wilcoxon tests and take the major vote as the result. How I count the cases is presented in the flowchart below.

⇒ The result shows that around 27.6% of the tests, quickSort() is faster than mergeSort() and 0% in the reverse way. This implies that around 72.4% of the tests, there is no significant difference in performance of quickSort() and mergeSort().

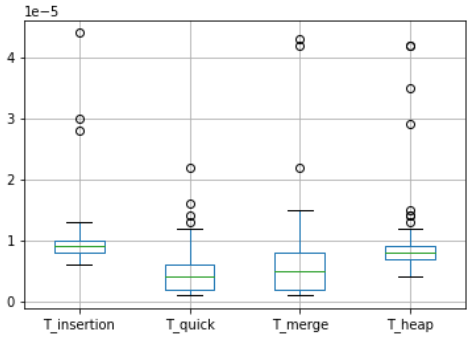
To conclude, for an array of 1000 numbers, either quickSort() or mergeSort() is the best algorithm for use.



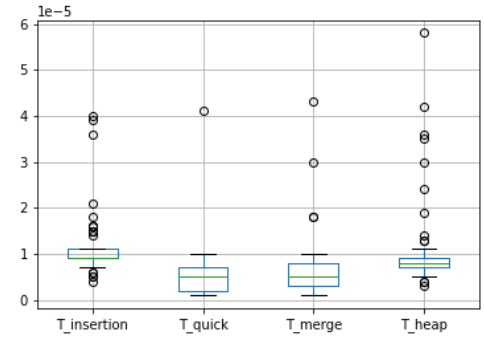
Attempt no.1



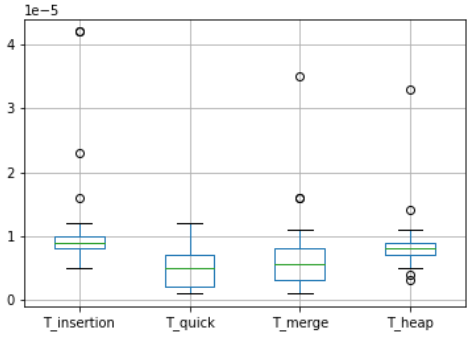
Attempt no.2



Attempt no.3



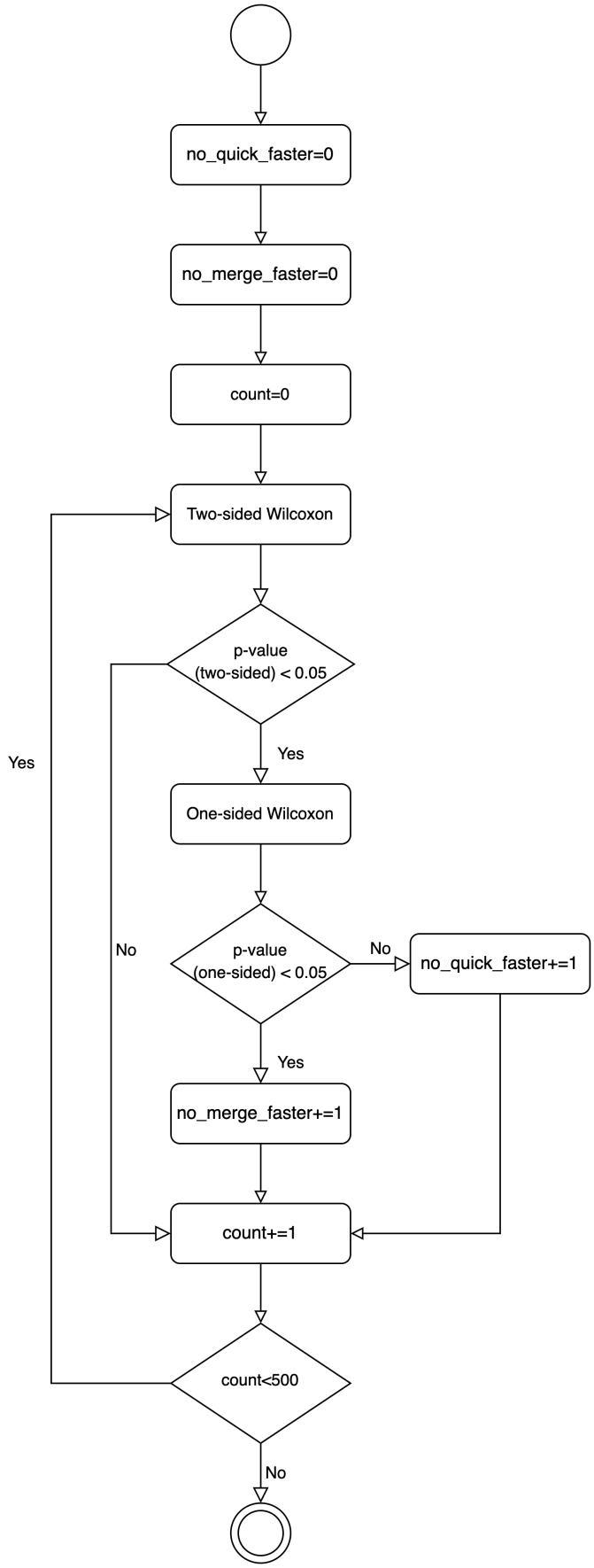
Attempt no.4



Attempt no.5

```
Wilcoxon two-sided
Attempt no.1
   T_quick-T_merge pvalue: 0.2458
Attempt no.2
   T_quick-T_merge pvalue: 0.0107
```

```
Attempt no.3
   T_quick-T_merge pvalue: 0.0286
Attempt no.4
   T_quick-T_merge pvalue: 0.0571
Attempt no.5
   T_quick-T_merge pvalue: 0.0608
```


no_quick_faster=0
no_merge_faster=0
count=0
Two-sided Wilcoxon
p-value
(two-sided) < 0.05
Yes
Yes
One-sided Wilcoxon
p-value
(one-sided) < 0.05
No
No
no_quick_faster+=1
Yes
no_merge_faster+=1
count+=1
count<500
No

```
%quickSort() faster: 27.60
%mergeSort() faster: 0.00
```