

**ADVANCED ALGORITHM ANALYSIS**

**SUB CODE: CP5602**

**ASSIGNMENT REPORT**

**PREPARED BY:**

**PRAKHAR DIXIT (13906375)**

**GUIDED BY:**

**Prof. Mrs. SHAILEY CHAWLA**

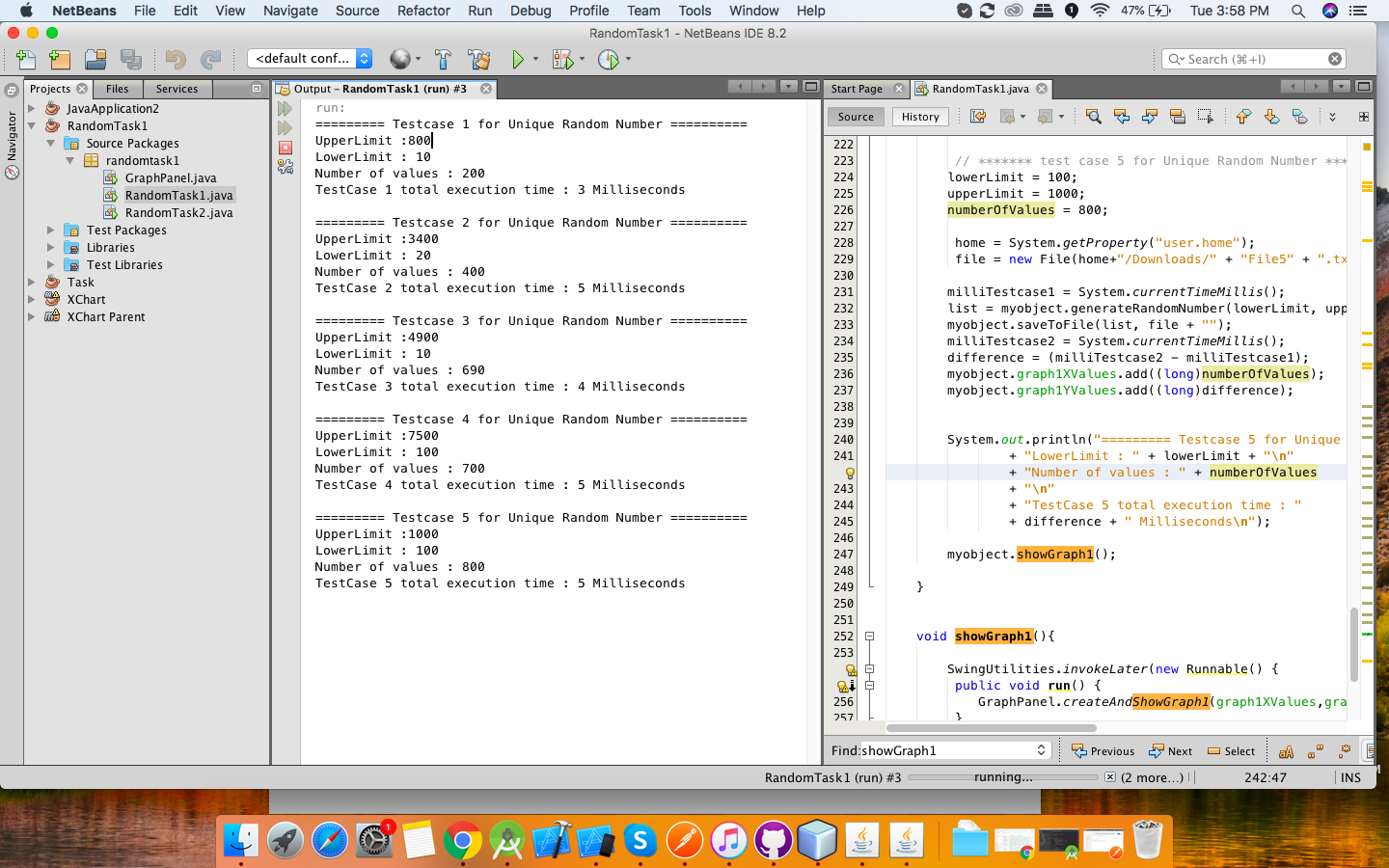
***Task 1 - Implementation and performance analysis of computation algorithms***

1. **Random number generator**

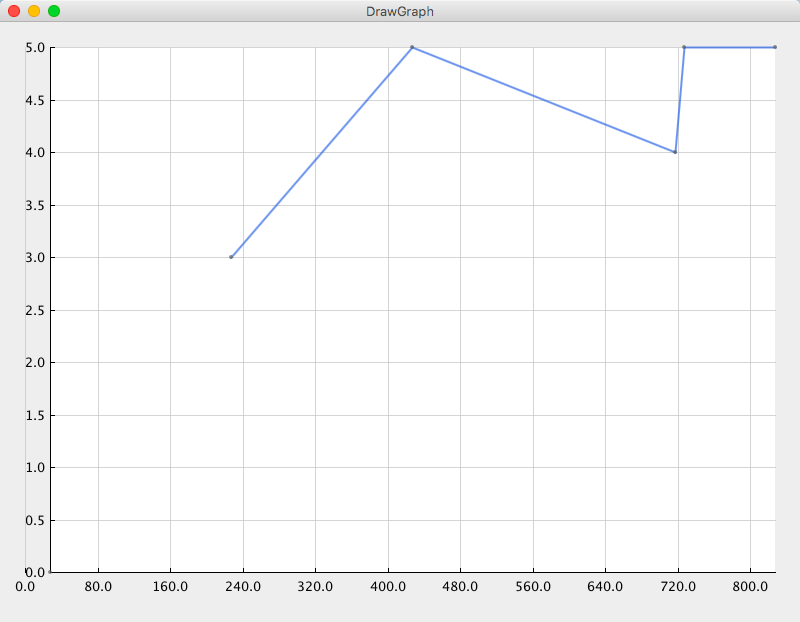
Here we tested various scenarios for random number generation with

1. Unique Random values
2. Duplicate Random values

1. **Unique Random Values**

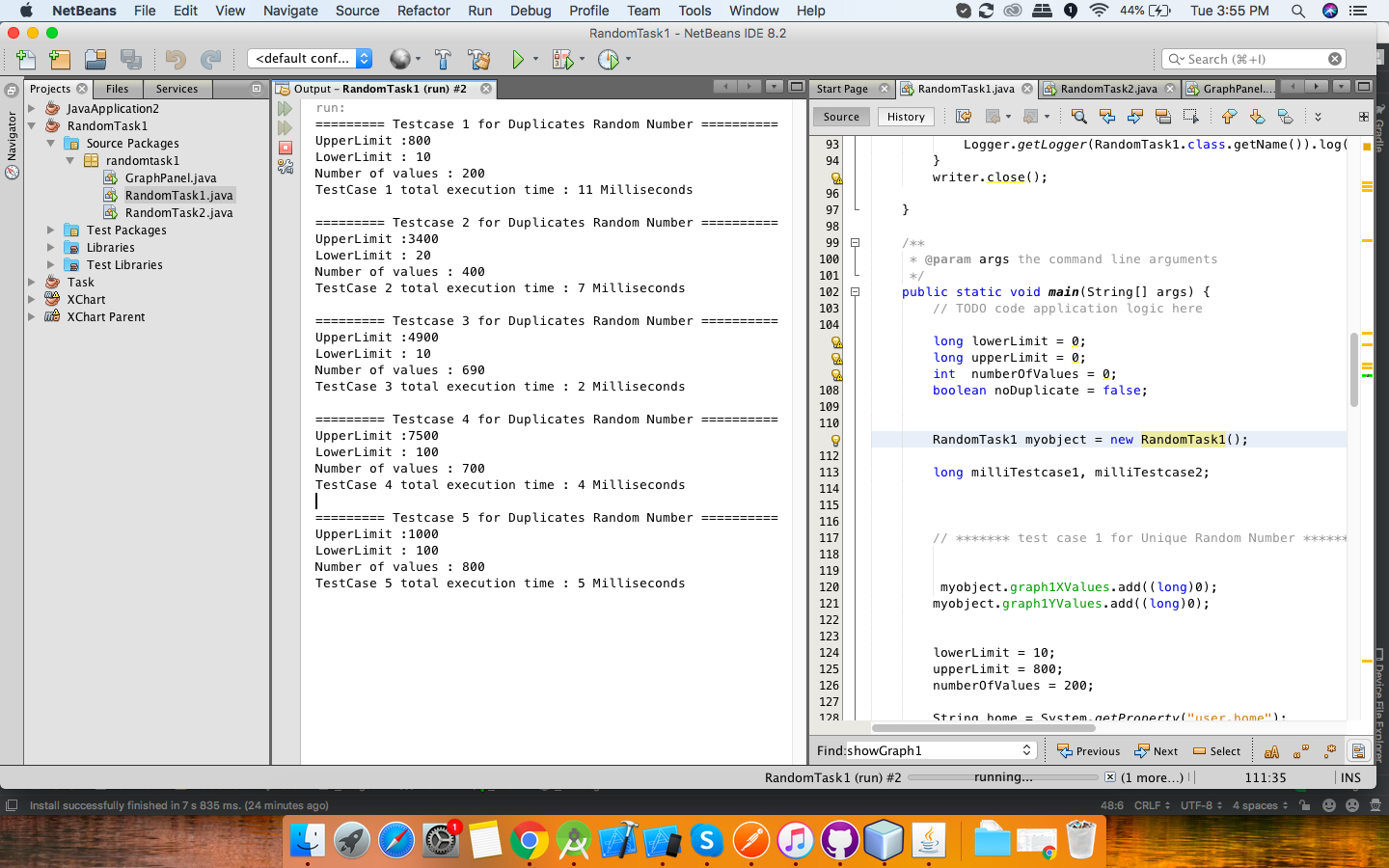


Here we took 5 different test cases as shown above with different set of ranges and different set of input with unique random values , we obtained the following graph

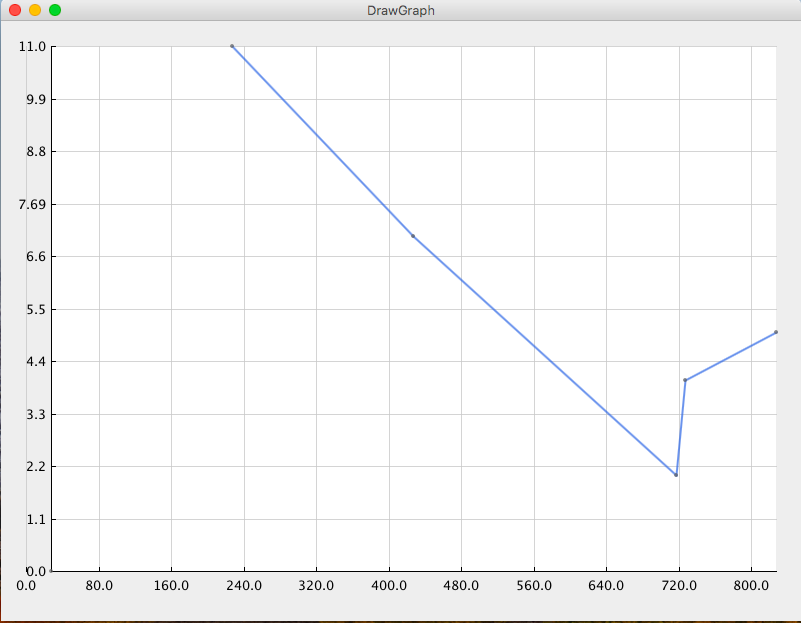


Based on the graph, The Big oh characterization of unique random value is O(N\*log N)

1. **Duplicate Random Values**



Here we took 5 different test cases as shown above with different set of ranges and different set of input with duplicate random values , we obtained the following graph



Based on the graph, The Big oh characterization of Duplicate random value is O(N\*log N)

Ideally, For random number generation the best case is O(N)

***COMPARISON GRAPH :***

Here is the time analysis graph between random number generated with and without duplicate values.

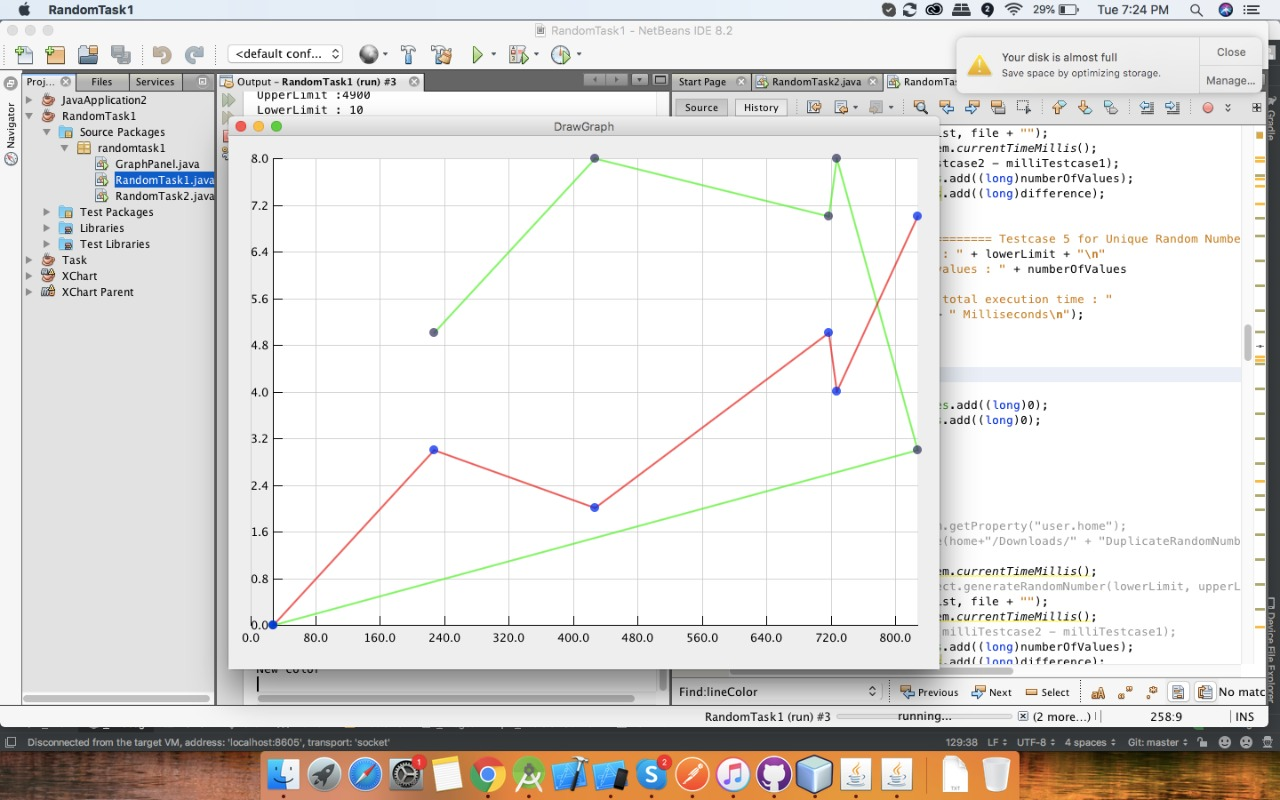
Based on the graph we found out that time taken by unique random value is comparatively less than time taken by duplicate random value. (Take Test scenario 1 in both the cases for example) Upper bound and lower bound limit is same in both cases(i.e., 800 and 100 respectively). The number of iterations is also the same (i.e.,200).

We see the output in both the cases in unique random value the time taken is 3 milliseconds where as in duplicate random values the time taken is 11 milliseconds.

Hence, By the above analysis we can say that time taken where the duplicates are allowed is more than where the duplicates are not allowed.

Time analysis graph:

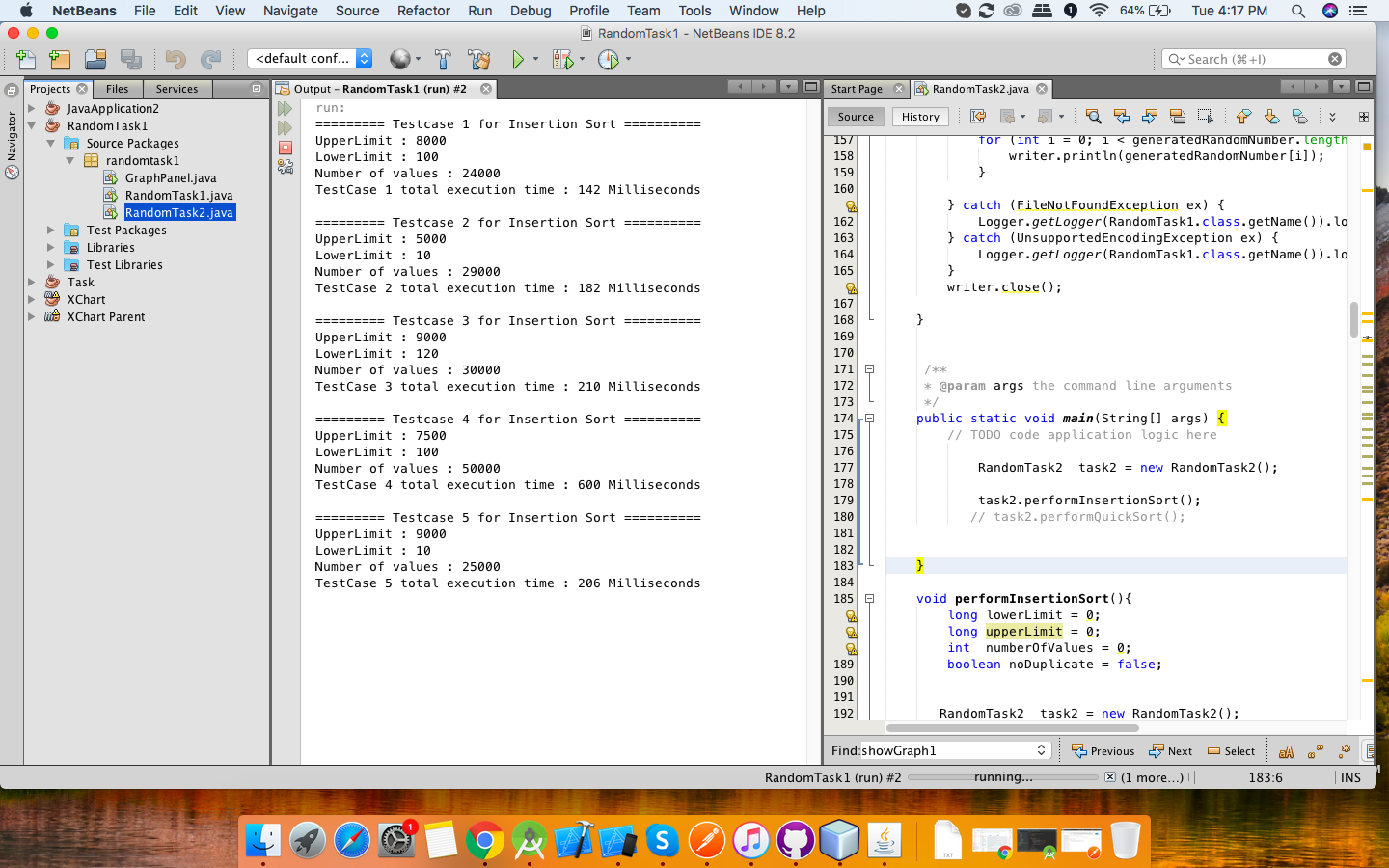
Green color line symbolizes unique random values where as Red color line symbolizes duplicate random values.



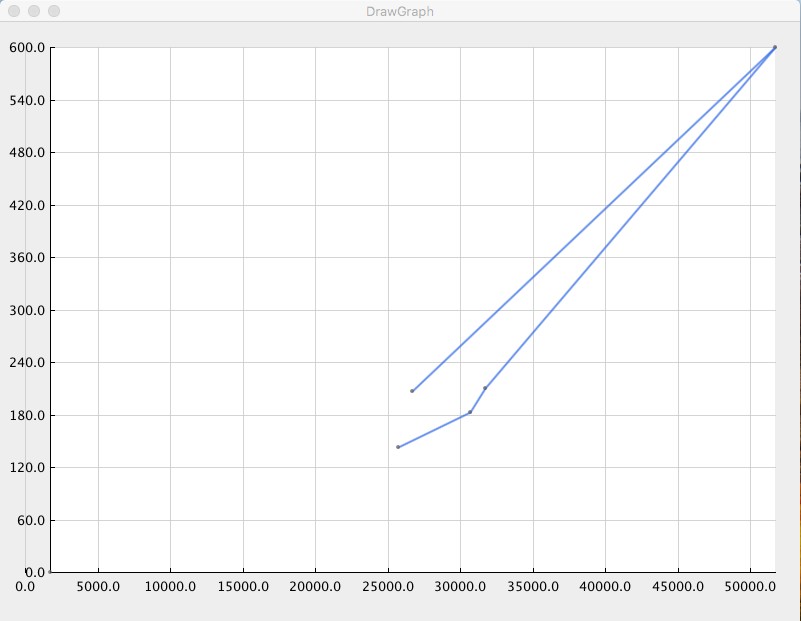
***Task 2 – Implementation and performance analysis of sorting algorithms***

Here we took the same range with different inputs of random generated numbers above to sort(Ascending order) in two ways:

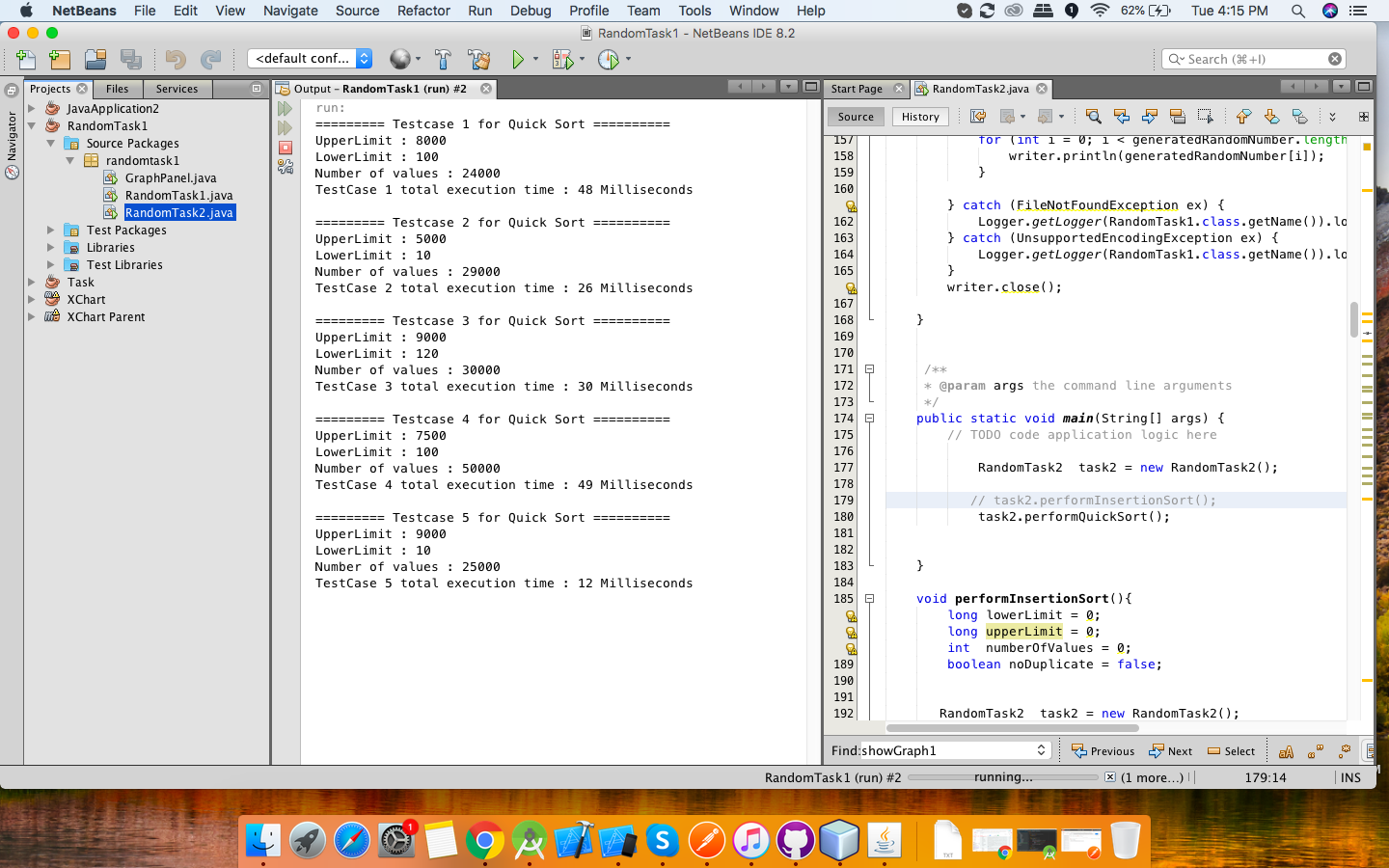
1. Insertion Sort
2. Quick sort
3. **Insertoin sort**



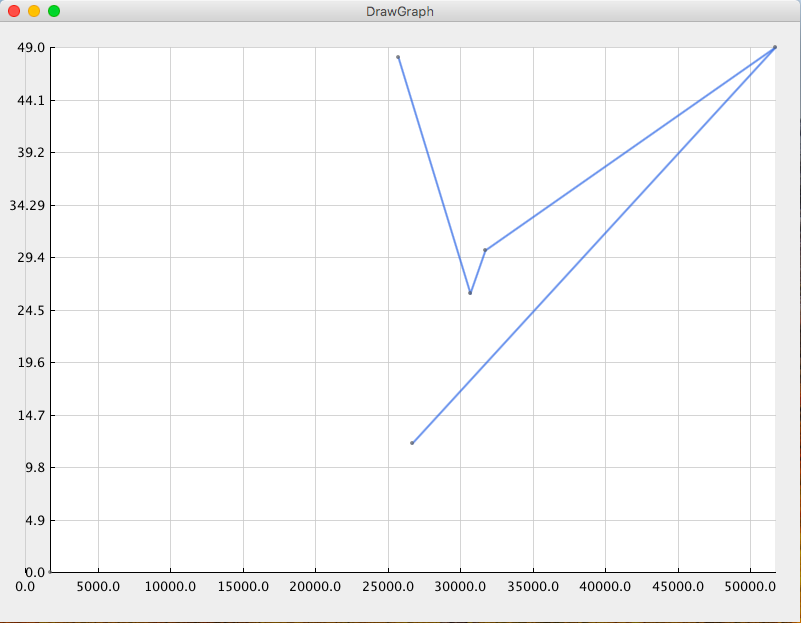
The above images describes 5 different test scenarios for insertion sort where the input was taken from the random generated function. After sorting we obtained the following graph:



1. **Quick sort**



The above images describes 5 different test scenarios for quick sort where the input was taken from the random generated function. After sorting we obtained the following graph:



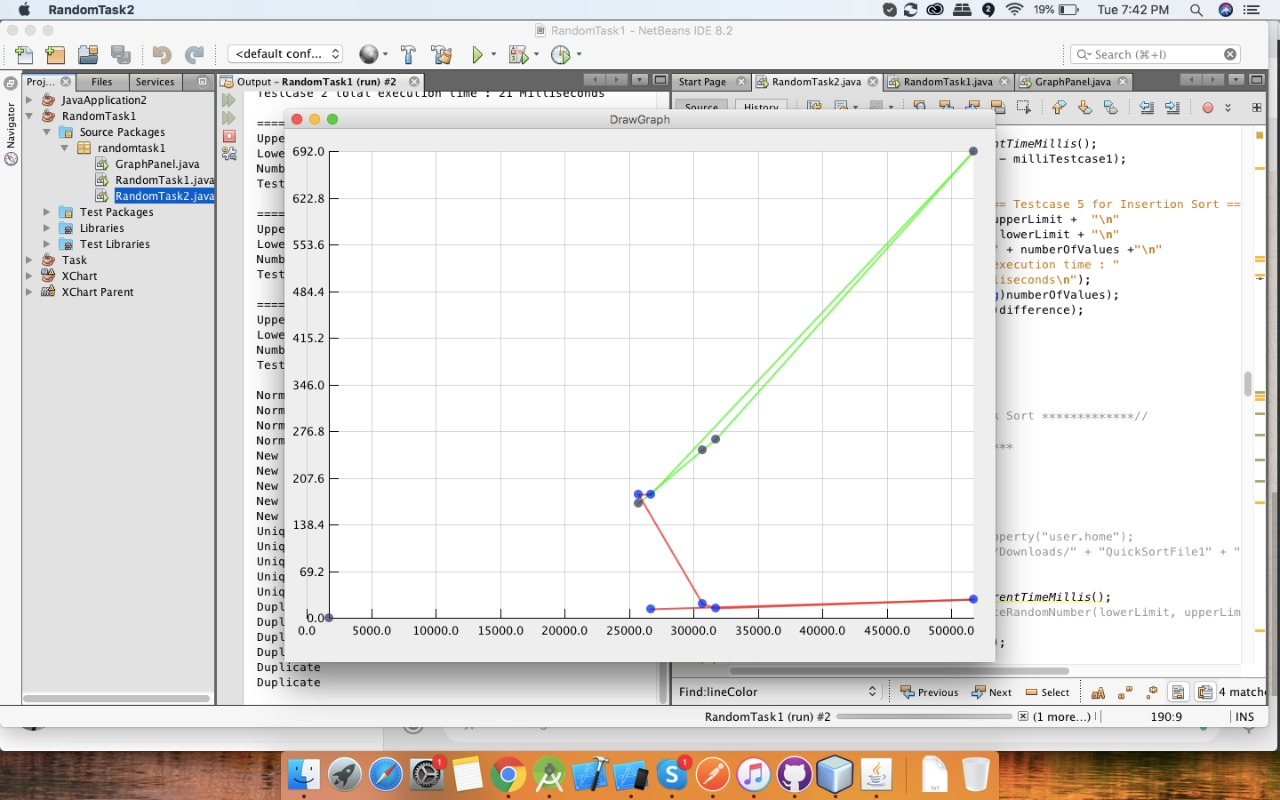
Based on the input values, range and the Time analysis graph, we got the best case, average case and the worst case of both the sorting algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Sort Algorithm | Sorting Time | | |
| Worst | Average | Best |
| Insertion Sort | O(n^2) | O(n^2) | O(nlogn) |
| Quick Sort | O(n^2) | O(nlogn) | O(nlogn) |

Ideally, the best case of insertion sort is O(n), but based on the input here we found out the best case for this inputs is O(nlogn).

**COMPARISON GRAPH** :

After comparing these two sorting algorithm, we found out that quick sort is better than insertion sort, though based on this input they have the same time complexity i.e.,O(nlogn), But the actual time taken by quick sort is very less.

In the below comparison graph, 

# ***Task 3 (Advanced)***

**Q.A) Does no of CPU in a machine significantly affect the performance?**

Ans A:- Yes the number of CPUs makes affect on performance of the algorithm, also it depends upon the optimization of the algorithm ,the algorithm is made to use only one thread or core of a CPU then it doesn’t matter what the number of cores a CPU contains, it will only use one thread according to algorithm.

Multi core processor can offer significant performance improvements over their single core counter part for certain kind of parallized task, often demanding new programming paradigms to efficiently utilize the complex Architecture involved.

**Q.B) Does having greater RAM help improves performance all input types?**

Ans B:- Yes, increasing the RAM capacity help improve performance cause in case where RAM size is less the data is then put to virtual memory which is indirectly a hard disk space. For that situation the OS needs to read from the hard disk which is slower than RAM.

In case of high capacity of RAM available the random number get loaded to RAM where the data can easily be access and there is no need to go to hard disk which gradually increase the performance.

The amount of time that is takes RAM to receive a request from the processor and receive a request from the processor and then read or write data.

**Q.C) Does the kind of OS running the program play an important role, in maximizing your performance?**

Ans C:- While testing it seems the in some cases the performance of the linux distribution is better. In Both the cases hardware are same i.e. i5 7th Gen 8 thread x64 bit architecture with 8GB RAM.

There could be many points that is responsible for the peroformance hike someof them are as follows:-

* Thread scheduling:- As compare to windows linux has much more optimized scheduling algorithms.
* RAM and memory management in linux is good
* I/O handling:- As the random number imput in our algorithm of sorting , there is a drastic performance hike in linux as it has better I/O handling mechanism.

But, in some cases where the input i.e. random number generation is too high, the system both windows and linux has very low performance, this could be due to system bottle neck condition.

**CONCLUSION :**

Task 1) Time taken is more where the duplicates are allowed.

Task 2) Time taken by quick sort is less than the insertion sort.

Note: All the code for above output is in the zip file.

Github link is: