| Worst Case | | | | | | | |
|---|---|---|---|---|---|---|---|
| elements | 10 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | **Complexity** |
| Bubble (ms) | 0 | 0 | 4 | 59 | 5409 | | O(n^2) |
| Insertion (ms) | 0 | 0 | 4 | 108 | 10420 | | O(n^2) |
| Quick (ms) | 0 | 0 | 7 | 180 | | | O(n^2) |
| Merge (ms) | 0 | 0 | 1 | 2 | 26 | 208 | O(n lg n) |

| Best Case | | | | | | | |
|---|---|---|---|---|---|---|---|
| elements | 10 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | **Complexity** |
| Bubble (ms) | 0 | 0 | 4 | 24 | 1716 | | O(n) |
| Insertion (ms) | 0 | 0 | 0 | 0 | 2 | 4 | O(n) |
| Quick (ms) | 0 | 0 | 5 | 347 | | | O(n^2) |
| Merge (ms) | 0 | 0 | 1 | 3 | 21 | 198 | O(n lg n) |

| Average Case | | | | | | | |
|---|---|---|---|---|---|---|---|
| elements | 10 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | **Complexity** |
| Bubble (ms) | 0 | 0 | 4 | 194 | 13310 | | O(n^2) |
| Insertion (ms) | 0 | 0 | 3 | 40 | 3429 | | O(n^2) |
| Quick (ms) | 0 | 0 | 0 | 4 | 31 | 231 | O(n lg n) |
| Merge (ms) | 0 | 0 | 0 | 4 | 26 | 262 | O(n lg n) |

## Conclusion

Although the merge sort is hard to implement and its performance is not the best across all test ( as shown above). It is the most consistant at sorting arrays even with large array sizes. Due to the original sorting method for getting the worst/best case scenarios the time to sort 1,000,000 elements is very long, coupled with the time it takes to execute the algorithm many of the 1,000,000 did not run.

## References


Bubble

What is a Bubble sort in Java, educative, July 2021. [Online]. Available: https://www.educative.io/edpresso/what-is-a-bubble-sort-in-java

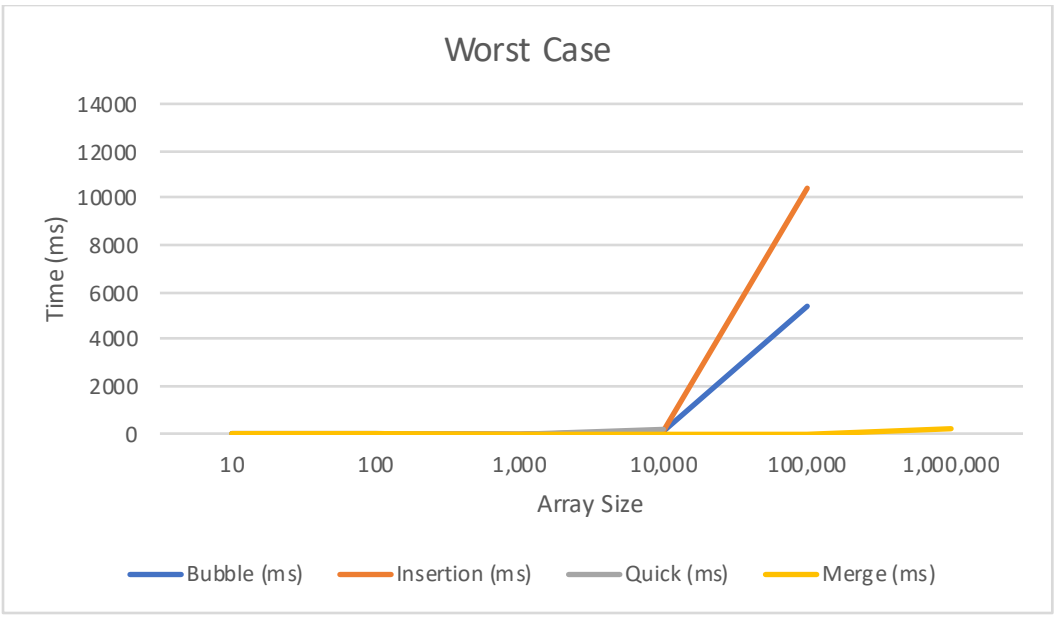
Insertion

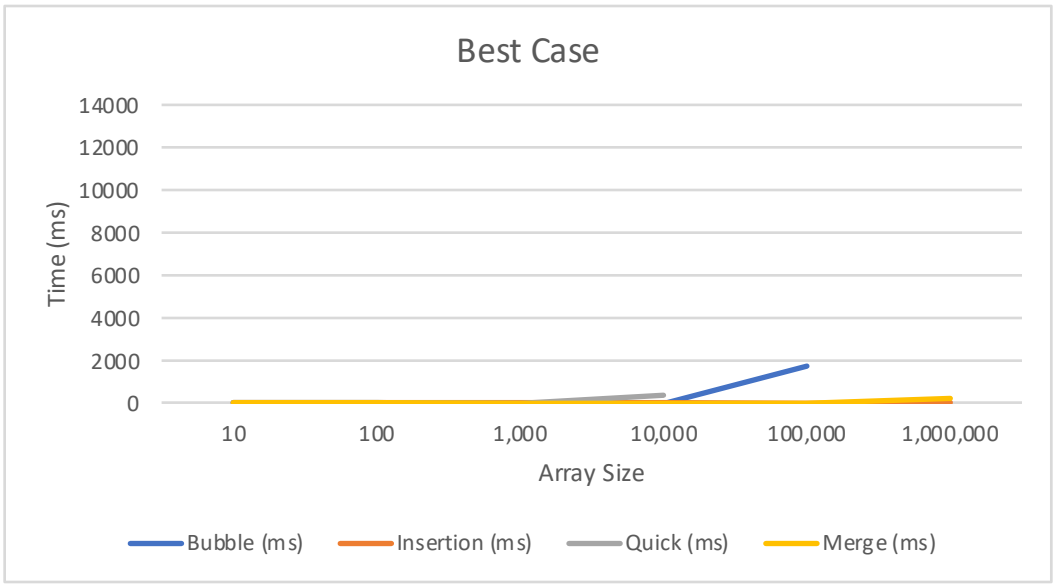
What is insertion sort in Java, educative, July 2021. [Online]. Available: https://www.educative.io/edpresso/what-is-insertion-sort-in-java

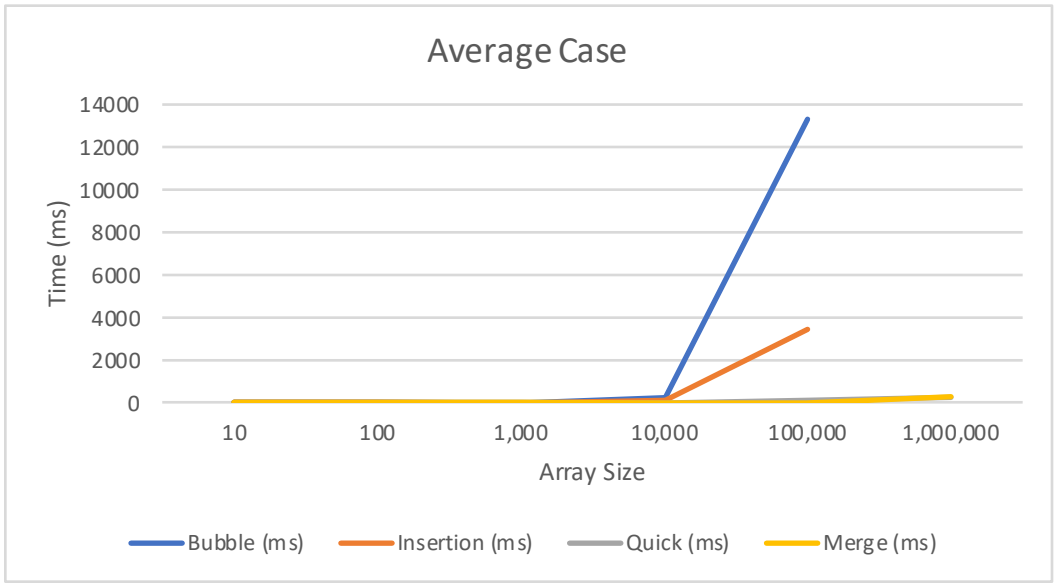
Quick

How to implement a QuickSort in Java, educative, July 2021. [Online]. Available: https://www.educative.io/edpresso/how-to-implement-quicksort-in-java

Merge


Worst Case
14000
12000
10000
8000
6000
4000
2000
0
Time (ms)
10
100
1,000
10,000
100,000
1,000,000
Array Size
Bubble (ms)
Insertion (ms)
Quick (ms)
Merge (ms)


Best Case
14000
12000
10000
8000
6000
4000
2000
0
Time (ms)
10
100
1,000
10,000
100,000
1,000,000
Array Size
Bubble (ms)
Insertion (ms)
Quick (ms)
Merge (ms)


Average Case
14000
12000
10000
8000
6000
4000
2000
0
Time (ms)
10
100
1,000
10,000
100,000
1,000,000
Array Size
Bubble (ms)
Insertion (ms)
Quick (ms)
Merge (ms)

Worst Case

In the worst case scenario(Descending Order) the Quick sort algorithm performed the worst of the four. In this sample size, the system cannot compute the time for sorting an array of 100,000 items originally ordered in descending order. This is because when the array is sorted in reversed order, the pivot point is the largest/smallest element. This is the result we expect to see from this algorithm due to how it uses the pivot element to sort an n number element array.
Interestingly enough, the merge sort was able to perform the best and closley matched its performance in the best case scenario.

Best Case

In the best case scenario (Ascending Order) the Quick sort algorithm performed the worst of the four. In this sample size, the system once again cannot compute the time for sorting an array of 100,000 items originally ordered in Ascending order. This is because when the array is sorted in ascending order, the pivot point is the largest/smallest element. This is the result we expect to see from this algorithm due to how it uses the pivot element to sort an n number element array.
After several runs of the best case scenario, the Insertion sort came out on top besting even the merge sort this could possibly due to the time it takes to split the array into the smallest component parts before recombinging them together.

Average Case

In the average case (Random Order) the worst performing algorithm is the bubble sort. Although it managed to run it formed worse in the random case than in the " worst case". The two advanced algorithms were the only algorithms that completed the sorting of the 1,000,000 elemnts array.