CMP201 - Presentation

Fraser Barker(1600196)

What is the problem?

- Sorting
- Large collections of integers

Algorithms

- Quicksort
  - Best/Average - $O(N \log N)$
  - Worst – $O(N^2)$
- Merge sort
  - Best/Average - $O(N \log N)$
  - Worst – $O(N \log N)$

Data structures

- Vector<int>
- ~~Array~~
- ~~List~~

Time Complexity of Data Structure

- Vector
  - Random access - constant *O(1)*
  - Insertion or removal of elements at the end - amortized constant *O(1)*
  - Insertion or removal of elements - linear in the distance to the end of the vector *O(n)*

Memory Complexity of Data Structure

- Vector
    - Automatic – Expansion/Contraction
    - Future growth
    - Memory exhausted

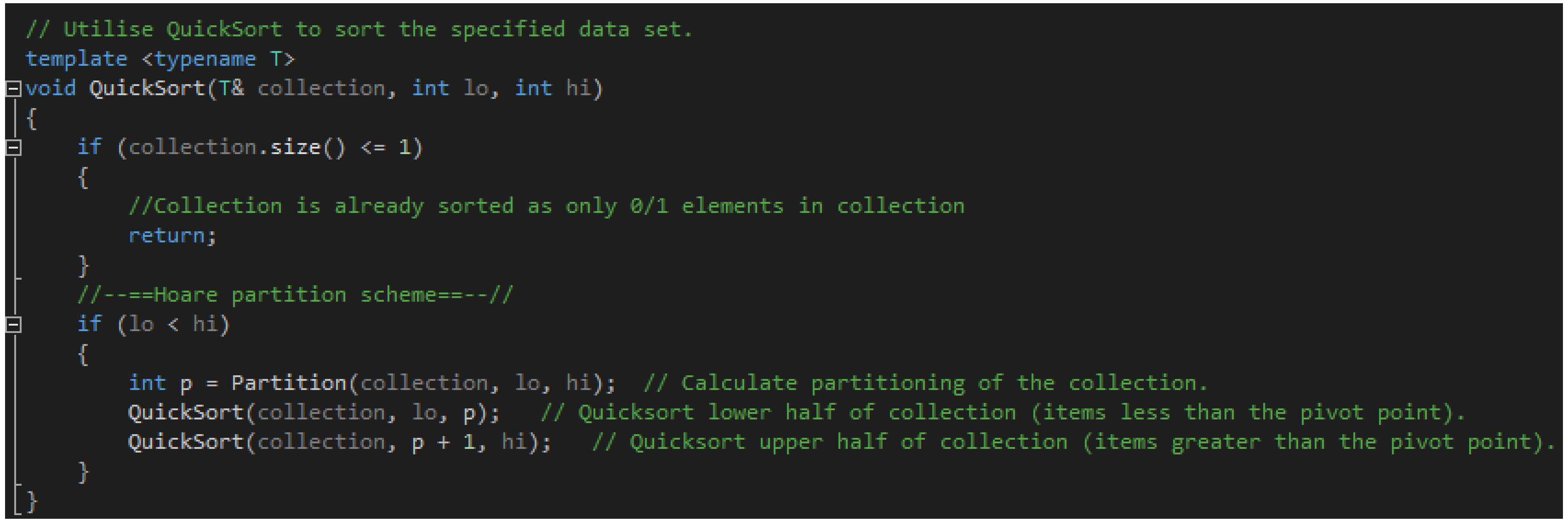
Needs of the Algorithm – Quicksort

- Requires data structure of array type or similar
- Requires O(1) random access to be fast

Needs of the Algorithm – Merge Sort

- Requires data structure of array type or similar

Place where vector<int> simplified code or improved performance



```
// Utilise QuickSort to sort the specified data set.
template <typename T>
void QuickSort(T& collection, int lo, int hi)
{
    if (collection.size() <= 1)
    {
        //Collection is already sorted as only 0/1 elements in collection
        return;
    }
    //--==Hoare partition scheme==--//
    if (lo < hi)
    {
        int p = Partition(collection, lo, hi);  // Calculate partitioning of the collection.
        QuickSort(collection, lo, p);   // Quicksort lower half of collection (items less than the pivot point).
        QuickSort(collection, p + 1, hi);   // Quicksort upper half of collection (items greater than the pivot point).
    }
}
```

Theoretical Time Complexity: Quicksort

- O(N log N)
- $O(N^2)$
- Faster

Theoretical Time Complexity: Merge Sort

- O(N log N)
- Slower

Sources of Error

- Timing errors
- Cache effects
- Random timing variation in the CPU/bus
- Activities of other CPUs and OS processes
  - **REPETITIONS**

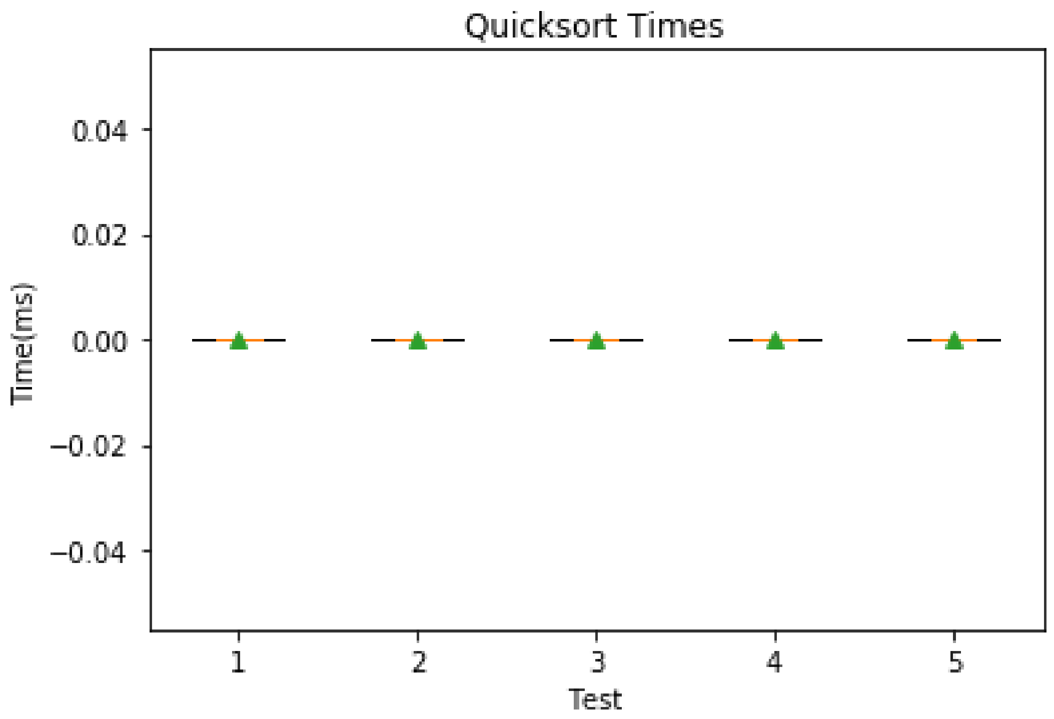
Performance Comparison

- Show variation (box plots)
- Appropriate statistics
- Varying size of input data
- Theoretical time complexity
- Results
- Compare results to theoretical performance.

Tests

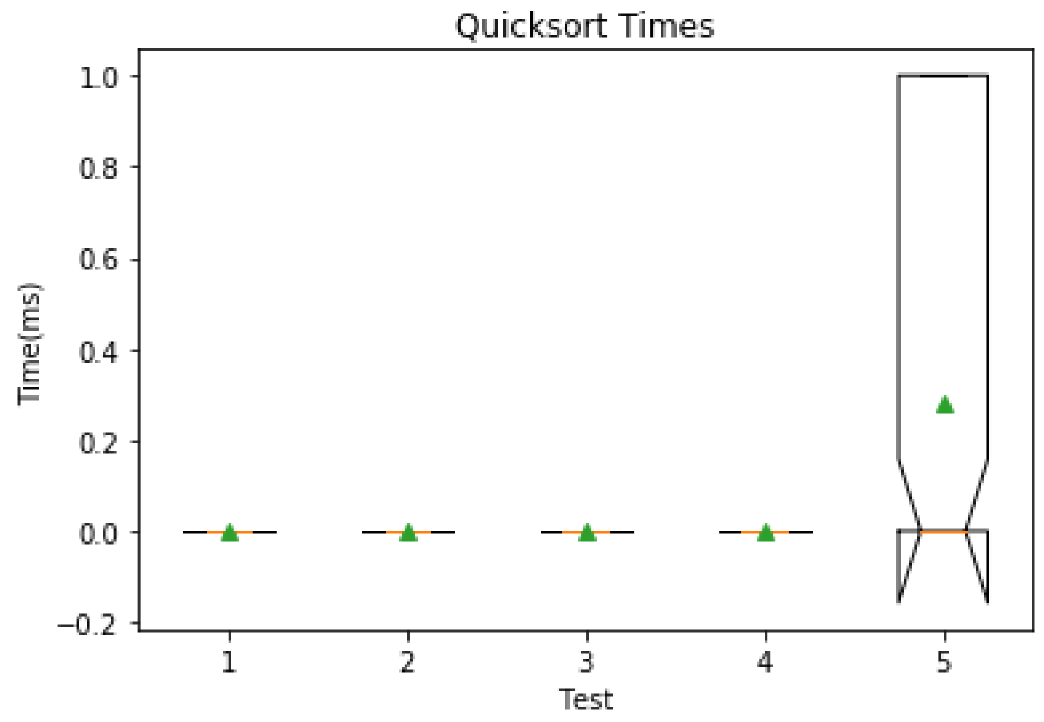
- Significance
  - Rank-sum test
  - Confidence interval
  - P-values
  - Effect size
  - A measure

Quicksort – 10,000



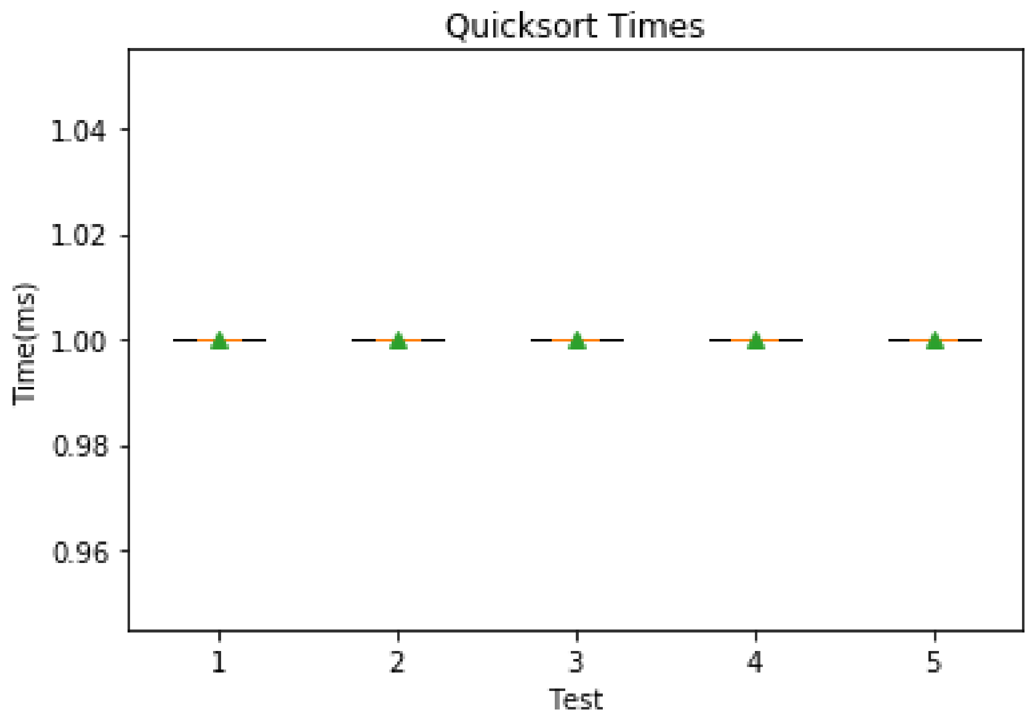
Test	Mean(ms)	Median(ms)
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
5	0.0	0.0

Quicksort – 20,000



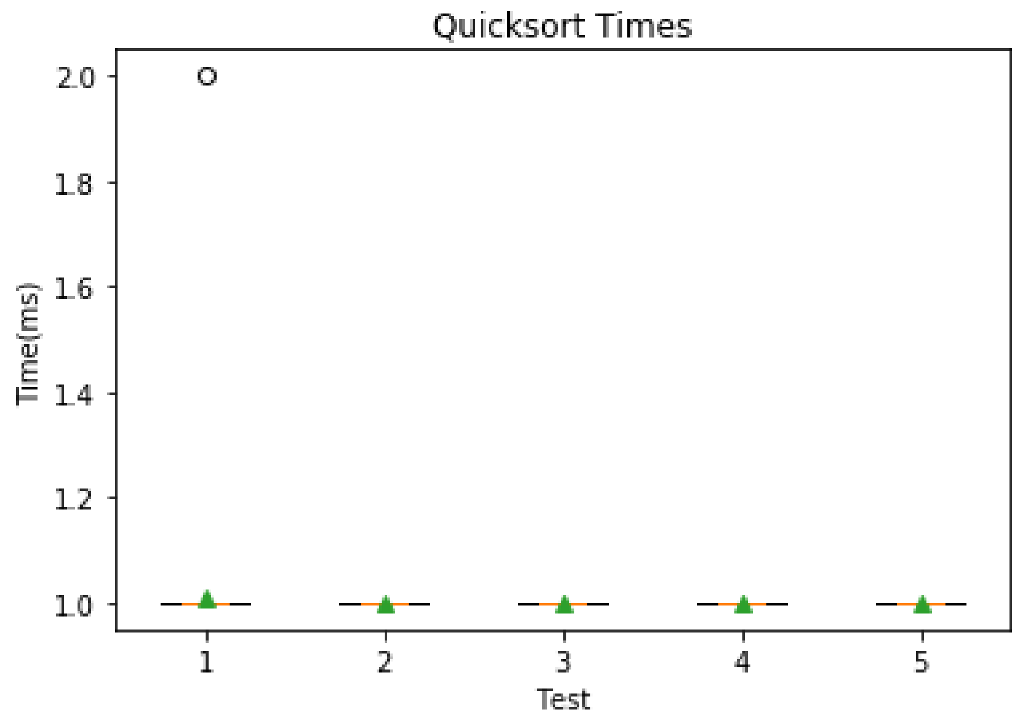
Test	Mean(ms)	Median(ms)
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
5	0.282828282828	0.0

Quicksort – 30,000



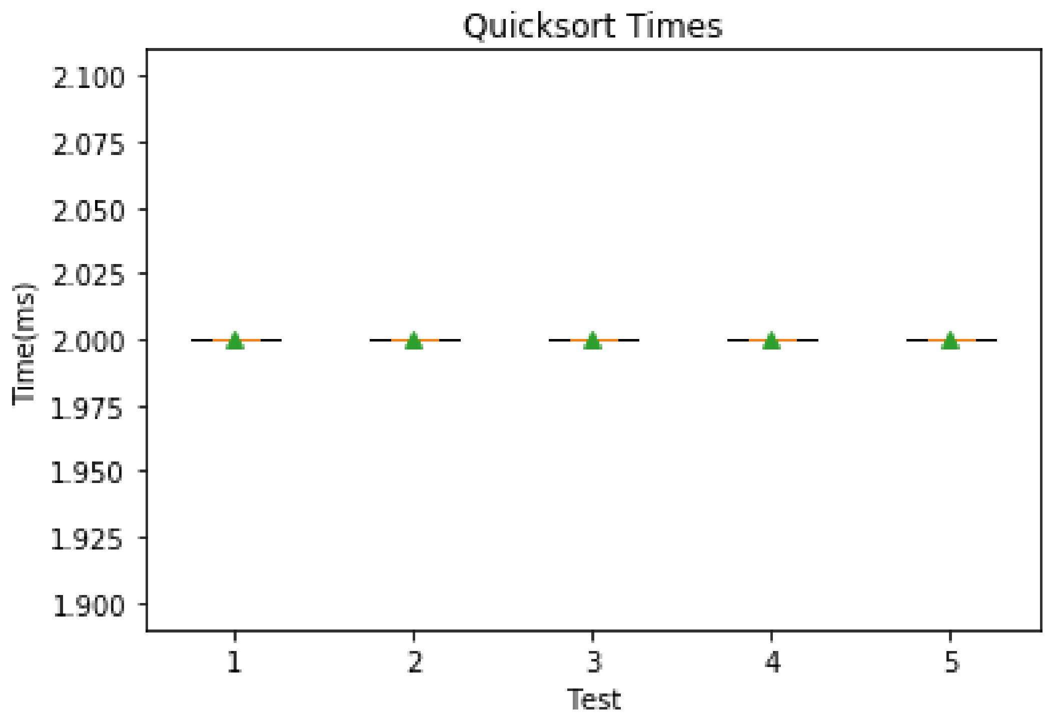
Test	Mean(ms)	Median(ms)
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	1.0	1.0

Quicksort – 40,000



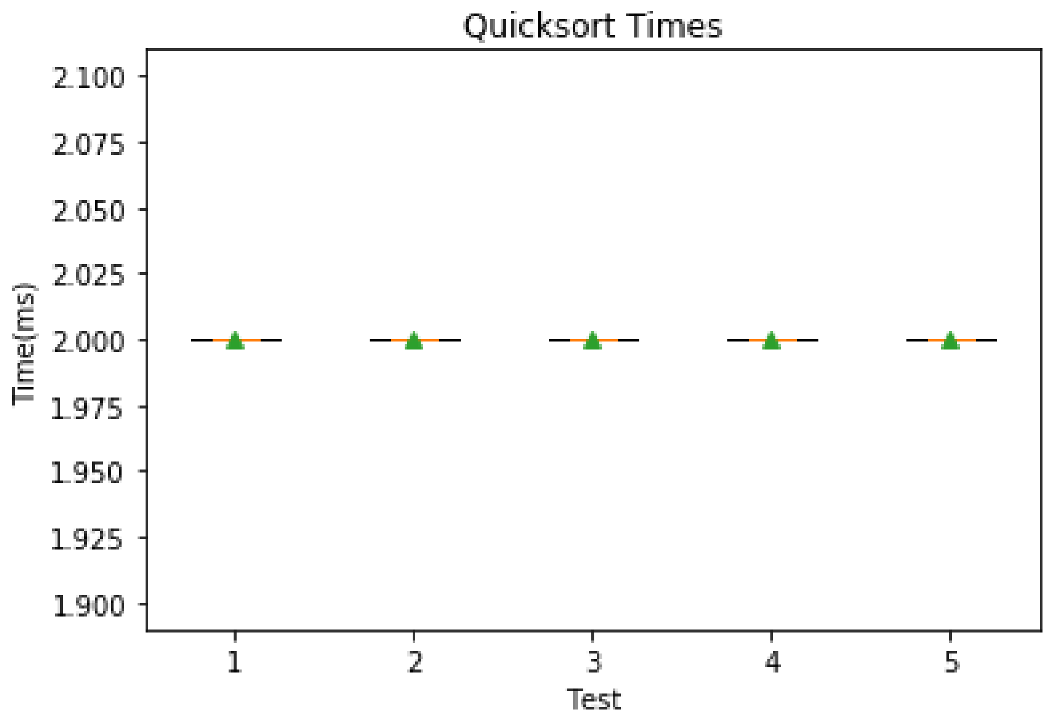
Test	Mean(ms)	Median(ms)
1	1.0101010101	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	1.0	1.0

Quicksort – 50,000



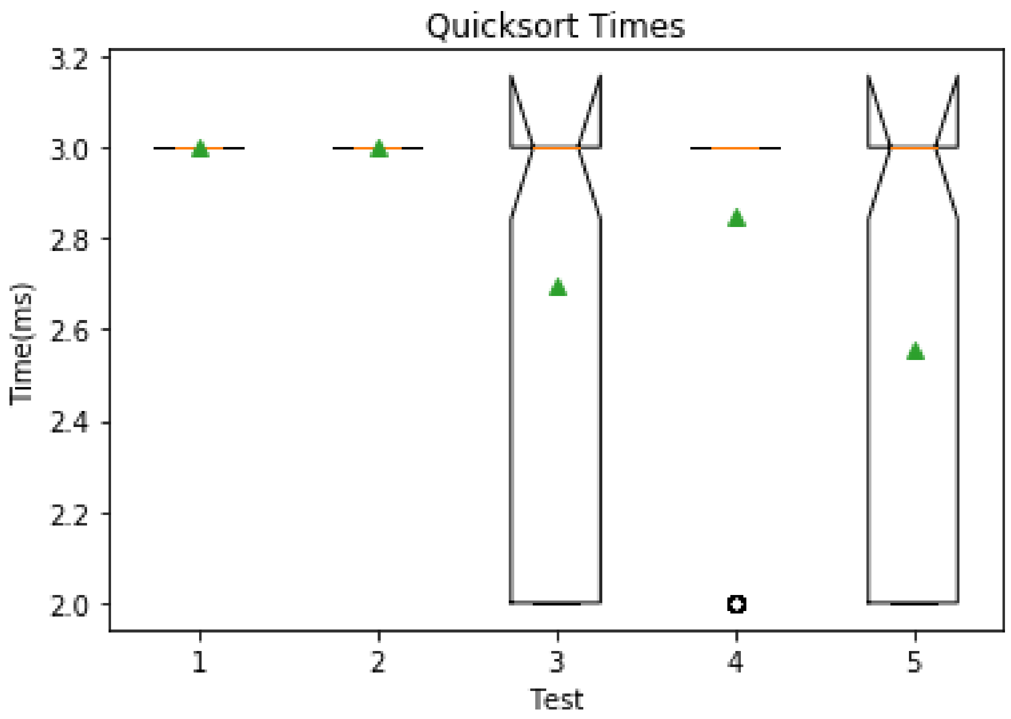
Test	Mean(ms)	Median(ms)
1	2.0	2.0
2	2.0	2.0
3	2.0	2.0
4	2.0	2.0
5	2.0	2.0

Quicksort – 60,000



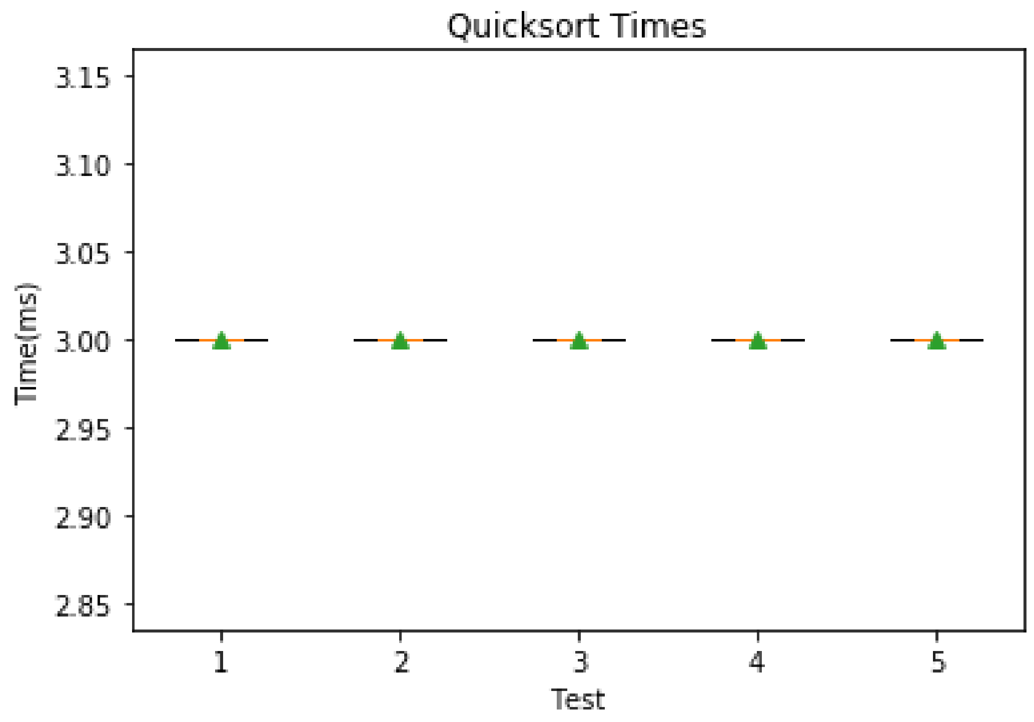
Test	Mean(ms)	Median(ms)
1	2.0	2.0
2	2.0	2.0
3	2.0	2.0
4	2.0	2.0
5	2.0	2.0

Quicksort – 70,000



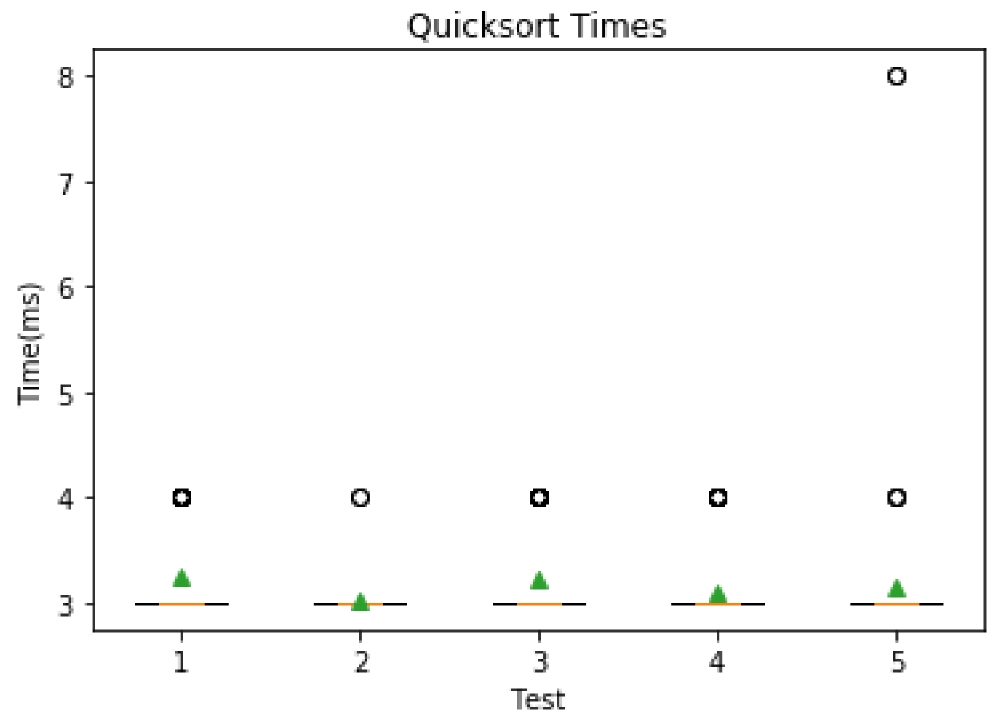
Test	Mean(ms)	Median(ms)
1	3.0	3.0
2	3.0	3.0
3	2.69696969697	3.0
4	2.84848484848	3.0
5	2.55555555556	3.0

Quicksort – 80,000



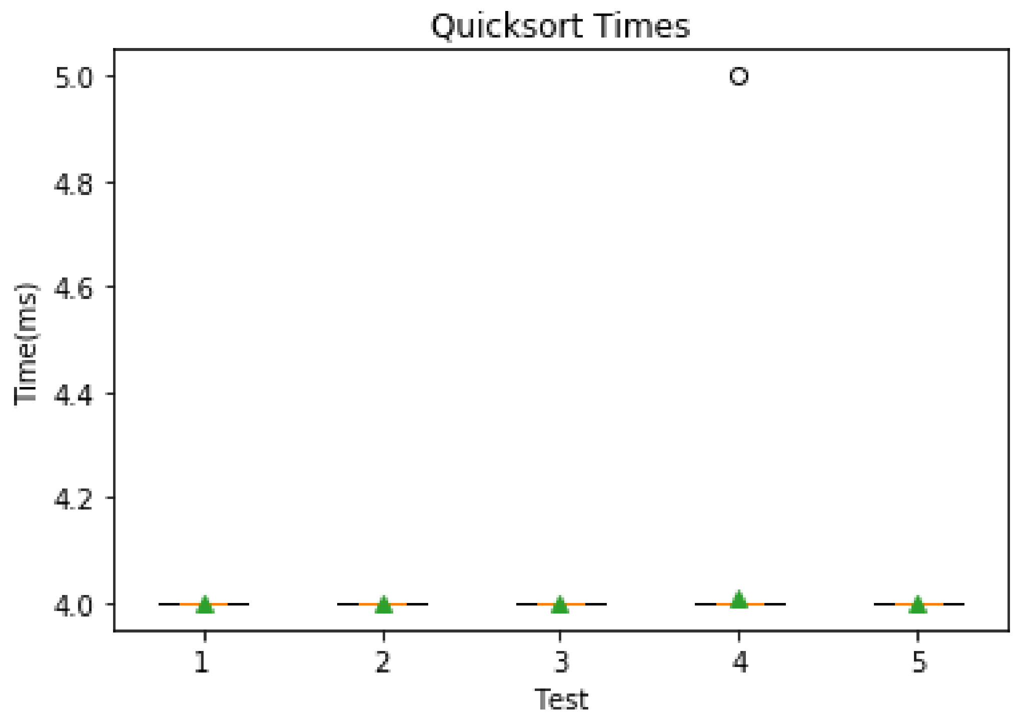
Test	Mean(ms)	Median(ms)
1	3.0	3.0
2	3.0	3.0
3	3.0	3.0
4	3.0	3.0
5	3.0	3.0

Quicksort – 90,000



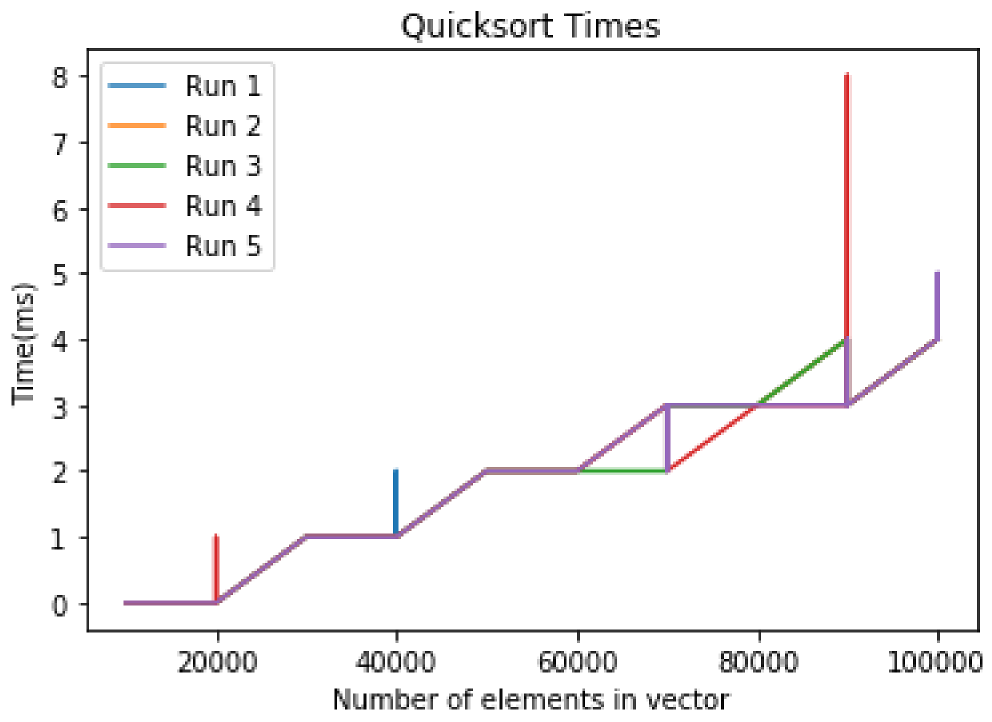
Test	Mean(ms)	Median(ms)
1	3.24242424242	3.0
2	3.0202020202	3.0
3	3.23232323232	3.0
4	3.09090909091	3.0
5	3.14141414141	3.0

Quicksort – 100,000

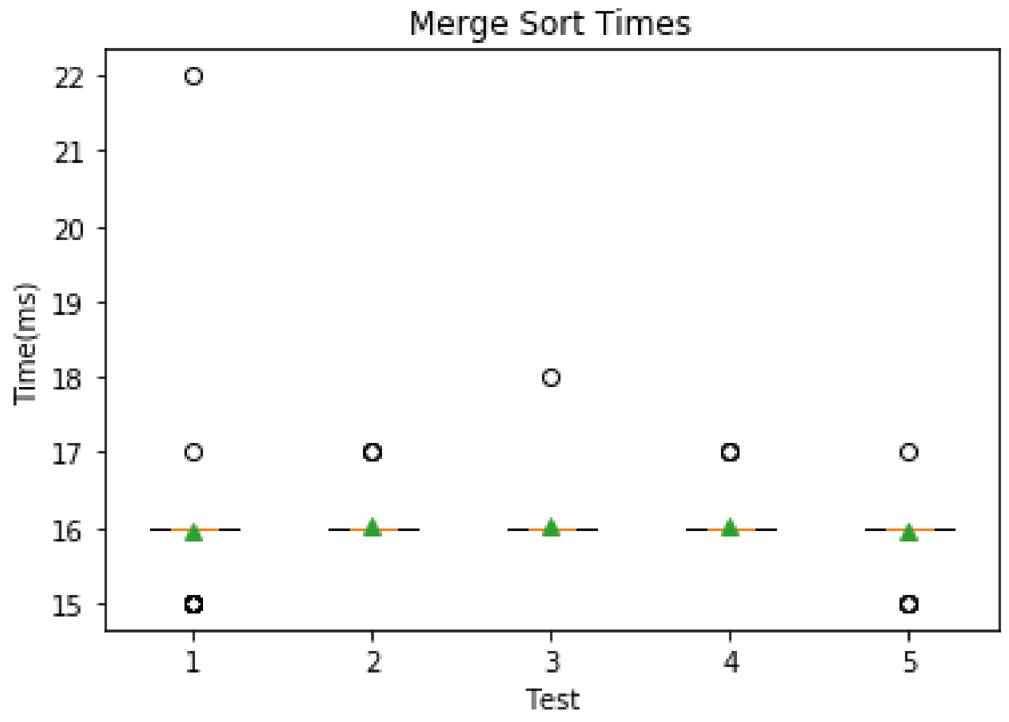


Test	Mean(ms)	Median(ms)
1	4.0	4.0
2	4.0	4.0
3	4.0	4.0
4	4.0101010101	4.0
5	4.0	4.0

Quicksort Run Comparison

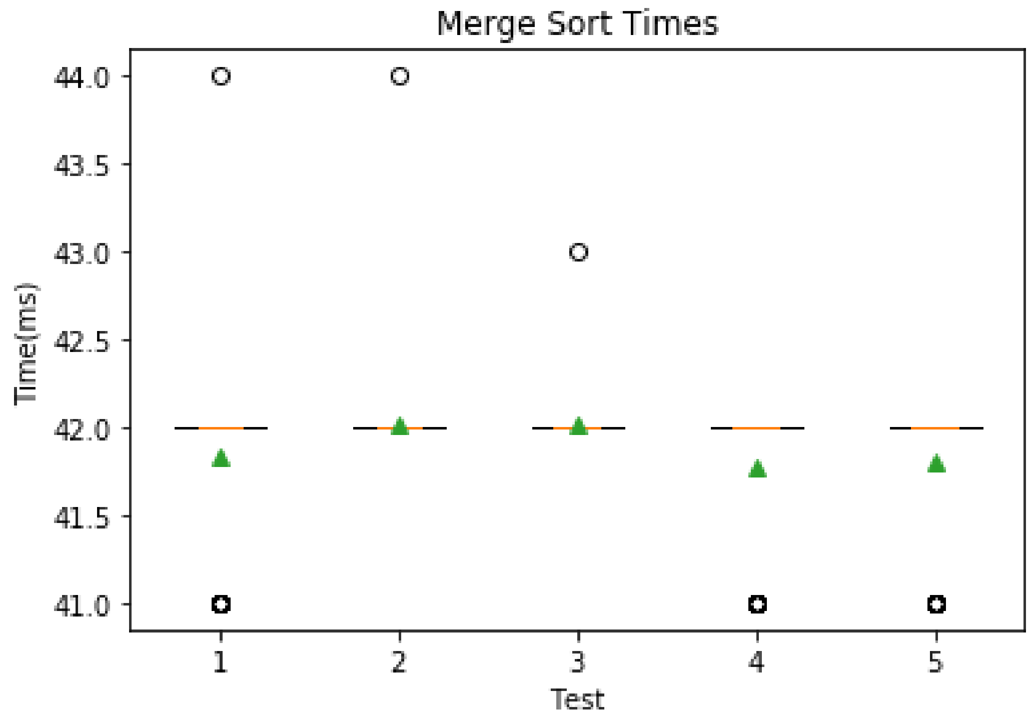


Merge Sort – 10,000



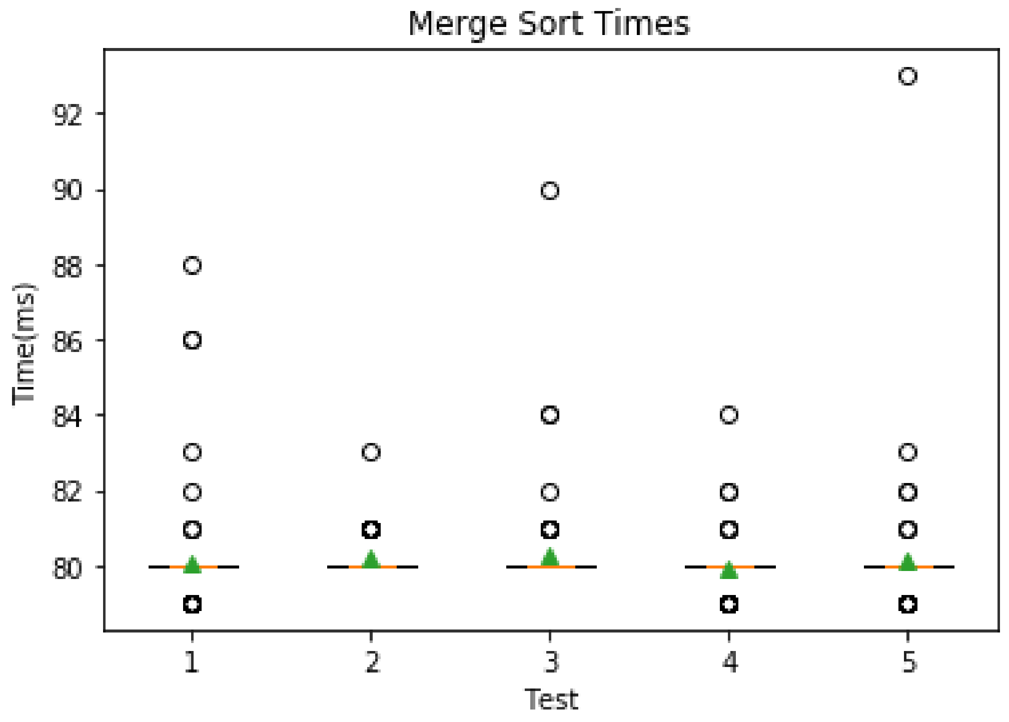
Test	Mean(ms)	Median(ms)
1	15.97	16.0
2	16.03	16.0
3	16.02	16.0
4	16.03	16.0
5	15.97	16.0

Merge Sort – 20,000



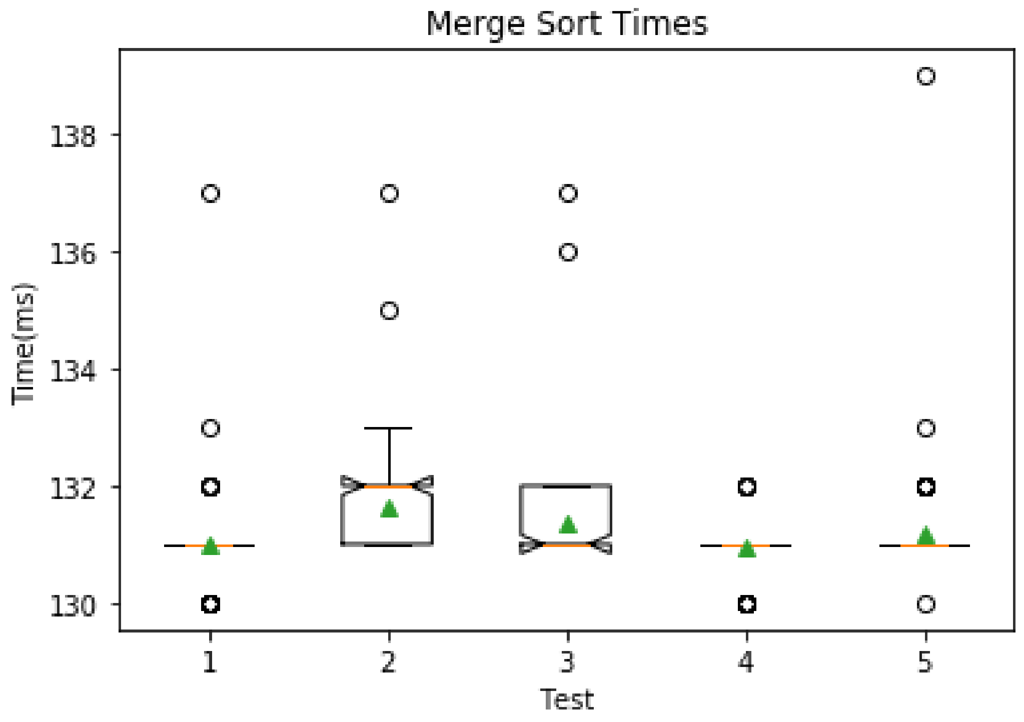
Test	Mean(ms)	Median(ms)
1	41.8383838384	42.0
2	42.0202020202	42.0
3	42.0101010101	42.0
4	41.7777777778	42.0
5	41.8080808081	42.0

Merge Sort – 30,000



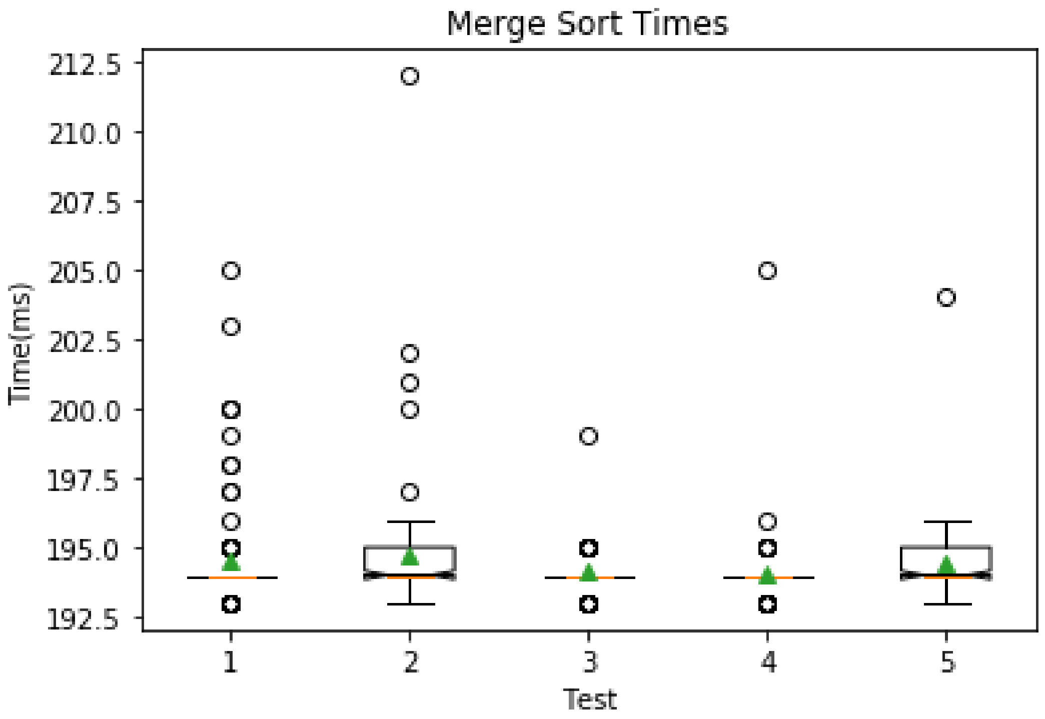
Test	Mean(ms)	Median(ms)
1	80.0404040404	80.0
2	80.1818181818	80.0
3	80.1818181818	80.0
4	79.9191919192	80.0
5	80.1111111111	80.0

Merge Sort – 40,000



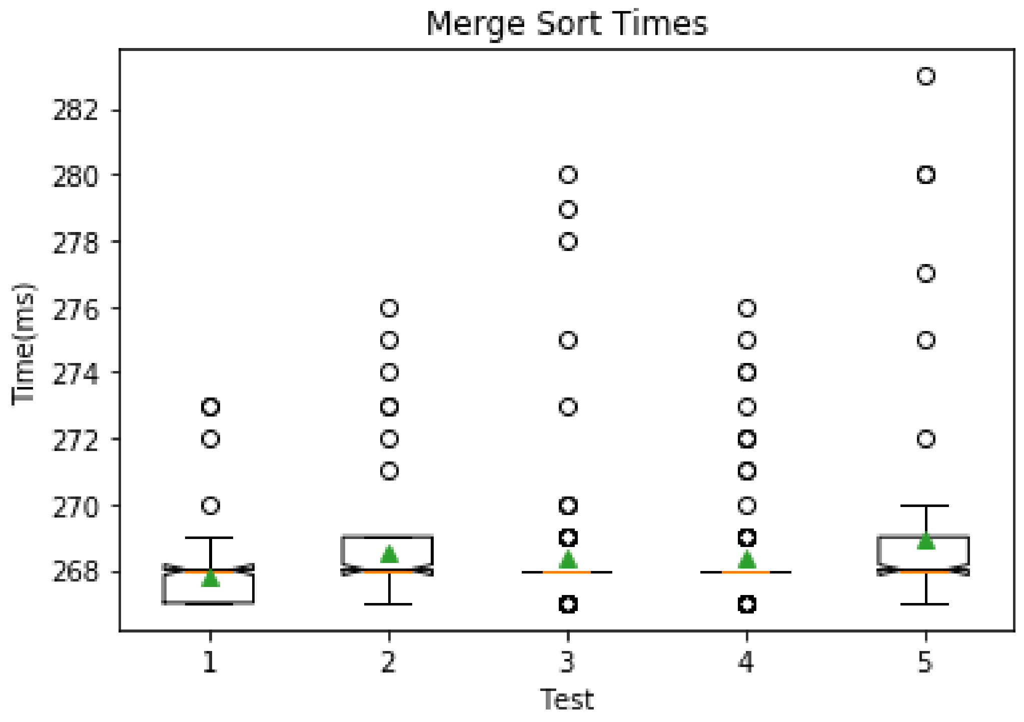
Test	Mean(ms)	Median(ms)
1	131.02020202	131.0
2	131.656565657	132.0
3	131.353535354	131.0
4	130.95959596	131.0
5	131.20202020	131.0

Merge Sort – 50,000



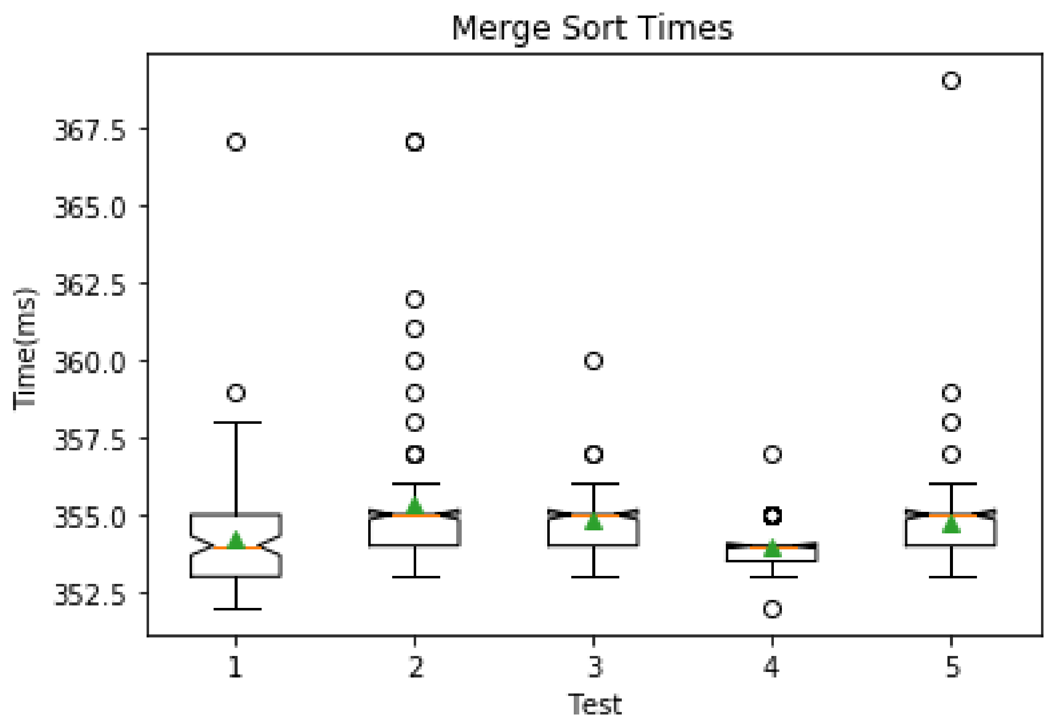
Test	Mean(ms)	Median(ms)
1	194.525252525	194.0
2	194.727272727	194.0
3	194.111111111	194.0
4	194.03030303	194.0
5	194.393939394	194.0

Merge Sort – 60,000



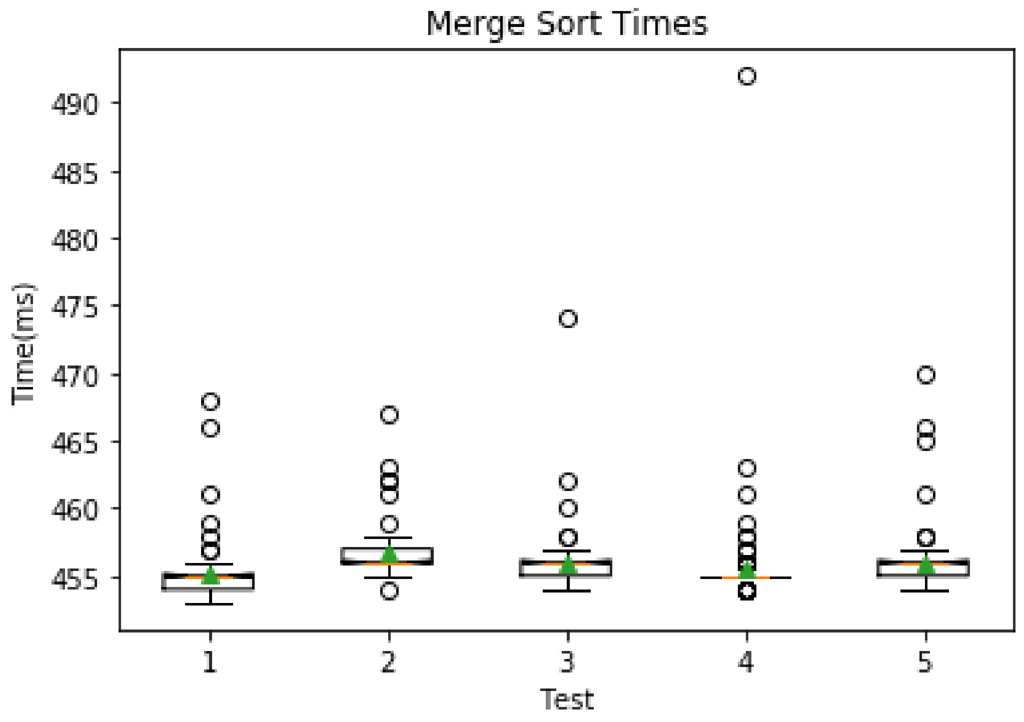
Test	Mean(ms)	Median(ms)
1	267.797979798	268.0
2	268.555555556	268.0
3	268.343434343	268.0
4	268.404040404	268.0
5	268.919191919	268.0

Merge Sort – 70,000



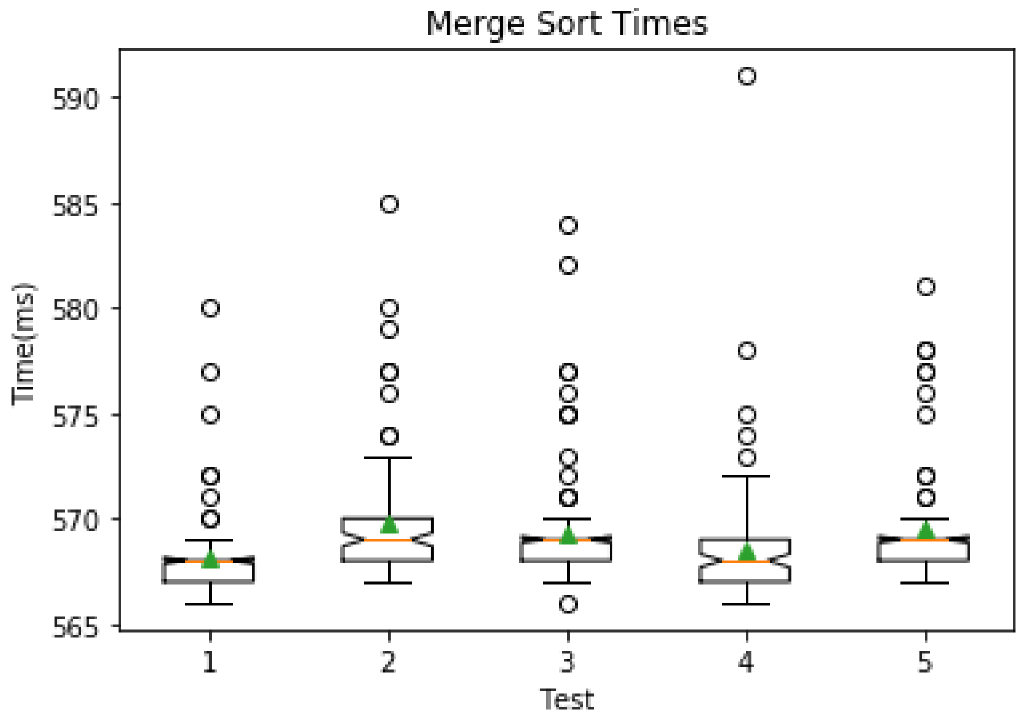
Test	Mean(ms)	Median(ms)
1	354.212121212	354.0
2	355.363636364	355.0
3	354.808080808	355.0
4	353.97979798	354.0
5	354.767676768	355.0

Merge Sort – 80,000



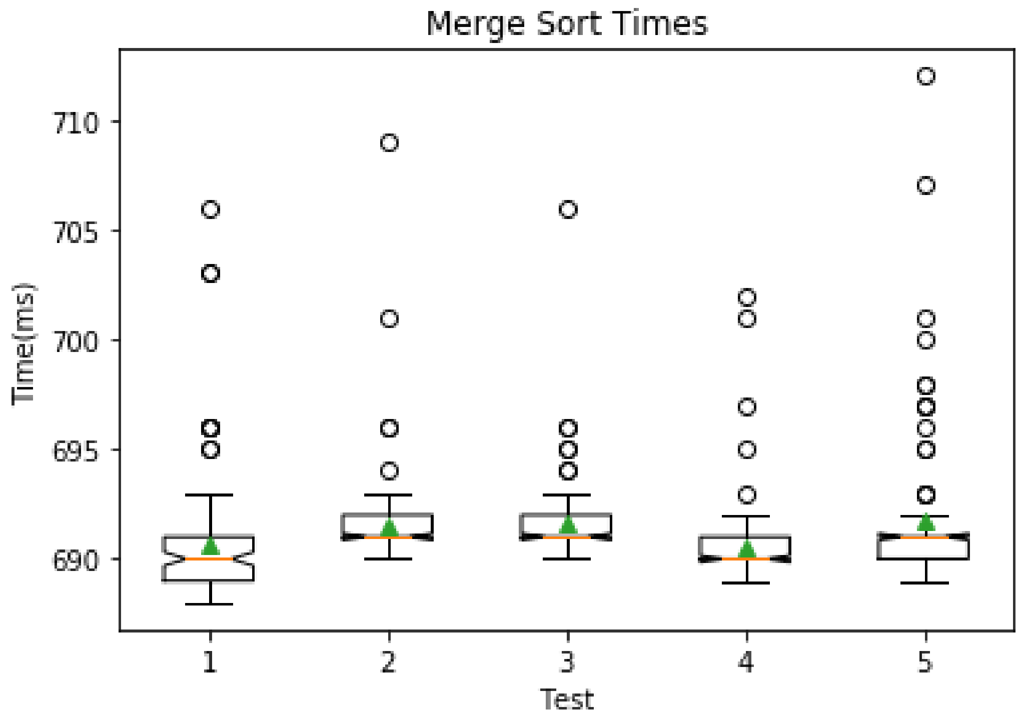
Test	Mean(ms)	Median(ms)
1	455.151515152	455.0
2	456.686868687	456.0
3	455.909090909	456.0
4	455.636363636	455.0
5	455.96969697	456.0

Merge Sort – 90,000



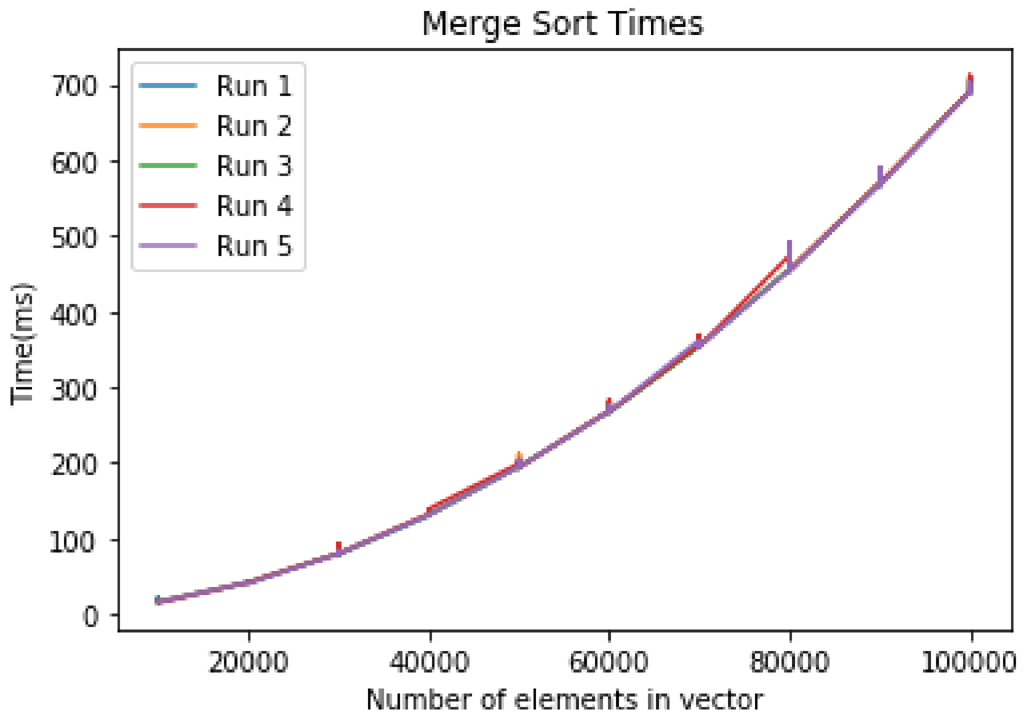
Test	Mean(ms)	Median(ms)
1	568.101010101	568.0
2	569.767676768	569.0
3	569.303030303	569.0
4	568.575757576	568.0
5	569.474747475	569.0

Merge Sort – 100,000

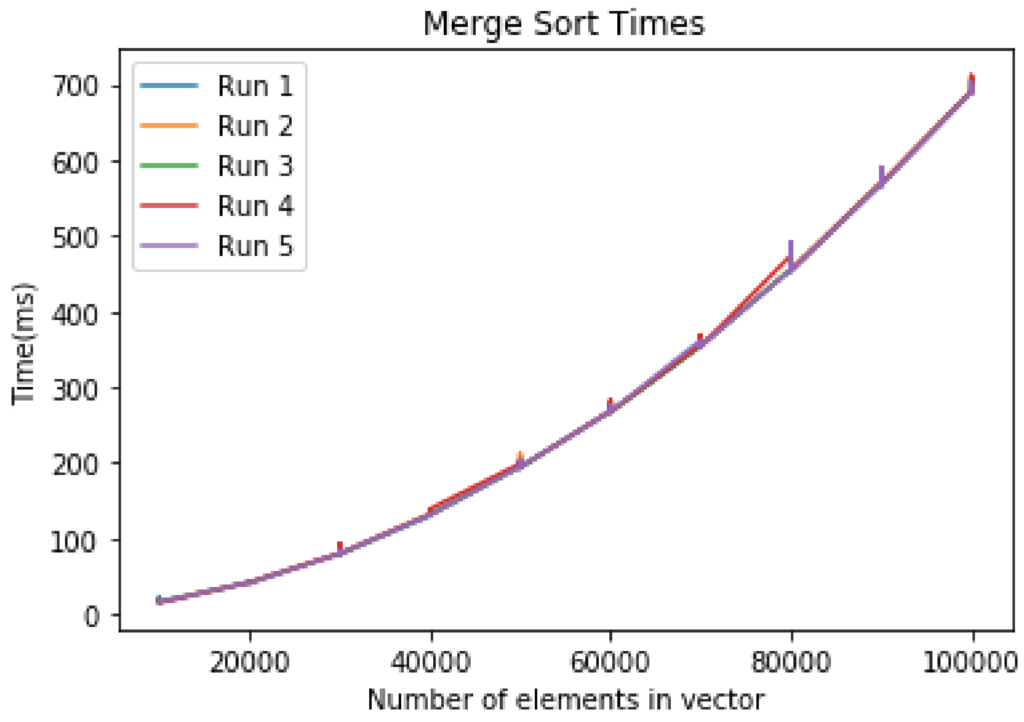


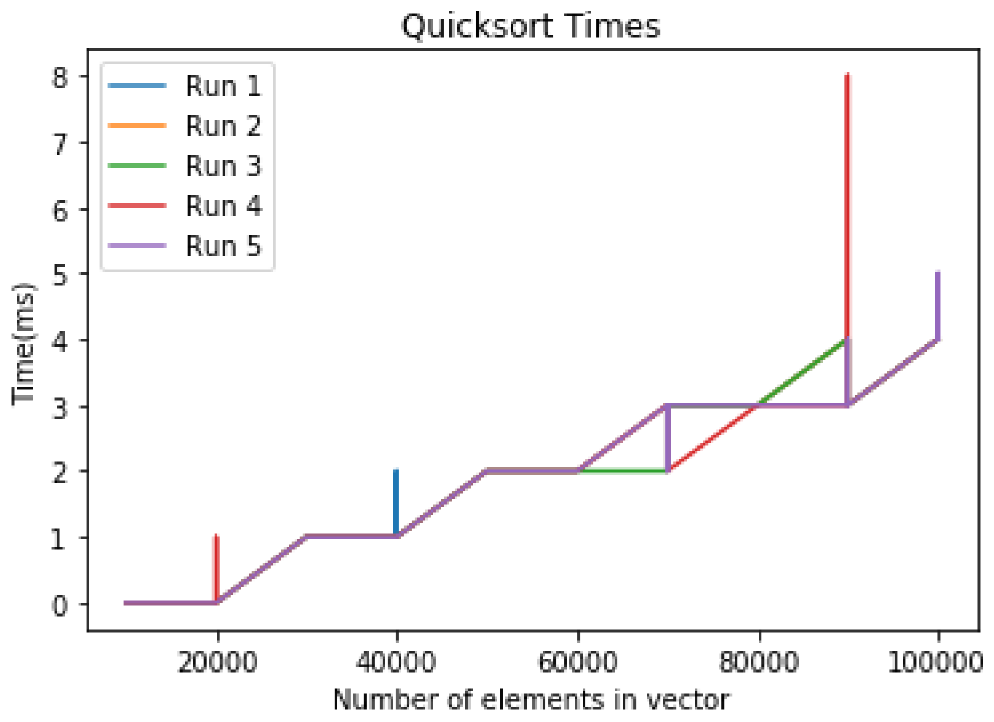
Test	Mean(ms)	Median(ms)
1	690.696969697	690.0
2	691.525252525	691.0
3	691.676767677	691.0
4	690.505050505	690.0
5	691.747474747	691.0

Merge Sort Run Comparison

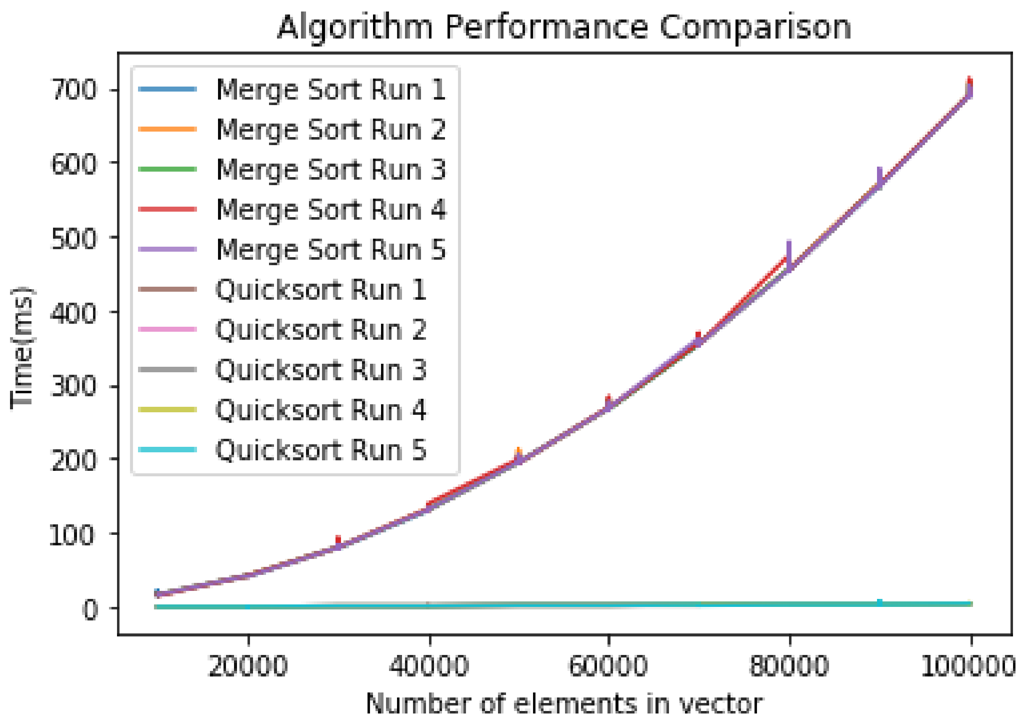


Algorithm Run Comparison





Quicksort vs. Merge Sort



Rank Sum	Statistic	P-Value
	38.72015463311832	0.0

Confidence Interval	Mean(ms)	+/-
Merge Sort	279.93400000000003	11.45899725200042

Confidence Interval	Mean(ms)	+/-
Quicksort	1.9259999999999999	0.0692712313501913

T-Test	Statistic	P-Value
	39.942285880875907	5.9447282982779241e-257

Theoretical vs. Results

- Quicksort

Resources

- Quicksort pseudocode - https://en.wikipedia.org/wiki/Quicksort (Utilising Hoare partition scheme).
- Merge Sort pseudocode – https://en.wikipedia.org/wiki/Merge_sort (Utilising Top-down implementation).
- SciPy – Statistics calculations, graphical representations.
- Adam Sampson – Lab 4 project which this submission is based off of.