Function	1000 size input (ms)	31k input (ms)	1 Million (ms)
Standard	0 0 0 Avg: 0	6 6 6 Avg: 6	257 254 256 Avg: 255.67
Merge	1 1 1 Avg: 1	59 59 59 Avg: 59	2138 2124 2143 Avg: 2135
In place Merge	0 0 0 Avg: 0	12 12 12 Avg: 12	457 455 455 Avg: 455.67
½ Selection	3 3 3 Avg: 3	3427 3469 3461 Avg: 3452.33	N/A (Too large)
½ Heap	0 0 0 Avg: 0	3 3 3 Avg: 3	167 165 166 Avg: 166
Quick Select	0 0 0 Avg: 0	30 30 30 Avg: 30	30 30 30 Avg: 30
Worst Case Quick Select	96 96 96 Avg: 96		

Generally speaking, it is interesting that my runtimes were quite short, could it be because I have a very good CPU? Quick Select definitely lives up to its name, considering it has the shortest times out of all the inputs, barring the worst case of course. Although In Place Merge Sort avoids using extra space and memory, it was still more efficient than Merge Sort. Could this also be due to having efficient PC parts?

Standard Sort:

- Time Complexity: O(n log n)
    - Combines QuickSort, HeapSort, and Insertion Sort for optimal performance.
- Space Complexity: O(log n)
    - Requires stack space for recursive calls in QuickSort.

MergeSort:

- Time Complexity: O(n log n)
    - Divides and conquers by splitting and merging.
- Space Complexity: O(n)
    - Requires additional space for temporary arrays during merging.

InPlaceMergeSort:

- Time Complexity: O(n log n)
    - Similar to MergeSort but optimized for in-place sorting.
- Space Complexity: O(1)
    - Sorts the array in place, reducing space requirements.

1/2 Selection Sort:

- Time Complexity: Approximately O(n^2/2)
    - Iteratively selects the next smallest/largest element.
- Space Complexity: O(1)
    - No additional space is needed beyond the input array.

1/2 Heap Sort:

- Time Complexity: Approximately O(n log n/2)
    - Constructs a heap and then sorts half of the elements.
- Space Complexity: O(1)
    - Heap is built in place in the array.

Quick Select:

- Time Complexity: Average O(n), Worst O(n^2)
    - Quick partitioning to find kth element; worst case when poor pivot.
- Space Complexity: O(log n)
    - Stack space for recursion; can be O(n) in the worst case.

WorstCaseQuickSelect:

- Time Complexity: O(n^2)
    - Worst-case performance with poor pivot choice in a reverse-sorted list.
- Space Complexity: O(n)
    - Due to recursive calls, especially in the worst-case scenario.