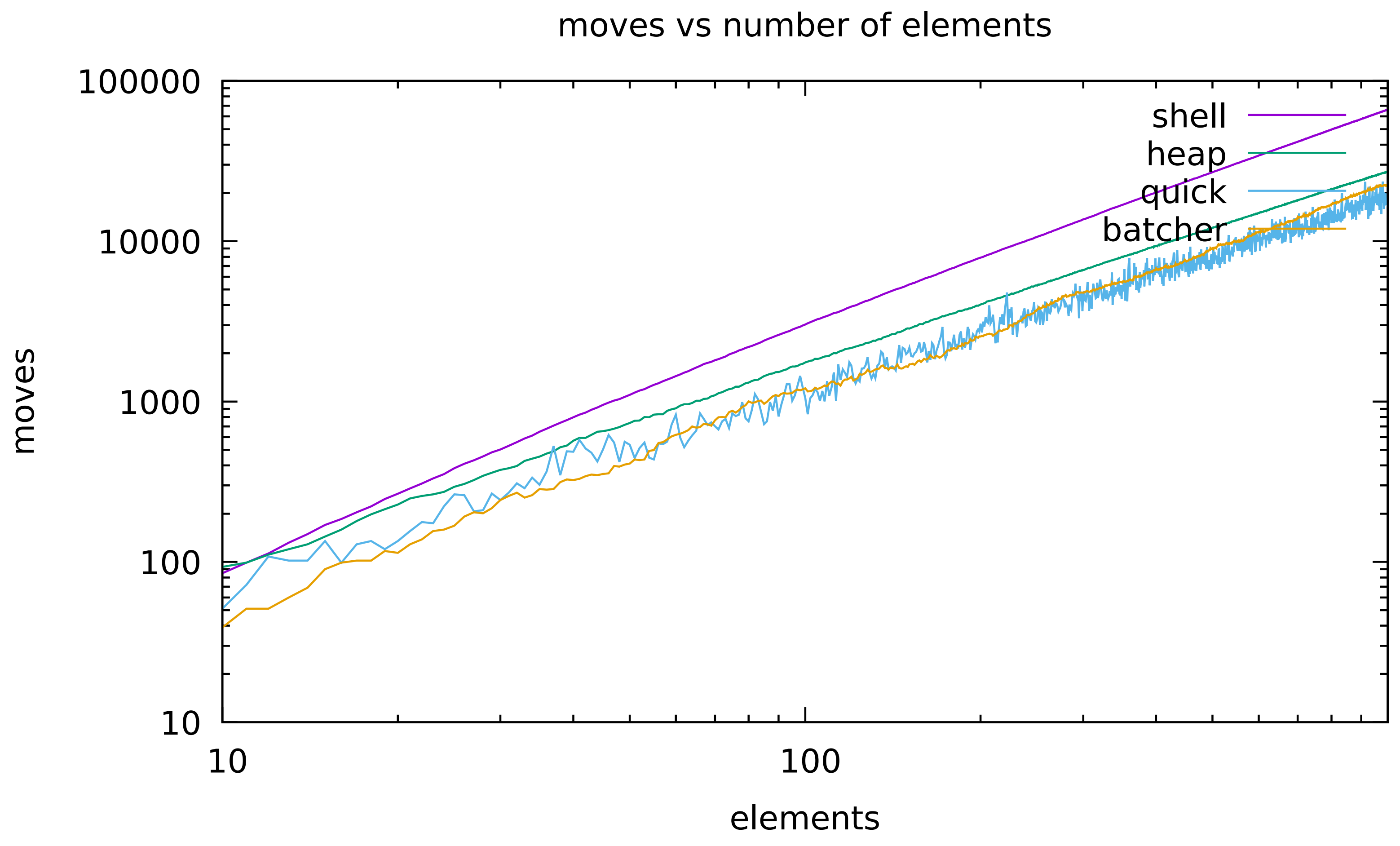
ASSIGNMENT 3 WRITEUP

Chris Moon

February 2023

1 GRAPHS: see below


moves vs number of elements
100000
10000
1000
100
10
10
100
elements
moves
shell
heap
quick
batcher

MOVES VS ELEMENTS

The above graph displays the number of moves each sorting algorithm requires to sort an array, as the number of elements in the array to be sorted increases.

The number of moves refers to the number of times an array element is shifted in the sorting process.

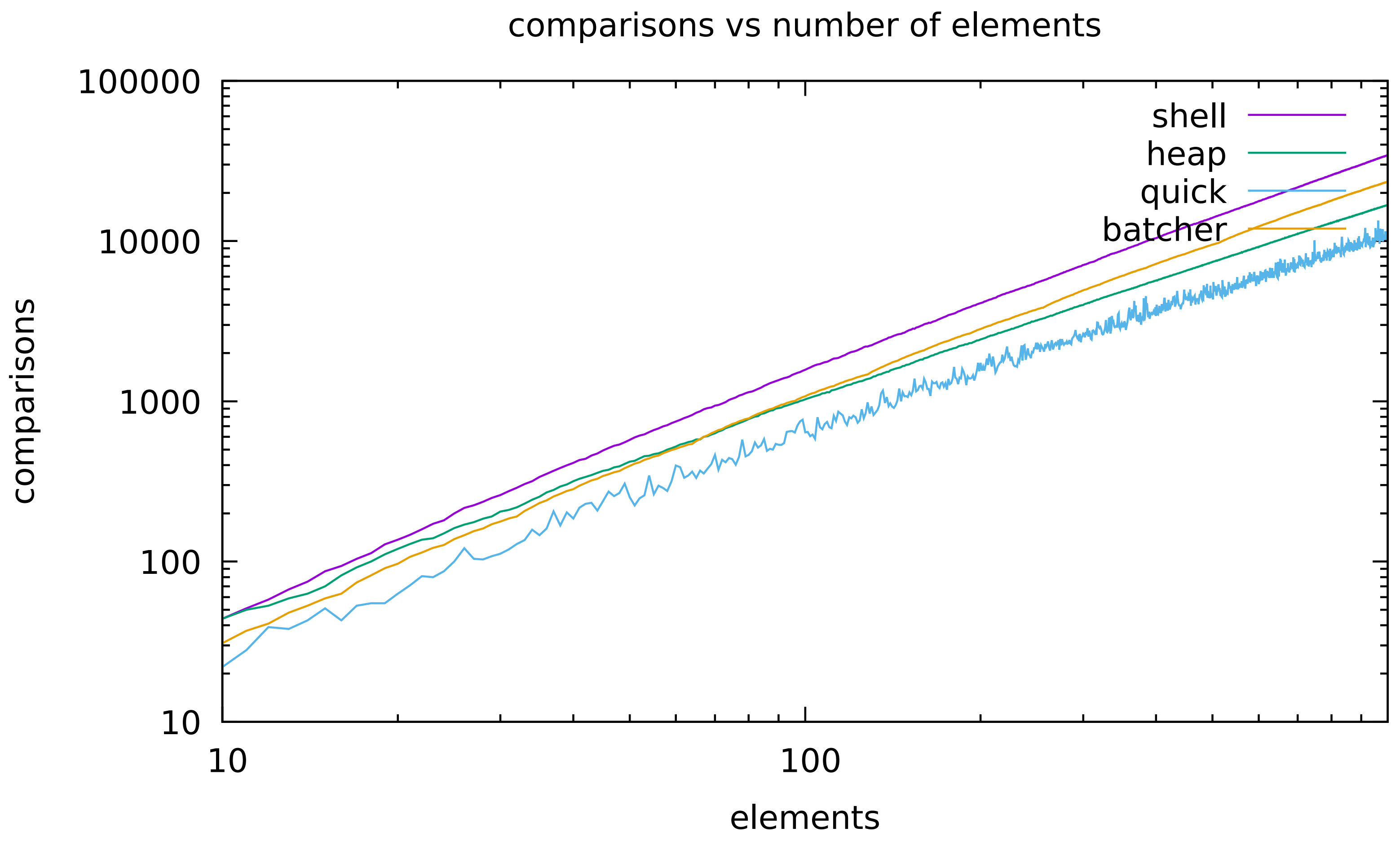
In general, the more moves, the less efficient the sorting algorithm is, and the longer it will take to sort an array.

As shown by the graph, for all four sorting methods (shell sort, heap sort, quick sort, and Batcher's merge sort), the number of moves increases linearly with elements.

Also shown, shell sort consistently had the highest number of moves. This is probably due to shell sort's exponential time complexity of n squared, compared to the other three sorting methods, which have a faster time complexity of nlog(n).

Quick sort was also shown to be the least linear pattern. This is most likely due to quick sort running better on partially sorted arrays than the other three methods, but running worse when the array is very unsorted.

Basically, quick sort has a faster best-case scenario but a slower worst-case scenario than the other graphs.


comparisons vs number of elements
comparisons
elements
100000
10000
1000
100
10
10
100
shell
heap
quick
batcher

COMPARISONS VS ELEMENTS

The above graph displays the number of comparisons each sorting algorithm requires to sort an array, as the number of elements in the array to be sorted increases.

The number of comparisons refers to the number of times two array elements are compared against each other in a sort.

Shell sort again is the least efficient, also probably due to its time complexity just being worse than the other three sorting methods.

Batcher has more comparisons relative to moves than quick and heap. This is probably due to Batcher exponentially calling a comparison function.