Programming Assignment – 4

Maulik Durani & Ryan Majd

Experiment-1:

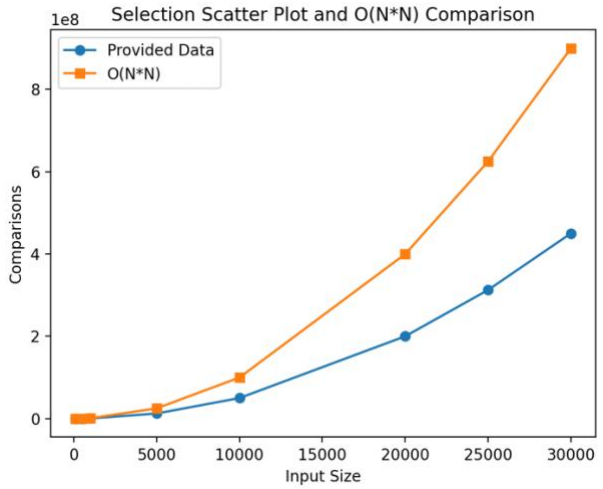
Algorithm	Input Type	# of Comparisons	Comments
SelectionSort	ordered.txt	49995000	For any of the inputs, the number of comparisons indicates a time complexity of N*N
SelectionSort	random.txt	50073371	
SelectionSort	reverse.txt	25000000	
MergeSort	ordered.txt	69008	This mergeSort function has a time complexity of N*log(N)
MergeSort	random.txt	120414	
MergeSort	reverse.txt	64608	
HeapSort	ordered.txt	302541	This heapSort function has a time complexity of N*log(N)
HeapSort	random.txt	280738	
HeapSort	reverse.txt	260316	
QuickSort-fp	ordered.txt	49995000	This quickSort-fp function has a time complexity of N*N
QuickSort-fp	random.txt	239041	
QuickSort-fp	reverse.txt	25000000	
QuickSort-rp	ordered.txt	62697	This quickSort-rp function has a time complexity of N*log(N)
QuickSort-rp	random.txt	79674	
QuickSort-rp	reverse.txt	115610	

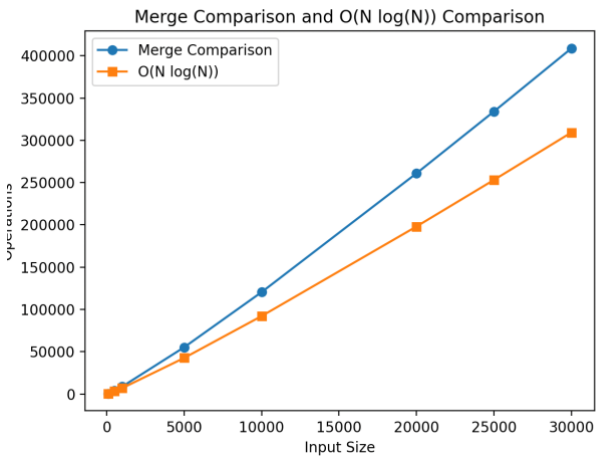
a) Yes. The long numComparisons in order to store the number of comparisons that each of the sorting methods do.

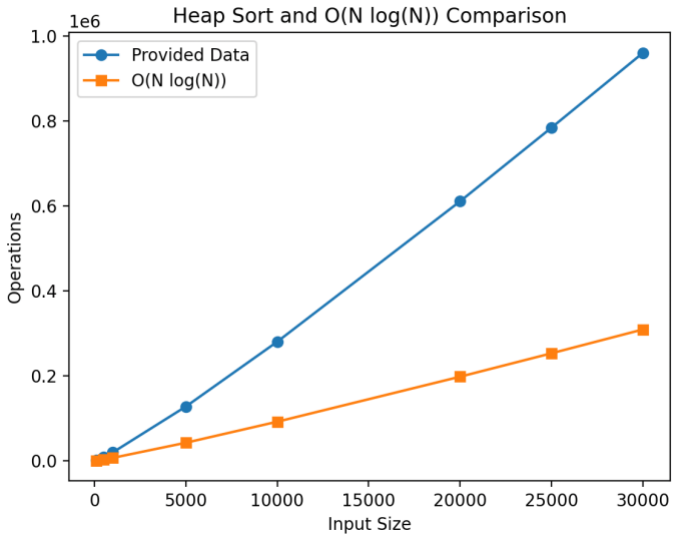
b) Either for reverse or ordered (sorted) input files, merge sort and heat sort are the most effective. For randomly sorted input files, quick-sort-rp is the most effective.

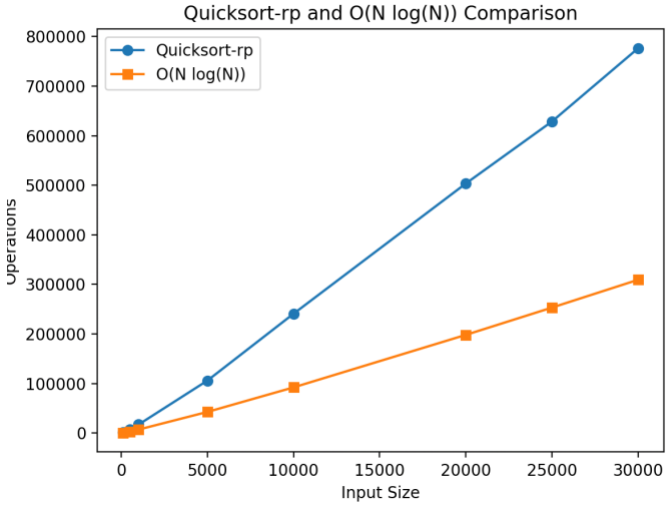
Experiment-2:

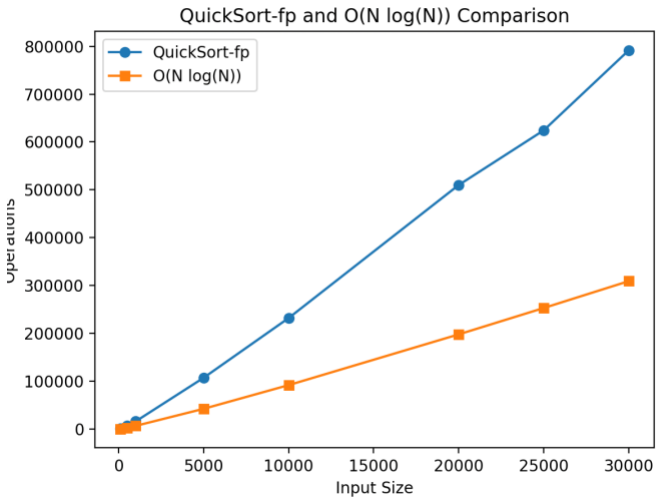
Algorithm	100	500	1000	5000	10000	20000	25000	30000
Selection	5280	127165	504972	12532759	50070432	200151296	312691408	450231228
Merge	545	3858	8719	55208	120431	260926	334099	408761
Heap	1155	8686	19794	127831	280482	610990	784418	960345
QuickSort-fp	919	7074	16272	107427	232238	510136	624378	791523
QuickSort-rp	988	7171	17002	105844	240740	503554	628708	776444

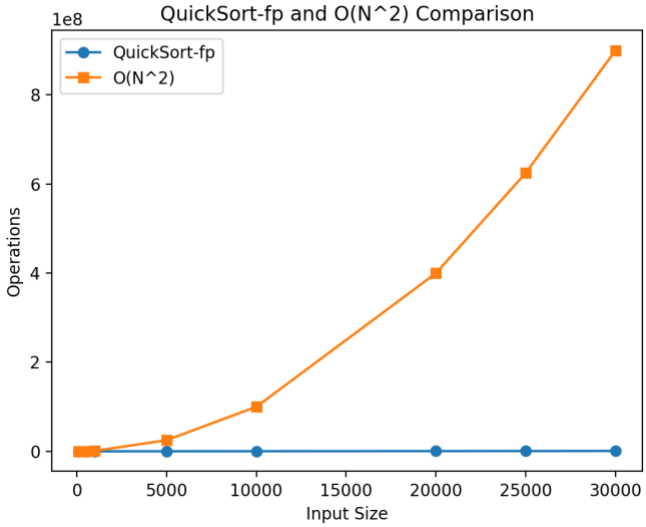












b.) Discussion:

Selection Sort and Merge Sort:

When the scatterplots for selection sort and merge sort closely follow their respective Big-O complexities, it confirms that the experimental results align well with theoretical expectations. This alignment is a positive indication that these algorithms perform as expected for the given datasets.

Heap Sort and Quicksort:

When the data for heap sort and quicksort starts at the same point but significantly deviates as the dataset size increases (e.g., for n > 10K), it signals potential inefficiencies or suboptimal implementations for larger datasets. Quicksort and heap sort are generally expected to perform well, with O(n log n) time complexity.

Randomized Pivot (rp) and First Pivot (fp):

The data for rp and fp starts similarly but becomes unoptimized for larger datasets, it indicates that the randomization strategy or pivot selection is not delivering the expected average-case time complexity improvements.