Exercise 2 Searching and Sorting (Report)

1. **Selection Sort** - Average Complexity: Θ(n^2)
Theoretical complexity & performance analysis: Selection is the slowest sorting method, because we need to loop till the end to find the smallest element then swap it with the front. We need to do this for n times, it takes O(n^2) time.

2. **Insertion Sort** - Average Complexity: Θ(n^2)

3**. Quick Sort** - Average Complexity: Θ(nlogn)
Theoretical complexity & performance analysis: We loop through the list to reorder the list by elements smaller than pivot, pivot, elements greater than pivot.

4. **Merge Sort** - Average Complexity: Θ(nlogn)
Theoretical complexity & performance analysis: Merging two arrays takes O(n) time. We do this logn times, which is O(nlogn) runtime. Worst case runtime of merge sort is O(nlogn), so it is generally faster than quick sort. But because we need extra memory to merge two arrays, it takes O(n) memory to do merge sort.

5. **Heap Sort** - Average Complexity: Θ(nlogn)

	Mean Time	Standard Deviation
Selection Sort*	59.07mins	38.51mins
Insertion Sort*	14.64mins	30.69sec
Quick Sort	3.932secs	39.63msec
Merge Sort	641.78msec	4.56msec
Heap Sort	1012.30msec	13.25msec

*(avg of 6 runs)
Conclusions:
Selection sort takes the maximum time followed by Insertion sort, both of which take minutes to sort the 126070 length array. Quick sort takes couple of seconds while Heap sort takes about a second and merge sort takes ~600mseconds for sorting. Merge sort is the fastest sort given this large size data.

Sorting	**Best**	**Avg**	**Worst**
Selection sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
Insertion sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Merge sort	$\Omega(n\log(n))$	$\Theta(n\log(n))$	$O(n\log(n))$
Quick sort	$\Omega(n\log(n))$	$\Theta(n\log(n))$	$O(n^2)$
Heap sort	$\Omega(n\log(n))$	$\Theta(n\log(n))$	$O(n\log(n))$