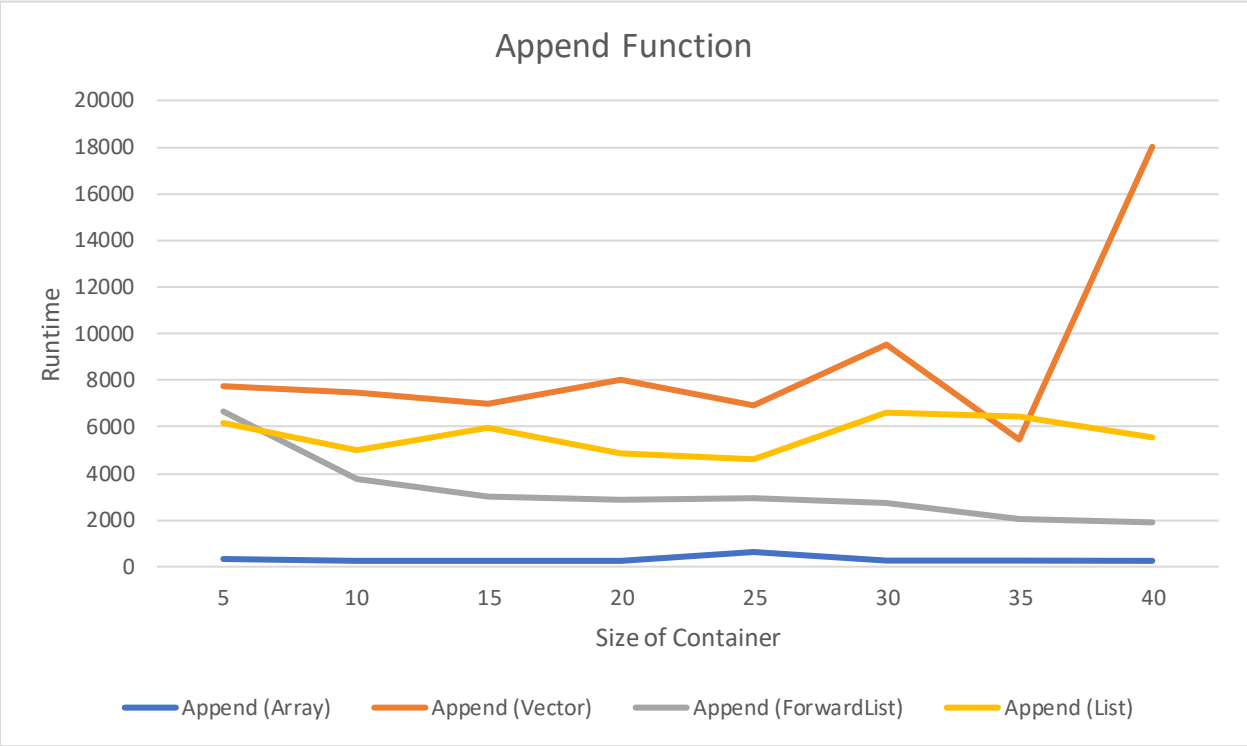
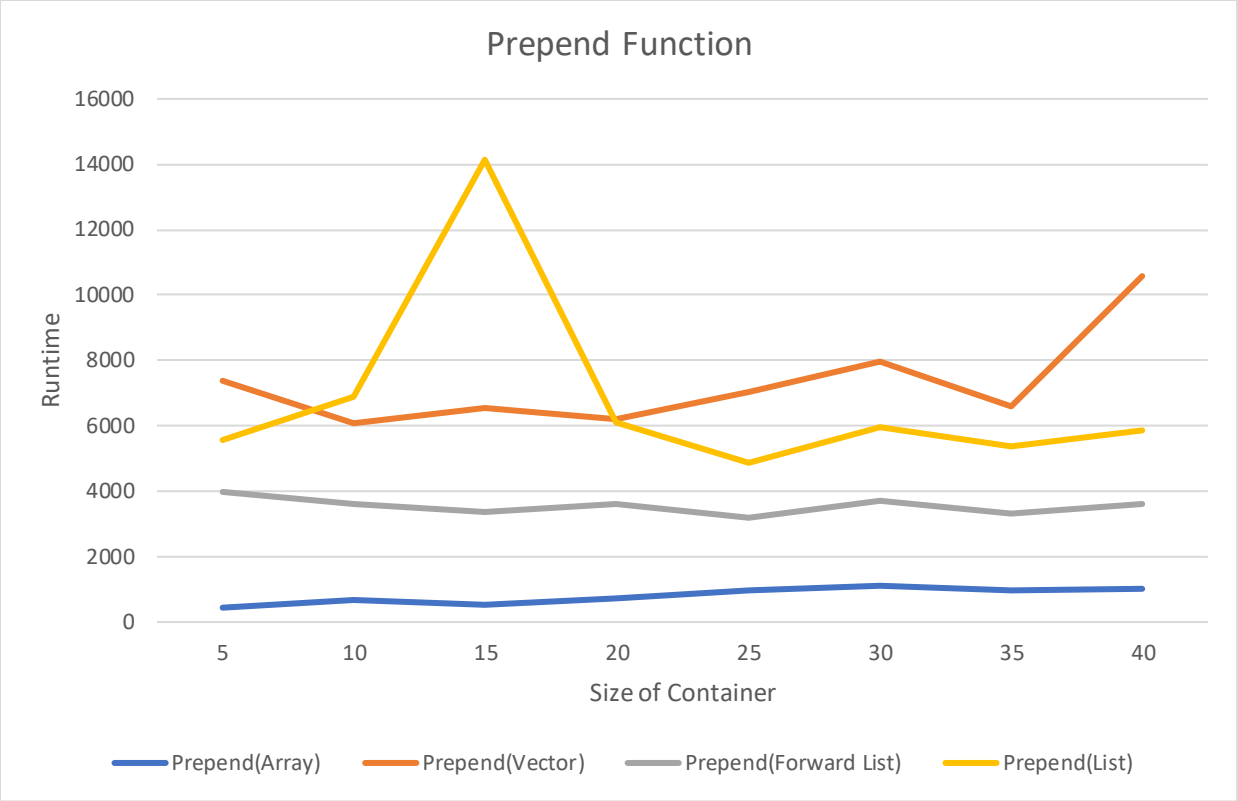
Functionality Benchmarks of Data Structure Operations:

In all cases, there may be fluctuations in data because of the inconsistency of the CPU clock.



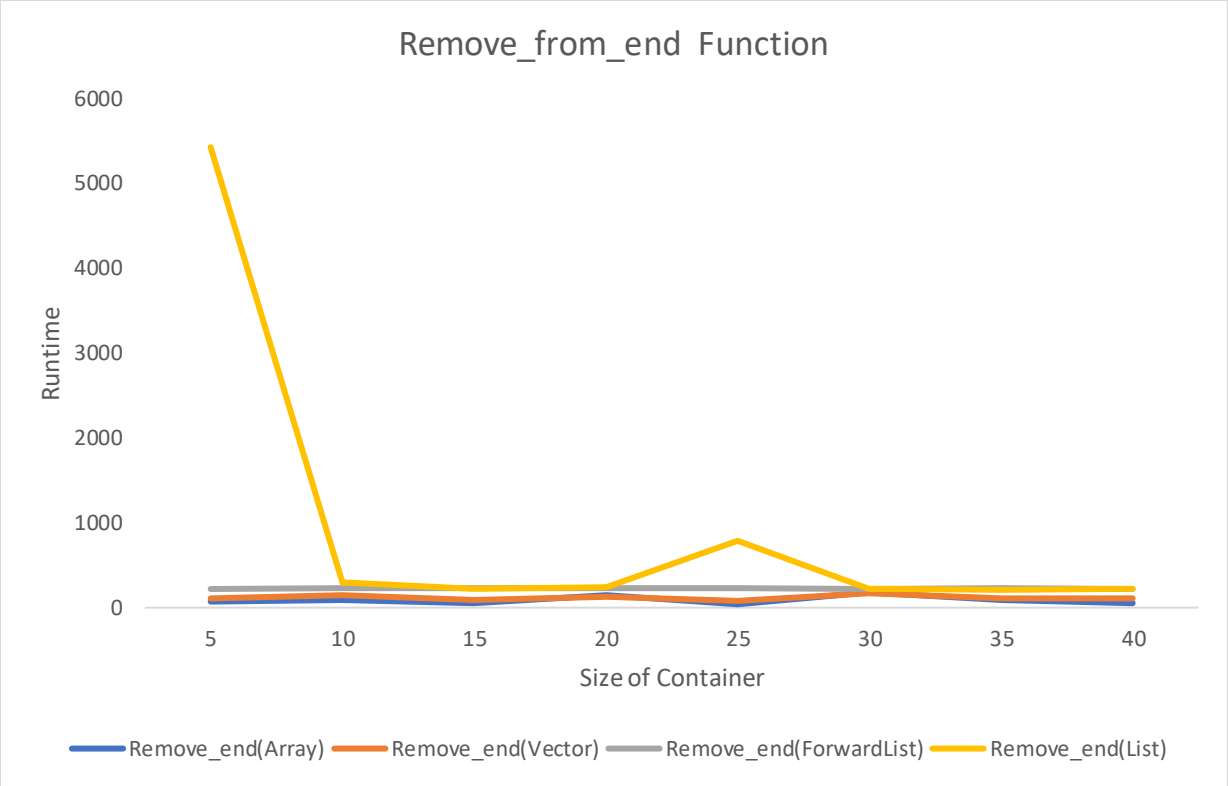
Size of Container	Append (Array)	Append (Vector)	Append (ForwardList)	Append (List)
5	310	7740	6660	6180
10	250	7480	3790	5021
15	250	7010	3010	5930
20	250	8020	2860	4850
25	630	6940	2920	4610
30	260	9540	2720	6610
35	280	5450	2020	6430
40	250	18010	1900	5570

Review: From the graph, it has been shown how array takes less run time than others where vector has taken the highest. Array takes constant time to append, so the graph has shown it to have constant runtime.



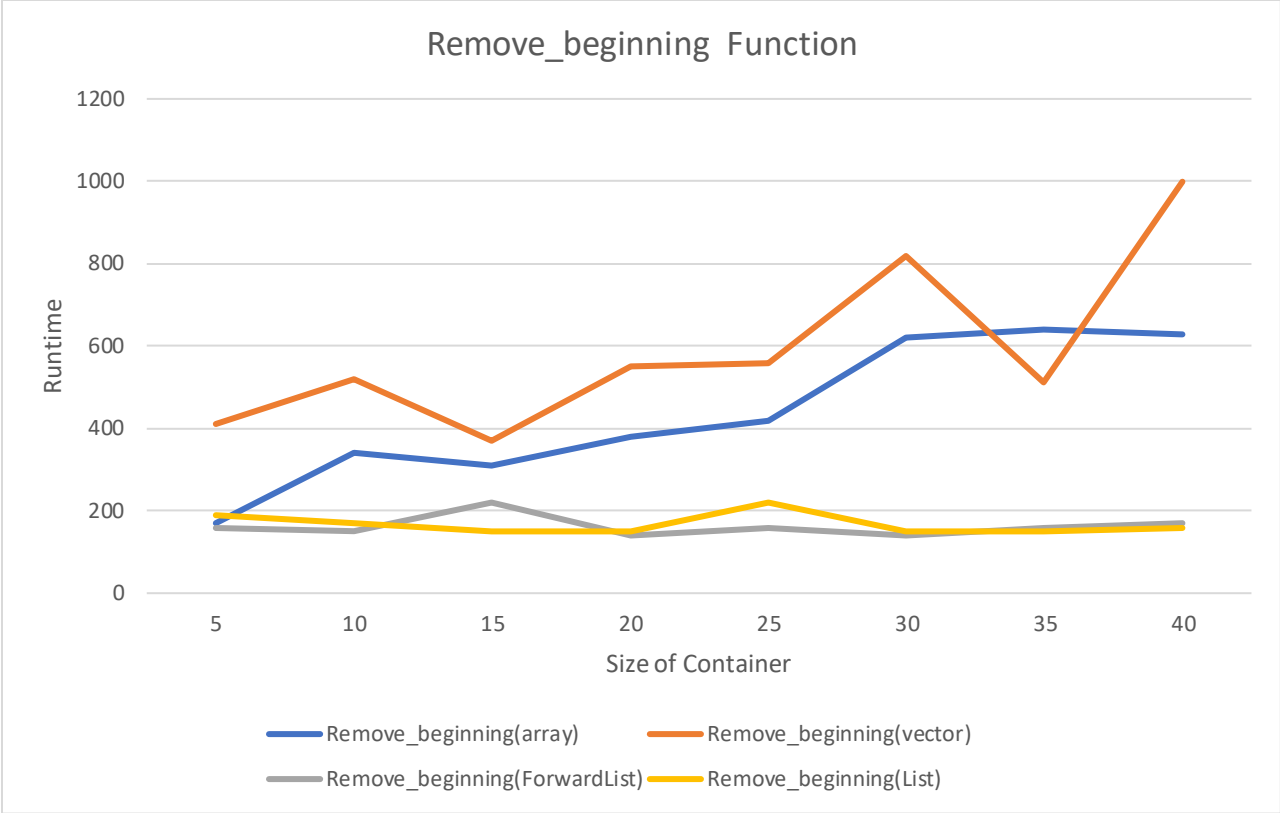
Size of Container	Prepend(Array)	Prepend(Vector)	Prepend(Forward List)	Prepend(List)
5	440	7390	3980	5570
10	660	6080	3590	6890
15	510	6540	3380	14140
20	720	6220	3610	6110
25	970	7050	3190	4870
30	1110	7980	3730	5980
35	980	6579	3330	5380
40	1040	10580	3610	5850

Review: A fluctuation has been observed in this graph for the list where the size is 15.



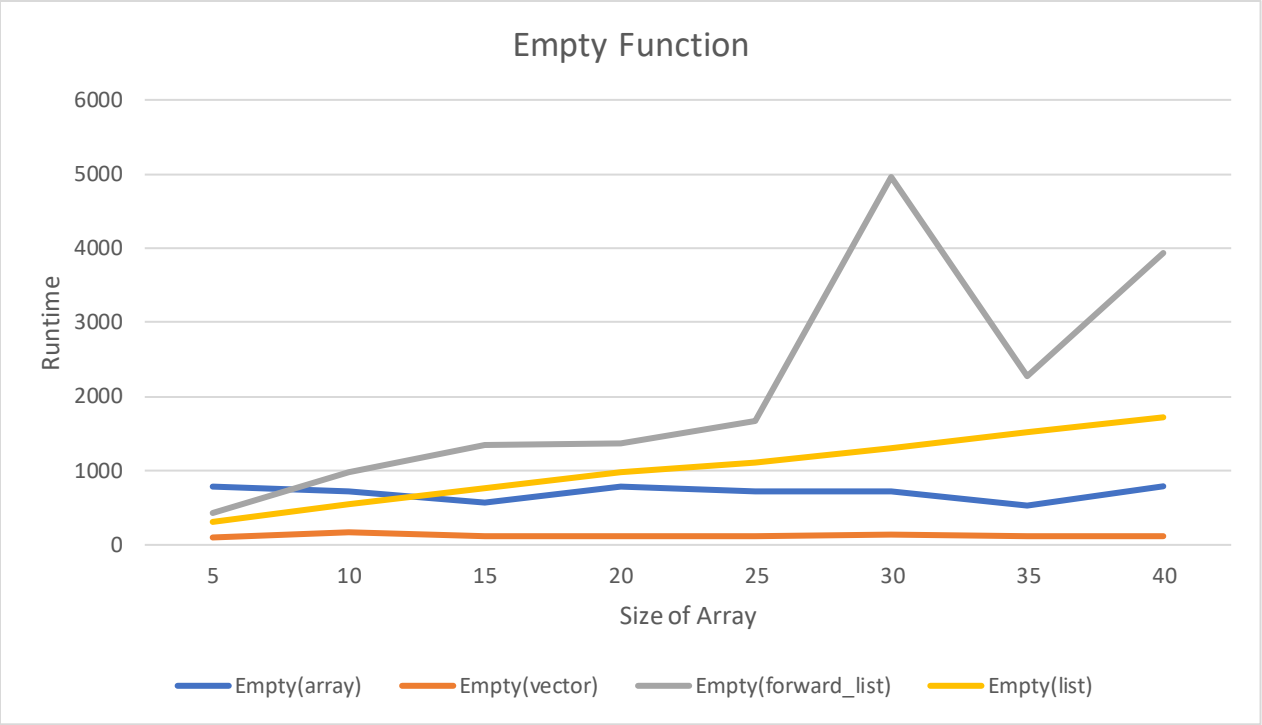
Size of Container	Remove_end(Array)	Remove_end(Vector)	Remove_end(ForwardList)	Remove_end(List)
5	70	110	220	5430
10	90	150	230	300
15	50	90	230	230
20	140	130	230	240
25	40	80	230	790
30	180	170	220	220
35	90	100	230	210
40	50	100	220	220

Review: From the graph, it is clearly observable that there has been a runtime error on the list when the size of list is 5, while the rest of the containers are stable.



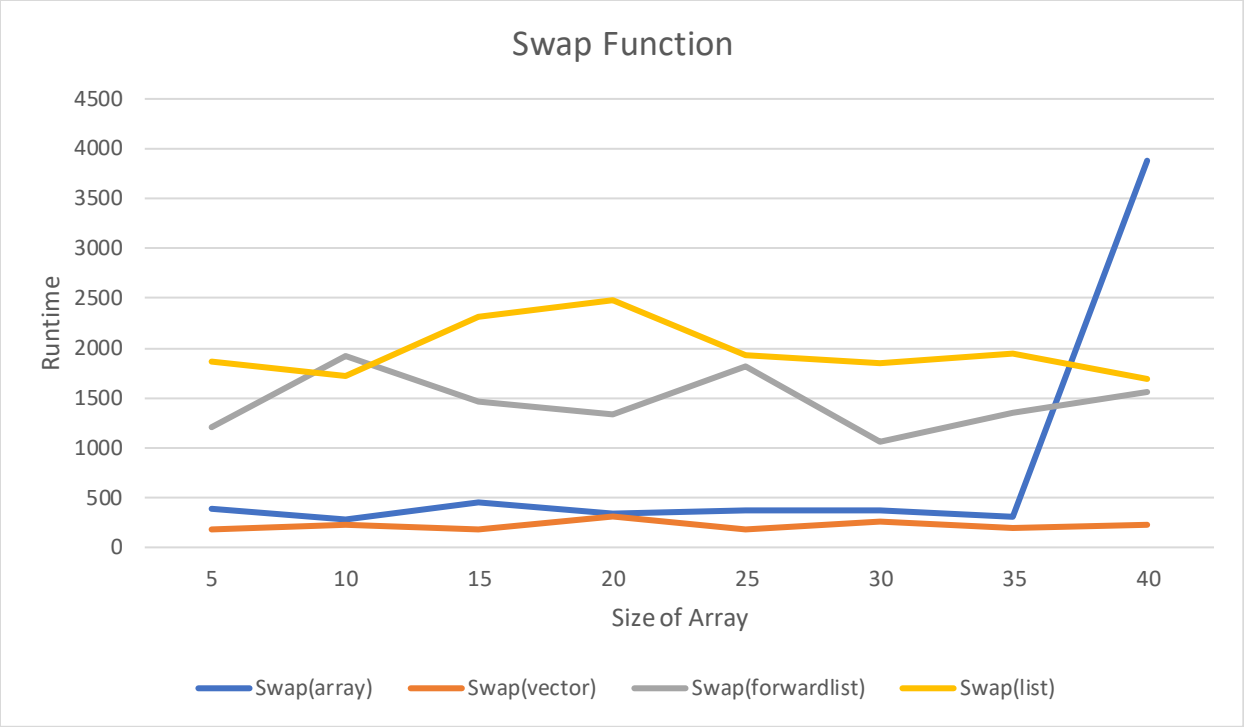
Size of Container	Remove_beginning (array)	Remove_beginning (vector)	Remove_beginning (ForwardList)	(List)
5	170	410	160	
10	340	520	150	
15	310	370	220	
20	380	550	140	
25	420	560	160	
30	620	820	140	
35	640	510	160	
40	630	999	170	

Review: This graph looks very promising with interesting observation where array has taken more runtime than usual. This shows how vector is more expensive than others, it takes more run time as all elements have to be moved to the end for spacing concerns, while for lists and forward lists, we can directly change the head node.



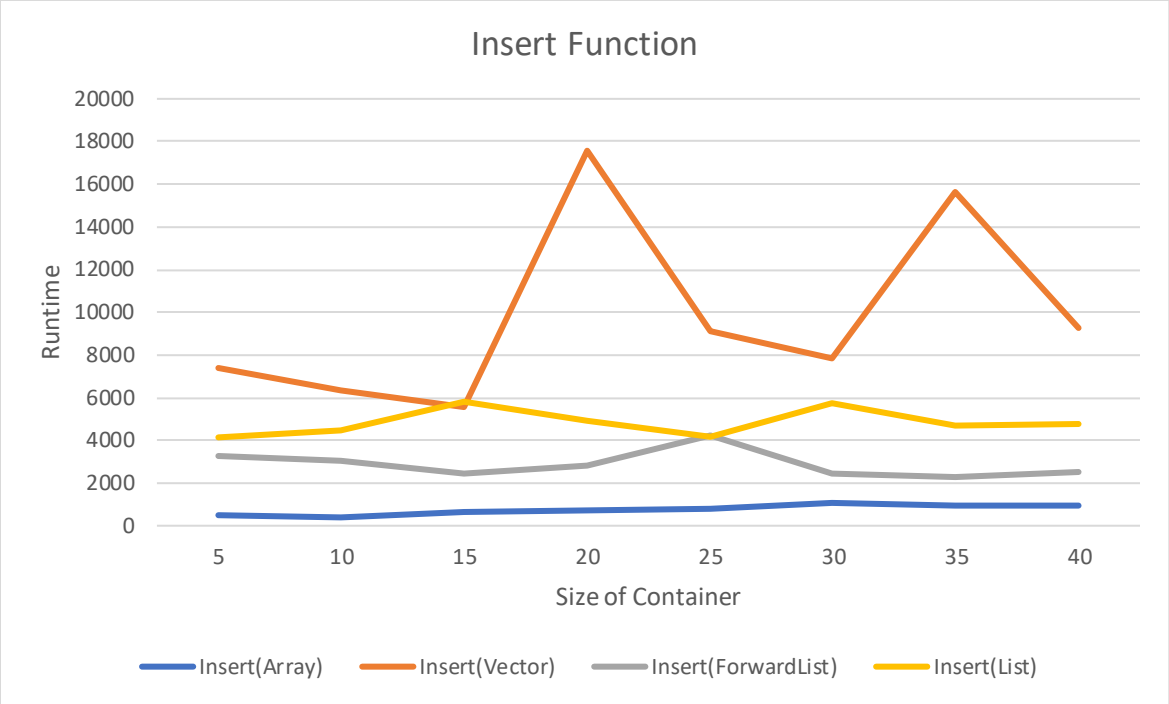
Size of Container	Empty(array)	Empty(vector)	Empty(forward_list)	Empty(list)
5	780	100	430	310
10	730	170	980	540
15	580	110	1350	760
20	780	110	1370	970
25	730	110	1660	1110
30	730	140	4960	1310
35	530	120	2270	1510
40	790	110	3930	1720

Review: The graph is mostly consistent, showing that emptying a vector does not make much run time error as most of the containers only used one function code for doing so.



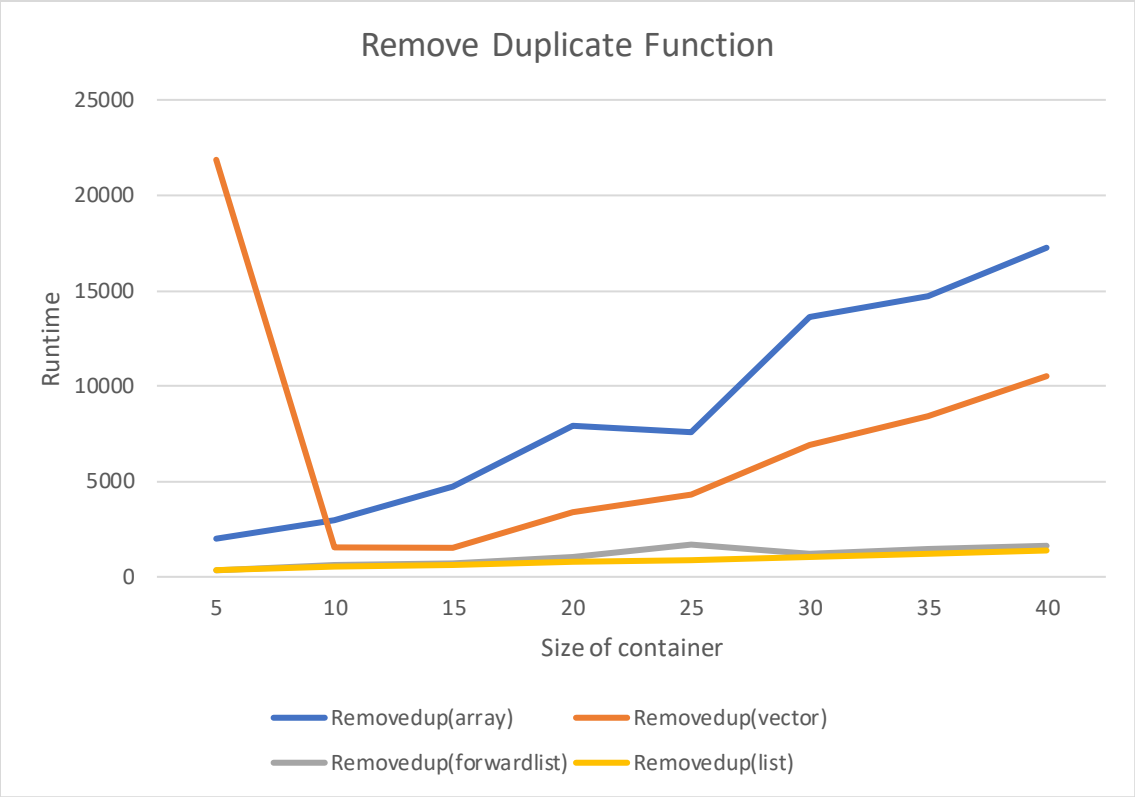
Size of Container	Swap(array)	Swap(vector)	Swap(forwardlist)	Swap(list)
5	380	180	1210	1860
10	280	230	1920	1720
15	450	180	1460	2320
20	340	310	1340	2480
25	370	180	1820	1930
30	370	260	1060	1850
35	310	190	1350	1950
40	3880	220	1560	1690

Review: The graph is stable showing the easiness of swapping, except for the last part for array when the size is 40, a runtime error has been observed. Array and vector have built in functions for swap which makes them have less runtime compared to lists.



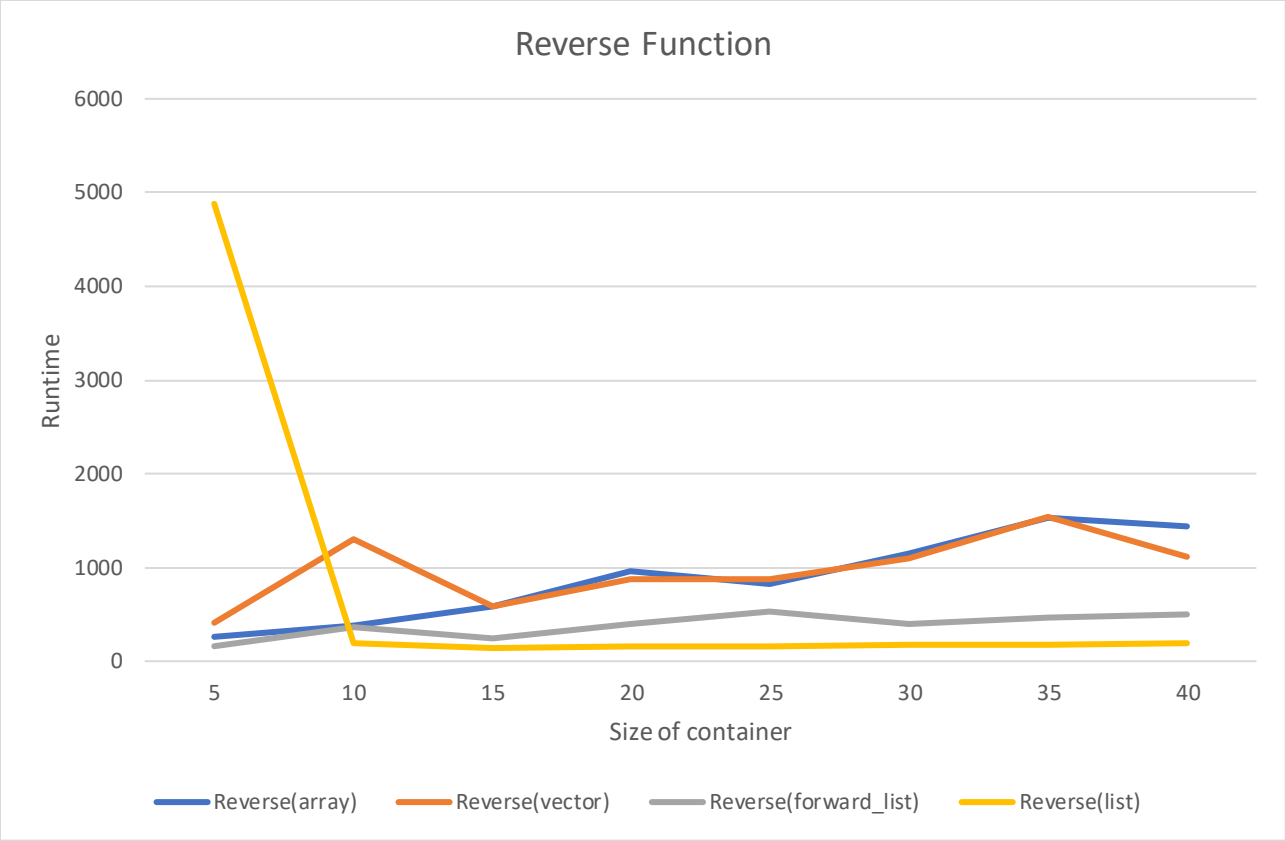
Size of Container	Insert(Array)	Insert(Vector)	Insert(ForwardList)	Insert(List)
5	490	7370	3230	4140
10	390	6370	3010	4440
15	650	5560	2480	5810
20	740	17570	2820	4920
25	800	9150	4230	4181
30	1070	7870	2480	5760
35	980	15670	2280	4670
40	970	9270	2490	4760

Review: This graph has shown a lot of fluctuation in the vector container, the rest of the containers have shown stable increase or decrease. Array takes less and constant run time compared to others because it is linear, while the other containers have dynamic features.



Size of Container	Removedup(array)	Removedup(vector)	Removedup(forwardlist)	Removedup(list)
5	2000	21860	340	350
10	2940	1530	590	500
15	4770	1520	690	620
20	7930	3420	1040	750
25	7620	4350	1690	900
30	13650	6950	1240	1060
35	14740	8390	1430	1200
40	17250	10520	1630	1380

Review: From the graph, the run time error for vector can clearly be seen when the size of vector is 5. The rest of the containers have a good stable growth in runtime. Array takes more time because it does not have a built-in function to check for duplicates and has to use for-loops to remove duplicates, this makes the run time increase. The case is different in other containers because they have built in functions such as unique, which takes less run time.



Size of Container	Reverse(array)	Reverse(vector)	Reverse(forward_list)	Reverse(list)
5	260	410	160	4880
10	380	1310	370	190
15	580	590	240	140
20	960	870	390	160
25	830	870	530	160
30	1150	1090	390	170
35	1530	1540	470	170
40	1430	1120	500	200

Discussion: From the graph, it can be observed that there is a run time error in the list when the size of array is 5. The rest of the containers have no problems with a stable growth in runtime as shown in the graph above. Array does not have a built-in function to reverse the elements, so it takes more run time. List and vector have built in reverse function which can directly return the output.