Report

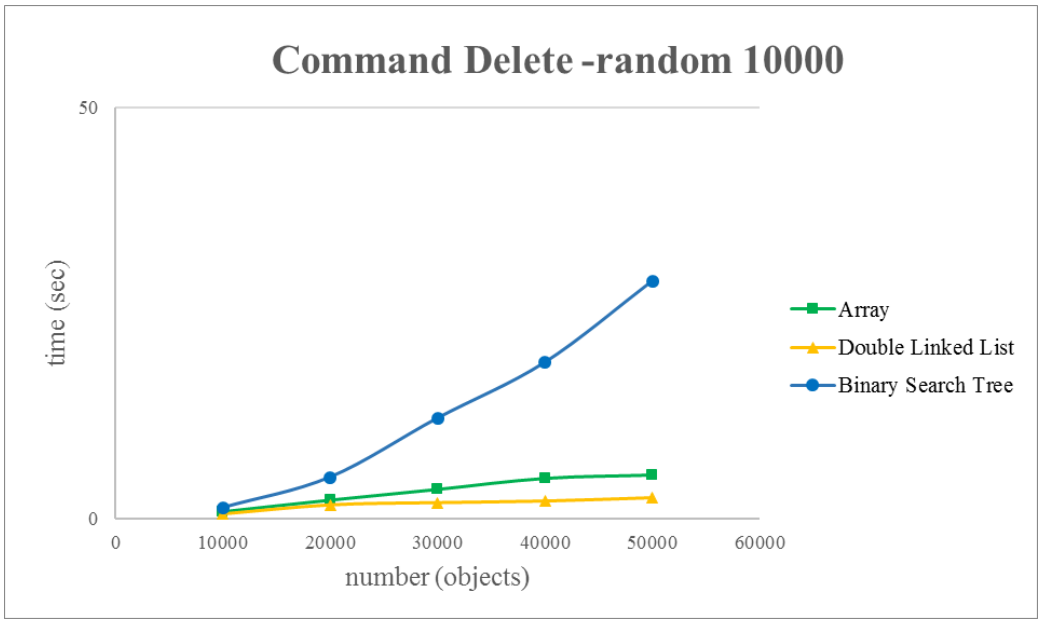
gray: worst when compared with others ; **yellow: best when compared with others;**

Command	Array			Double linked list			Binary search tree		
	Period Time (sec)	Total Time (Sec)	Memory (MB)	Period Time (sec)	Total Time (Sec)	Memory (MB)	Period Time (sec)	Total Time (Sec)	Memory (MB)
Case 1 (common use)									
adta –r 5000	0	0	0.9883	0	0	0.668	0	0	0.793
adtp	0	0	0.9883	0	0	0.688	0	0	0.793
adtp –r	0.01	0.01	0.9883	0	0	0.688	0.01	0.01	0.793
adts	0	0.01	0.9883	0.62	0.62	0.688	0	0.01	0.793
adtp	0.01	0.02	0.9883	0	0.62	0.688	0.01	0.02	0.793
adta –r 50000	0.03	0.05	3.73	0.01	0.63	3.676	0.08	0.10	3.746
adtd –r 10000	4.41	4.46	3.73	2.13	2.76	3.691	12.12	12.22	3.762
adts	0.04	4.50	3.73	35.39	38.15	3.691	0	12.22	3.762
adtd –r 10000	3.68	8.18	3.73	2.29	40.44	3.691	13.98	26.20	3.762
adts	0.01	8.19	3.73	15.61	56.05	3.691	0	26.20	3.762
adtd –b 10000	0	8.19	3.73	0.01	56.06	3.691	0.01	26.21	3.762
Case 2 (delete random)									
adta –r 50000	0.02	0.02	3.492	0.01	0.01	3.57	0.08	0.08	3.668
adtd –r 10000	5.31	5.33	3.492	2.46	2.47	3.641	28.83	28.91	3.867
adtd –r 10000	4.9	10.23	3.492	2.07	4.54	3.703	18.94	41.07	3.867
adtd –r 10000	3.54	13.77	3.492	1.88	6.42	3.703	12.16	60.01	3.867
adtd –r 10000	2.21	15.98	3.492	1.61	8.03	3.703	4.99	65	3.867
adtd –r 10000	0.77	16.75	3.492	0.53	8.56	3.703	1.28	66.28	3.867
Case 3 (sort)									
adta –r 50000 adts	0.04	0.09	3.598	49.94	49.95	3.5/ 3.625	0		

adta –r 40000 adts	0.04	0.06	3.426	30.23	30.24	2.852/ 3.203	0		
adta –r 30000 adts	0.02	0.03	1.895	20.09	20.09	2.336/ 2.598	0		
adta –r 20000 adts	0.02	0.03	1.895	7.48	7.48	1.508/ 1.676	0		
adta –r 10000 adts	0.01	0.01	1.203	1.86	1.86	0.9297/ 1.188	0		
adta –r 5000 adts	0	0	0.9414	0.56	0.56	0.6641/ 0.6641	0		

Observation.

1. When we add objects randomly, if the total number of objects is small, the difference of execution time among the three data structure is very small to detect by human, but the memory used of array is larger than the other two. With the number of objects increases, we can observe the execution time and the required memory of double linked list will be the most efficient, while binary search tree requires most time and memory space.
2. The command of deleting front, deleting back, printing forward and backward won't take much time.
3. With the number of objects increases, the execution time of deleting randomly increases and we can find that the double linked list takes least time while the binary search tree takes most time.
   The graph of deleting time v.s. number of object:



4. With the number of objects increases, the execution time of sorting objects increases and we can find that the double linked list takes more time when compared to array.
   (We don't take binary search tree into consideration because the structure is designed to be sorted already.)

The graph of sorting time v.s. number of object: