n *:* number of items in the data structures	**Search**	**Insert**	**Delete**	**Get to the *i* th element**	**Find minimum**	**Find maximum**
Unsorted array	O(*n*)	O(*n*)	O(*n*)	O(1)	O(*n*)	O(*n*)
Sorted array	O(log *n*)	O(*n*)	O(*n*)	O(1)	O(1)	O(1)
Unsorted linked list	O(*n*)	O(1) if done in the way done in the sample project	O(*n*)	O(*n*)	O(*n*)	O(*n*)
Sorted linked list	O(*n*)	O(n)	O(*n*)	O(*n*)	O(1)	O(*n*)
Binary search tree Worst-case: when the tree degenerates into a linked list ➔ The height of the search tree is O(*n*). On average ➔ The height of the search tree is O(log *n*)	Worst-case O(*n*) On average O(log *n*)	Worst-case O(*n*) On average O(log *n*)	Worst-case O(*n*) On average O(log *n*)	O(*n*)	Worst-case O(*n*) On average O(log *n*)	Worst-case O(*n*) On average O(log *n*)
Self-balancing variants of search trees such as red black trees, B trees	O(log *n*)	O(log *n*)	O(log *n*)	N/A	O(log *n*)	O(log *n*)
Heap	O(*n*)	O(1)	O(1)	N/A	O(1)	O(*n*)
Hash table	O(1)	O(1)	O(1)	N/A	O(*n*)	O(*n*)