Dictionary <TKey, TValue>

- Dictionary<TKey, TValue> is a generic collection that stores key-value pairs in no particular order.
- Keys must be unique and cannot be null, but Values can be null or duplicate.
- Elements are stored as KeyValuePair<TKey, TValue> objects.
- The capacity of a Dictionary is the number of elements that Dictionary can hold.
- Time Complexity(searching, adding, delete) => O(1)

HashSet <T>

- HashSet is an unordered collection of unique elements.
- It is generally used when we want to prevent duplicate elements from being placed in the collection.
- The performance of the HashSet is much better in comparison to the list.
- Time Complexity: O(1)

LinkedList <T>

- LinkedList<T> class uses the concept of linked list.
- It allows us to insert and delete elements fastly.
- It can have duplicate elements.
- insertion and removal are O(1) operations.
- Used for performing arithmetic operations on long integers.
- Linked List can be used in cases when faster insertion and deletion are required.

List<T>

- List<T> is a collection of strongly typed objects that can be accessed by index and having methods for sorting, searching, and modifying list.
- resize dynamically when an element is added or removed.
- It provides compile-time type checking and doesn't perform boxing-unboxing because it is generic.
- Time Complexity
    - Adding and removingAt => O(1)
    - Searching O(n)

PriorityQueue<TElement,TPriority>

- Represents a collection of items that have a value and a priority.
- On dequeue, the item with the lowest priority value is removed.
- If two elements have the same priority, they are served according to their order in the queue.
- Time Complexity:
    - Enqueue operation has O(1)
    - Dequeue has O(n).

Queue<T>

- Queue is a special type of collection that stores the elements in FIFO style (First In First Out), exactly opposite of the Stack<T> collection.
- It provides compile-time type checking.
- Time Complexity:
    - Adding and deleting => O(log(n))
    - searching : O(n)

Stack<T>

- is a collection class that works on the principle of Last In First Out (LIFO).
- Stack is useful to store temporary data in LIFO style, and you might want to delete an element after retrieving its value.
- It provides compile-time type checking and doesn't perform boxing-unboxing because it is generic.
- In Stack, you are allowed to store duplicate elements.
- Time Complexity: Adding =>O(1)

SortedDictionary<TKey, TValue>

- SortedDictionary<TKey, TValue> class uses the concept of hashtable.
- It stores values on the basis of key.
- It contains unique keys and maintains ascending order on the basis of key.
- It is dynamic in nature means the size of the sorted dictionary is growing according to the need.
- Time Complexity:
    - Adding and deleting => O(log(n))
    - searching : O(n)

SortedList<Tkey, Tvalue>

- SortedList is a collection of key/value pairs which are sorted according to keys.
- By default, In SortedList, an element can be accessed by its key or by its index.
- this collection sort the key/value pairs in ascending order.
- Same as List but sorted using a provided compare function and it uses binary search.
- Time Complexity:
    - Adding and deleting => O(n)
    - searching : O(1)

SortedSet<T>

- provides many mathematical set operations, such as intersection, union, and difference.
- dynamic collection means the size of the SortedSet is automatically increased when the new elements are added.
- In SortedSet, the elements must be unique.
- In SortedSet, the order of the element is ascending.
- In SortedSet, the user can only store the same type of elements.
- Time Complexity O(n).