- **List<T>:**
  - It allows duplicate values
  - Its not sorted
  - Insertion O(1) if count < capacity , O(n) if count > capacity ,searching with index O(1),RemoveAt certain index o(n) because you renumber the array again where n=count-index
  - You use It when you want dynamic array that is no sorted by default
- **LinkedList<T>**:
  - Allow duplicate values
  - Its not sorted
  - Searching O(n),insertion and removal O(1)
  - Use it if you want to insert in the middle of the list a lot and you don't need direct acces to elements
- **Queue<T>**:
  - It allows duplicate values
  - Its ordered First in First out
  - Insertion O(1) if you still in default capacity O(n) if you exceeded the default capacity, deleting , searching O(1) it gives you the first item in the queue
  - Same as list<T> so you use it if you want to access the element in the same order they stored
- **PriorityQueue<TElement,TPriority>:**
  - It allow duplicate values but its doesn't gurantee good behaviour in case of same priority
  - Its ordered accoding to the priority highest priority first
  - Insertion and deletion O(log n) ,dequeue O(1) it gives you the first item in the queue according to its priority
  - You use it when you want to run tasks or anything according to its priority
- **Stack<T>:**
  - it allows duplicate values
  - its ordered LIFO
  - Insertion O(1) if you still in default capacity O(n) if you exceeded the default capacity, deleting , searching O(1) it gives you the last item in the stack
  - Same as list<T> so you use it if you want to access the element in the reverse order they stored

- **Dictionary<Tkey,TValue>:**
  - It doesn’t allow duplicate keys
  - Its not ordered
  - Searching with key O(1),insertion O(1) and if the hashtable must be enlarged O(n) , Remove O(1)
  - You use it when you want to search for elements with some key and you don’t care that they sorted or not
- **SortedDictionary<Tkey,TValue>:**
  - It doesn't allow duplicate keys
  - Its sorted based on the key
  - Searching based on key O(log n) , add O(log n) ,remove O(log n)
  - You use it when you want to search for elements with some key and you care that they are sorted
- **SortedList<Tkey,TValue>:**
  - It doesn’t allow duplicate keys
  - Its also sorted based on the key
  - Searching O(log n) , add , remove O(n)
  - Similar to sorted dictionary except tree is implemented in an array so has faster lookup on preloaded data but slower loads
- **HashSet<T>:**
  - It doesn’t allow duplicate values
  - Its unordered
  - Add , delete search O(1)
  - You use it when you want unique unordered set its like dictionary but key and value are the same
- **SortedSet<T>:**
  - It doesn’t allow duplicate values
  - Its sorted
  - Add, search , delete O(log n)
  - You use it when you want unique sorted set its like sorted dictionary but key and value are the same