|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Structure** | **Summary** | **Main Operations (textbook’s definition)** | **Some Java Classes** | **Typical Implementations/ Interesting Properties** |
| **Bag (1-3)** | Collection that allows duplicate; provide convenient access to frequency counts. Unordered. | add, remove, contains, isEmpty, clear, getFrequencyOf, getSize | N/A | Resizable Array  Linked |
| **Set (n/a)** | Collection that does not allow duplicates.  Unordered. | add, remove, contains, isEmpty, clear, getSize | TreeSet, HashSet, *Set* | Resizable Array  Linked |
| **List (13-14)** | Collection that allows elements to be accessed via index. Ordered. | add, remove, get, clear, getSize | ArrayList, Vector *AbstractList* | Resizable Array - adds/removals to interior of list require data to be shifted.  Linked - Slow lookup |
| **Sorted List (16)** | Sorted collection that allows elements to be accessed via index. Ordered. | add, remove, get, clear, getSize | Might be able to use TreeSet or PriorityQueue | Resizable Array - gets can use binary search, adds require data to be shifted right.  Linked - lookup can be slow, if item is at end of list |
| **Queue (10-11)** | FIFO - First in first out collection. Adds to rear of queue and removes from front. | enqueue, dequeue, getFront, isEmpty, clear | LinkedList ,  *Queue* | Resizable Array  Linked  Two-part circular linked |
| **Priority Queue (10-11)** | Queue sorted by priority, not arrival order. | add, remove, peek, isEmpty, clear | PriorityQueue,  *AbstractQueue* | Not done in CSE 274 Heap is a common approach. |
| **Deque (10-11)** | Queue that allows access to front and rear. | addFront, removeFront,  addBack, removeBack, isEmpty, clear, | LinkedList,  ArrayDeque,  *Deque* | Doubly linked - keep track of front and rear. Bidirectional links links provide convenient ability to add/remove from either end. |
| **Stack (5-6)** | LIFO - Last in first out collection. | push, pop, peek, clear, isEmpty | Stack | Resizable Array  Linked  Vector |
| **Dictionary (19-20)** | Collection of Key-Value pairs. Keys must be unique. Unordered | add, remove, getValue, contains, isEmpty, getSize, clear | HashMap, TreeMap, *Map* | Resizable Array  Sorted Array  Sorted Linked |
| **Hash Table (21-22)** | Collection that allows fast insertions and removals. Can be used to represent sets, dictionaries, etc. | add, remove, getSize | Hashtable (represents a dictionary) | Resizable Array with linear probing  Resizable Array with separate chaining |
| **Trees and Binary Trees (23-24)** | Represents hierarchical information. Each node can have 0 or more children.  Ordering is usually important. | Operations to allow construction of tree and traversals | N/A | Linked |
| **BST (25)** | Stores data in an order based on “sortedness”. Can allow very fast search. Unordered. | add, remove, getSize, contains, isEmpty, clear | N/A | Linked - performance of average random BST is O(log N) |
| **Heap (26)** | Stores data in an order based that allows quick removal of max element. Can be a basis of priority queue implementations. Unordered. | add, removeMax, getMax, isEmpty, getSize, clear | N/A | Resizable Array - adds and removes are O(log N) |
| **AVL Tree (27)** | Enhancement of BST. Ensures tree is balanced. Very fast search. Unordered. | See BST operations | N/A | Linked - O(log N) search add, contains, and remove |
| **Graphs (28-29)** | Represents a network of entities. Relationships between entities matters. Ordered. | Operations to allow construction of graph and lookup of adjacencies | N/A | Adjacency Matrix - useful for graphs with many connections.  Adjacency List - useful for graphs with few connections. |