1. Primitive Type vs. Composite Type
2. ADT and DS

|  |  |
| --- | --- |
| **Abstract Data Type** | **Implementation (DS or ADT)** |
| Container (Single Value, Associative) |  |
| List (Sequence) | Array, Linked List |
| Set and Multiset (Bag) | List (Inefficient), Tree, Trie, Hash Table |
| Stack | Array (Efficient Dynamic), Singly Linked List |
| Queue | Circular Array or Buffer, Linked List (Doubly, Circularly, Singly w. Pointer) |
| Double-ended Queue | Dynamic Array, Doubly Linked List |
| Priority Queue | Sorted or Unsorted List, **Heap** |
| Associative Array (Map, Dictionary) | **Hash Table**, Search Tree, (Uncommon) Associative List. |
| Multimap | Map with lists or sets. |
| Tree | Linked Tree, (Binary) Array, List of Lists. |
| Graph | Adjacency list, Adjacency matrix, Incidence matrix |

Container ADT or DS:  whose instances are collections of other objects. Access, Storage, Traversal.

List: a countable number of ordered values.

Tuple: finite ordered list (sequence) of elements.

Set: Unique Value, Unordered. Static (Frozen) Set vs. Dynamic or Mutable Set. Set Operations.

Multiset: Set allowing duplicates. Equivalent relation.

Stack: Collection of elements. Push. Pop LIFO.

Queue: Collection of elements. Enqueue. Dequeue FIFO

Deque: Generalized queue. Enqueue and dequeue on either end.

Map: Collection of (key, value). Key appears at most once.

Multimap: Generalize map. More than one value for a given key.

Tree ADT: A hierarchical tree structure, with a root value and subtrees of children with a parent node.

Graph: Set of vertices with set of pairs of these vertices (Edge). Ordered vs. Unordered.

Array: collection of elements identified by indices or keys.

Linked List: A collection of nodes representing a sequence, with a reference (link) to the next node.

Heap: A tree-based DS that almost complete and satisfies heap property.

Hash Table: DS maps keys to values using hash function to compute index into busket.

Search Tree: A tree DS, Left < Node < Right.

Tree DS: A linked tree is a group of [nodes](https://en.wikipedia.org/wiki/Node_(computer_science)), where each node has a value and a list of [references](https://en.wikipedia.org/wiki/Reference_(computer_science)) to other nodes (its children). Nodes in a tree could have next/previous references or references to their parent nodes. **Traversal methods**.

Tree ADT vs Tree DS <-> Tree vs. Linked Tree <-> List vs Linked List

|  |  |
| --- | --- |
| **Array-Based Sequence** | **Linked-Based Sequence** |
| O(1) access to element based on index |  |
| Generally runs faster than linked. |  |
| Use proportionally less memory. |  |
| Need to shift elements. | O(1) insertion and deletion at arbitrary position. (Not including searching time) |
|  | Bouned by non amortized operation. |

1. Non-general ADT and DS from [3]

Positional List ADT and Doubly Linked List, 7.4.

Adaptable Priority Queue and Locators, 9.5.

Sorted Map and Skip List, 10.3 & 10.4.

[1] <https://en.wikipedia.org/wiki/List_of_data_structures>

[2] <https://en.wikipedia.org/wiki/Abstract_data_type>

[3] Data Structures and Algorithms in Python, Goodrich, Tamassia & Goldwasser