**LINK LIST:**

* A linked list is a linear data structure where each element is a separate object. Each element (we will call it a node) of a list is comprising of two items – the data and a reference to the next node.
* The last node has a reference to null. The entry point into a linked list is called the head of the list.

**ADVANTAGES:**

* **.Dynamic Data Structure**
* **. Insertion and Deletion**
* **. No Memory Wastage**
* **. Implementation**

**DISADVANTAGE:**

* **Traversal**
* **Reverse Traversing**

**QUEUE:**

* Queue is an abstract data structure, it is similar to Stacks. Unlike stacks, a queue is open at both its ends.
* One end is always used to insert data (enqueue) and the other is used to remove data (dequeue).
* Queue follows First-In-First-Out methodology, it means the data item stored first will be accessed first.

**ADVANTAGES:**

* Speed
* Flexibility
* Multiple jobs

**DISADVANTAGES:**

* New element can only be inserted when *all* of the elements are deleted from the queue

**STACK:**

* Stack is a LIFO(Last in First out) structure or we can say FILO(First in Last out). push() function is used to insert new elements into the Stack and pop() function is used to remove an element from the stack.
* A stack is a limited access data structure - elements can be added and removed from the stack only at the top.

**ADVANTAGES:**

* Stack allows the recursion.  
  Stack have conserves storage.

**DISADVANTAGES:**

* **i**n stack Overhead of allocation and also deallocation. Subprograms cannot be history sensitive  
  stack have an Inefficient references (indirect addressing)

**HEAP:**

* A heap is an area of pre reserved computer storages in which a program can use to store data variables.
* It won’t be known until the program is running .and also we can say that The size of heap cannot be determined at compile time.

**ADVANTAGES:**

* heap provides for dynamic storage management.

**DISADVANTAGES:**

* It is an inefficient and unreliable.

**HASH TABLE:**

* Hashing is the process of mapping large amount of data item to a smaller table with the help of a hashing function.
* The essence of hashing is to facilitate the next level searching method when compared with the linear or binary search.

**ADVANTAGES:**

* No duplications are allowed because one hash value will contain only one key value.
* Main advantage is synchronization.
* In many situations, hash tables turn out to be more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer softwares, particularly for associative arrays, database indexing, caches and sets.

**DISADVANTAGES:**

* Hash collisions are practically unavoidable. when hashing a random subset of a large set of possible keys.
* Hash tables become quite inefficient when there are many collisions.
* Hash table does not allow null values, like hash map.

**PRIORITY QUEUE:**

* Every item has a priority associated with it
* An element with high priority is dequeued before an element with low priority.  
   If two elements have the same priority, they are served according to their order in the queue.

**OPERATIONS:**

Is\_empty.

insert\_with\_priority:

pull\_highest\_priority\_element:

**TREE (BINARY SEARCH TREE & AVL TREE):**

* Tree contain a root value and subtrees of children with a parent node, represented as a set of linked nodes.
* The BST follow the left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key.
* The AVL is a self-balancing binary search tree. It was the first such data structure to be invented it is also Similar to red–black trees, AVL trees are height-balanced.

**ADVANTAGES OF BST:**

* We can also do range queries - find keys between N and M (N <= M).
* We can implement order statistics with binary search tree - Nth smallest, Nth largest element. This is because it is possible to look at the data structure as a sorted array.

**DISADVANTAGES OF BTS:**

* The main disadvantage is that we should always implement a balanced binary search tree - AVL tree, Red-Black tree, Splay tree. Otherwise the cost of operations may not be logarithmic and degenerate into a linear search on an array.

**ADVANTAGES OF AVL TREE:**

* Due to balance factor it never take a form of chain, in fact AVL tree is nearly complete Binary tree.
* Operations like Insertion, Deletion, Searching in a tree take O(log n) Time in worst case and Average case.

**DISADVANTAGES:**

* Slow Inserts and Deletes
* The code for AVL tree is much more complex than the code for BST and we have to handle a lot of corner cases.
* Deletion operations cost high in AVL trees as they involve a lot of pointer changes and rotations.

**GRAPHS(DIRECTED AND UNDIRECTED):**

* A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links.
* A directed graph is  a graph, contain a set of objects called vertices or nodes that are connected together, where all the edges are directed from one vertex to another vertex. A directed graph is sometimes called a digraph or a directed network.
* A graph whose edges are unordered pairs of vertices. That is, each edge connects two vertices.

**ADVANTAGES:**

* Representation is easier to implement and follow. Removing an edge takes O(1) time.

**DISADVANTAGES:**

* Consumes more space.

**APPLICATIONS:**

* Circuits
* Maps
* Schedules
* Computer networks

**DISJOINT SETS:**

* A disjoint-set data structure is a data structure that keeps track of a set of elements partitioned into a number of disjoint and non-overlapping subsets.

**ADVANTAGES:**

* The disjoint set data structure is used to keep track of the connected components of an undirected graph when the edges are added to the graph dynamically.

**DISADAVANTAGES:**

* Time complexity.

**APPLICATIONS:**

* Connected component labeling
* Online maintenance of biconnected components
* Alias analysis.