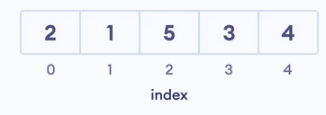
**Linear data structures:**

Elements are arranged in sequence one after the other.

1. A picture containing text, electronics

   Description automatically generated**Array:**

Store data sequentially in the memory.

Elements are of the same type.

1. **Stack:**

Diagram, icon

Description automatically generatedLIFO principle.

1. **Queue:**

FIFO principle.

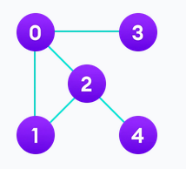
1. **Linked List:**

Elements are connected through a series of nodes.

Each node contains data items and address to the next node.

**Non-Linear data structures:**

Elements are not in any sequence. Arranged in a hierarchical manner.

1. **Graph:**

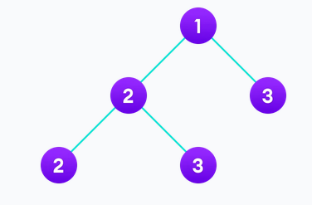
Nodes are called vertices.

Vertices are connected through Edges.

* Spanning Tree and Minimum Spanning Tree.
* Strongly Connected Components.
* Adjacency Matrix.
* Adjacency List.

1. **Trees:**

A tree is a collection of Vertices and Edges.

Only one edge between two vertices.

* Binary Tree
* Binary Search Tree
* AVL Tree
* B-Tree
* B+ Tree
* Red-Black Tree

**Solving Big O notations:**

1. **Iterative Method:**

- Check 3 iterations manually.

- Substitute the found iterations in the first iteration.

- Find the “" iteration.

- Using the former step, find the stopping condition for the recursion.

- Substitute the found condition and reach the answer.



**1.5-** **Variables Substitution:**



1. **Master Theorem:**

- a >= 1 & b > 1

- T(n) is defined by positive integers, as follows:

I) if,

then,

II) if,

then,

III) if, AND (regularity condition)

then,

**Array Data Structure:**

For one dimensional, a simple type varName[] (JAVA) \ var\_name = […] (Python).

**Stack Data Structure (LIFO):**

Operations of stack:

- push()

- pop()

- isEmpty()

- isFull()

- peek(), returns the value of the top element (without actual popping)



**Queue Data Structure (FIFO):**

Simple Queue - This DS has its limitations.

Until the queue is reset (emptied), the dequeued indexes can’t be re-used.

Operations of simple queue:

- enqueue()

- dequeue()

- isEmpty()

- isFull()

- peek(), returns the value of the front element (without removing it)



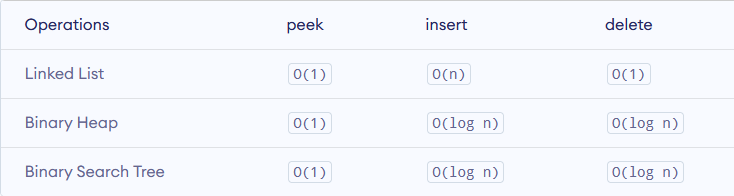
Circular Queue – It’s advantage over the simple queue is better memory utilization.

If the last position is full, and first is empty -> insert element in first.



Priority Queue –

Different methods of implementing:



RETURN TO – AFTER IMPLEMENTING EACH OF THE ABOVE DATASTRUCTURES

Deque (Double Ended Queue) –

Allows the user to insert\remove elements from both the front and the rear.



**Linked List Data Structure:**