**Algorithms**

# Sorting:

**01: Bubble Sort**

* Run two loops
* If **a[ j ] > a[ j+1 ]** then replace that

**02: Selection**

* Run Two Loops
* Store value of first loop each time
* Each time the second loop will be like that
  + // j=1->len // j=2->len // j=3->len
* If a[ j ] < a[ mainElement ] then mainElement = j
* When the second loop completed check that condition
  + If mainElement != i then swap them like arr[mainElement]⬄ arr[i]

**03: Insertion**

Picking One value and set it into its exact position

* Two Loops
* **Second loop is a reverse loop**

04: Merge

The Concept behind Merge Sort Algorithm is, take two sorted array and merge them and if we have only one unsorted array then divide them using recursion technique.

* Two Functions
* Recursion

05: Quick

The Concept behind the Algorithm is we select a value and place them in its correct position like all the elements in left hand side is less then the value and all the elements on right hand side are greater than the selected value.

06: Counting Sort

07: Radix Sort

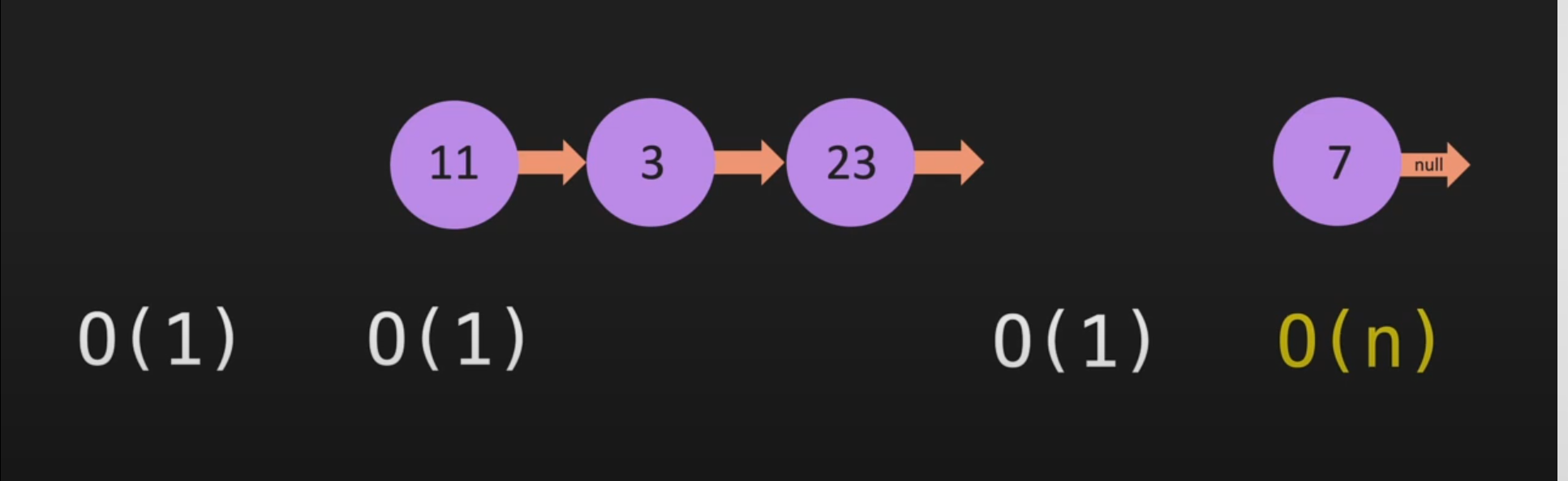
# Stack And Queues: [Link](https://www.youtube.com/watch?v=--W233f-hCI&list=PLdPTfo6Ung1DKV0IUPsoRCZUXokM04nXf&index=1)

01: Stack => LIFO

* Data Structures that we can use with Stack
  + Array
  + Linked List
* If you are implementing Stack with Array then use Pop() Push() “O(1)” and do not use shift and unshift because then the index will always change and the complexity will be order of O(n)

02: Queues => FIFO

Data Structures that we can use

* Array
* Linked list
  + 

# Searching:

01: Linear

02: Binary

# Recursion:

I think there are two types of recursions

1. Reverse
2. Forward
3. We can use multiple recursion inside a function

01: Factorial

02: Fibonacci

03: Memorization

04: Tail Recursion

# Graphs:

01: Breadth-First Search (BFS)

02: Depth-First Search (DFS)

03 Flood Fill

# Dynamic Programming:

01: Knapsack Problem

02: Longest Common Subsequence

# Linked list:

01: Single

02: Double

# Hash Table:

01: With collision

02 Without collision

# Tree Traversal:

01: BFS

02: DFS

03: Binary Search

# Two Pointers:

01: Simple two pointer

# Backtracking:

01: General purpose

02: Simple Example

# Binary Heaps:

# Union Find

# Ad hoc / String manipulation:

# Greedy Algorithm:

# Trie:

# Segment Tree:

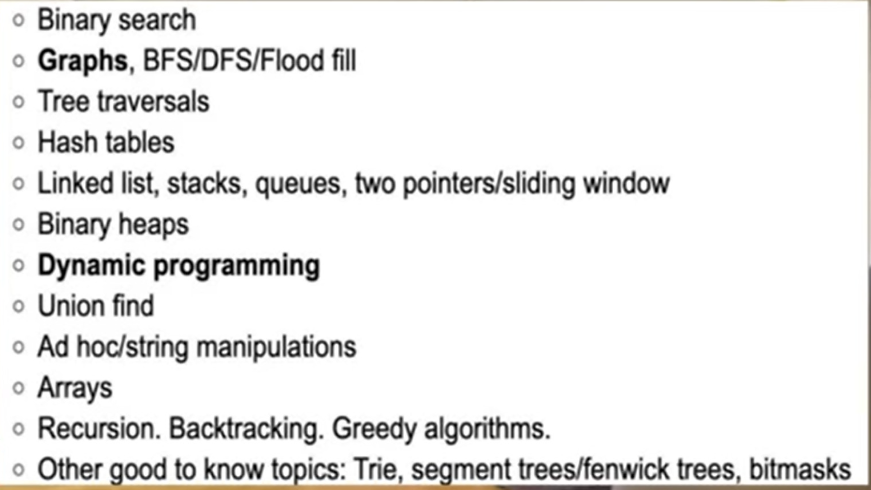
# Fenwick Trees:

# Bitmask:

# Time Complexity:

// --- Important Resources

All the Algorithms which we should know before apply to Google



All Algorithms are present there with Animation:

~ <https://www.w3schools.com/dsa/index.php>

GPT Chat link:

~ <https://chatgpt.com/share/01b0c8ed-2a3d-448b-8e38-7f91c28cc1aa>