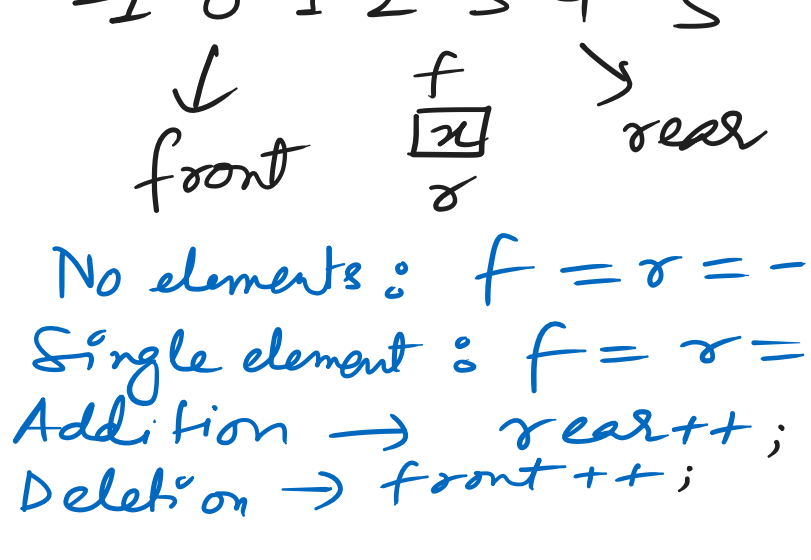
Queue Data Structure (FIFO)

Array Implementation

Removal — Addition



No elements : $f = r = -1$
Single element : $f = r = 0$
Addition → rear++;
Deletion → front++;

→ enqueue/dequeue

Applications

* BFS traversal
↓
Level Order traversal

Trees/Graphs

Complete DSA RoadMap for Engineers

* Basics of C, File Handling, Dynamic Memory Allocation.
*** Mandatory
* Any one of C++ or Java for DSA
* Arrays → 1D, 2D, MD, Jagged
* Searching & Sorting Algos
* linear
* binary
* recursive binary
* jump * interpolation

} Time Complexity

* Bubble / Selection / Insertion
* Merge / Quick
* Count / Radix
* Shell
* Heap
* Wave Sort

} Sorting

Linear Data Structures : →

Stacks, Queues → Problems

Linked Lists → Singly
→ Doubly
→ Circular

Collections Library (java.util)
List, Stack, Queue, Deque,
Map, HashMap, Set, HashSet,
HashTable <vv imp>, PriorityQueue
ArrayList, LinkedList

PriorityQueue → Heap → Max Heap / Min Heap

{ These are built-in data structures in Java }
* We use them in coding interviews to save time.

Hashing*, HashCode*, Hash Collision*

Non-linear Data Structures : →

* Trees
  - ↳ Simple Trees
  - ↳ Binary Trees
  - ↳ Tree Traversals (Pre, In, Post Orders) — DFS
  - ↳ BFS or Level Order Traversal
  - ↳ Binary Search Trees
  - ↳ Height of Binary Tree
  - * ↳ AVL Trees
  - * ↳ Red Black Trees } height balanced (rotations)
  - * ↳ Segment Trees
  - * ↳ Fenwick Trees / Binary Index Trees
  - ↳ Skewed Trees
  - ↳ KDimensional Trees
  - ↳ Orthogonal range Trees
  - (*) ↳ B & B+ Trees (DBMS)
  - ↳ Complete Binary Tree (Heap)

(GLPA)

Advanced Tree → Trie
LCP
↓
prefix → coding ninja
→ coder
→ code → Trie
beginning → coding
Phone Book / Dictionary O(n)

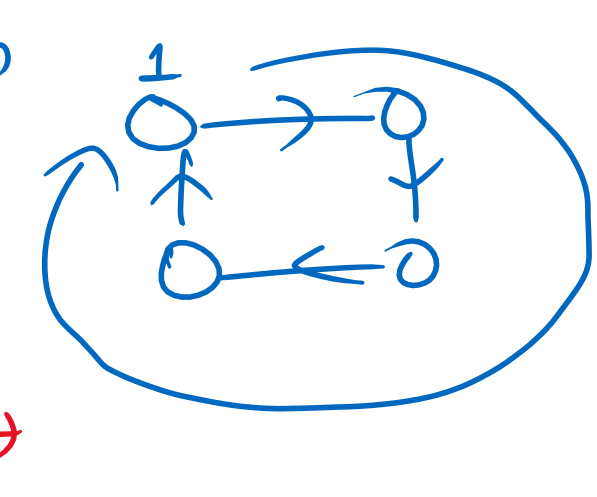
Heaps
↳ Priority Queue
Max Min

Queue → Interface
↑
Priority Queue
↳ class

Heap Sort Algo

Graphs : Theory & Types
Representation → Adj List
→ Adj Matrix
Traversals → DFS
→ BFS

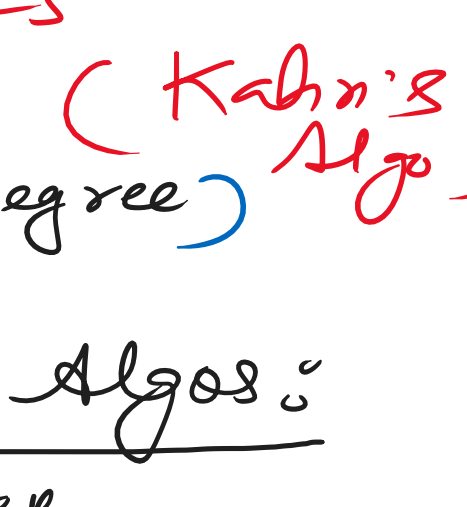
Cycle Detection
DFS BFS



Graph Algos : →
Topological Sort (DAG)
Linear Ordering of Nodes.
DFS BFS
Topological Sort (Kahn's Algo)
New Concept (In-Degree)

Shortest Distance Algos :

* Dijkstra's Algorithm
  It doesn't work for -ve weights/values.
  * Infinite loop.
* Bellman Ford
* Floyd Warshall's



$-2 + -2 = -4$

Single Source

MST → Minimum Spanning Tree
* Prim's Algo
* Krushkal's Algo
  * Disjoint Set → Union (Rank, Size) → find parent
* Kosaraju's Algo ***
* Strongly connected components

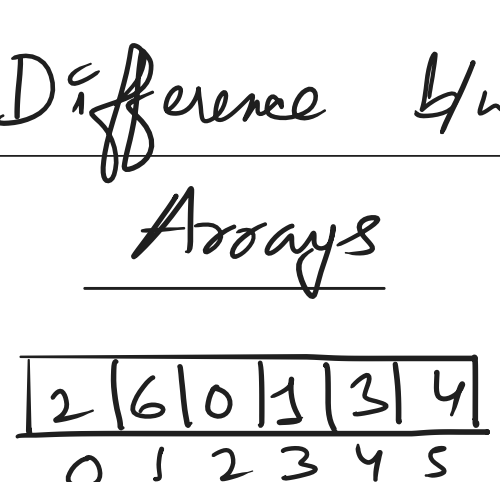
Greedy Algos	Bit Manipulations (Masking)

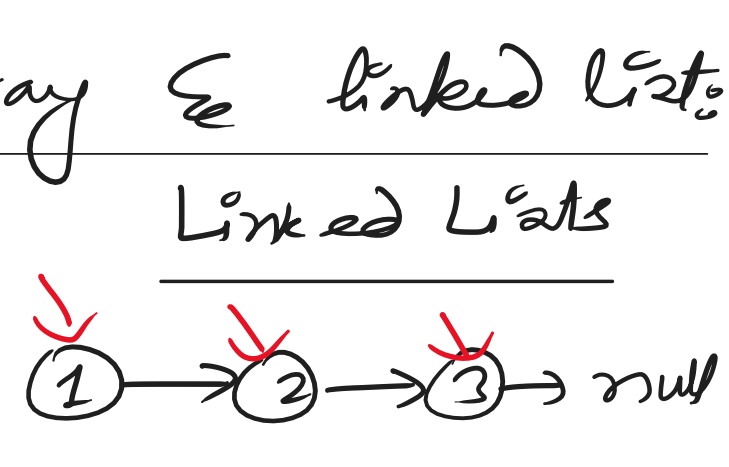
* Back Tracking → Recursion
  * N Queens
  * Rat In A Maze
  * Sudoku Solver
  * Subsets of a String/Array
  * Phone keypad Problem

* Dynamic Programming
  * Recursion
  * Memoization
  * Tabulation
  * Space Optimization **

* Difference b/w array & linked list:

Arrays	Linked Lists

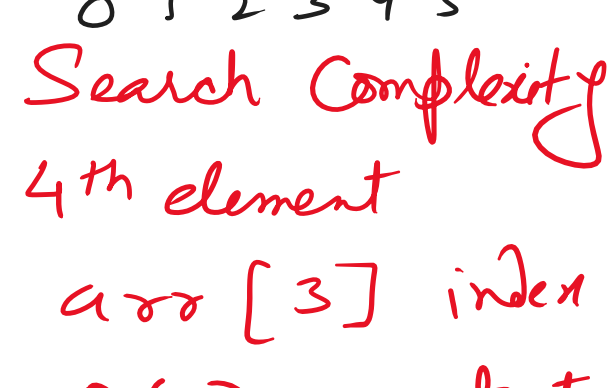




① Search Complexity
4th element
arr[3] index
O(1) constant

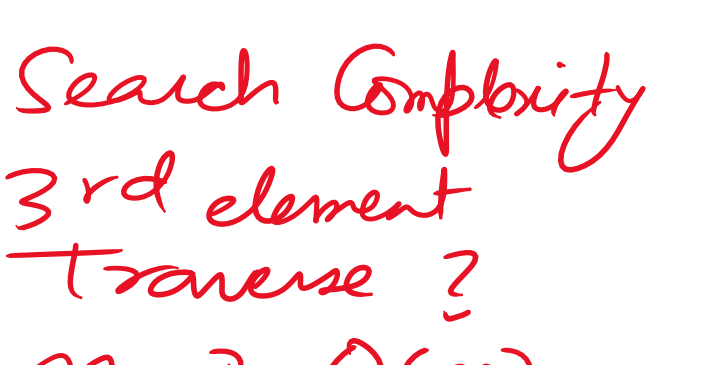
① Search Complexity
3rd element
Traverse ?
n → O(n)

② Insert Complexity



O(n)

Insert Complexity

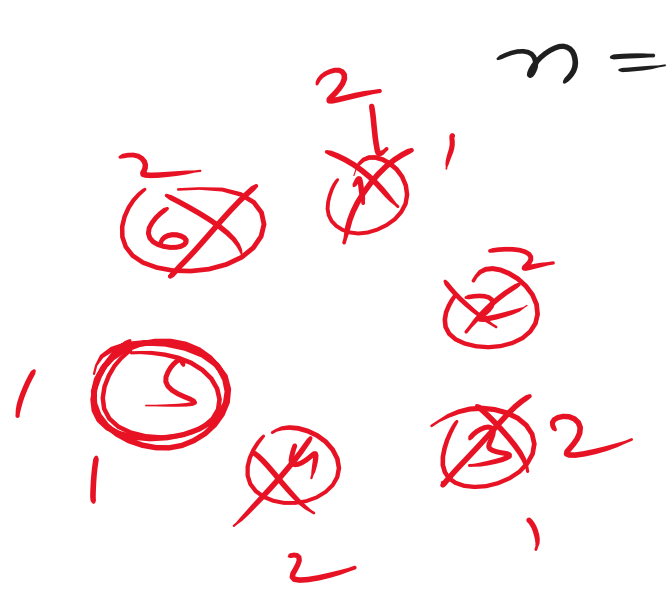


O(1) 3 steps

1823 : Winner of the Circular Game (Josephus Problem)

Sample Input 1 :
$n = 5, k = 2$, winner = 3

$n = 5, k = 2, w = 3$

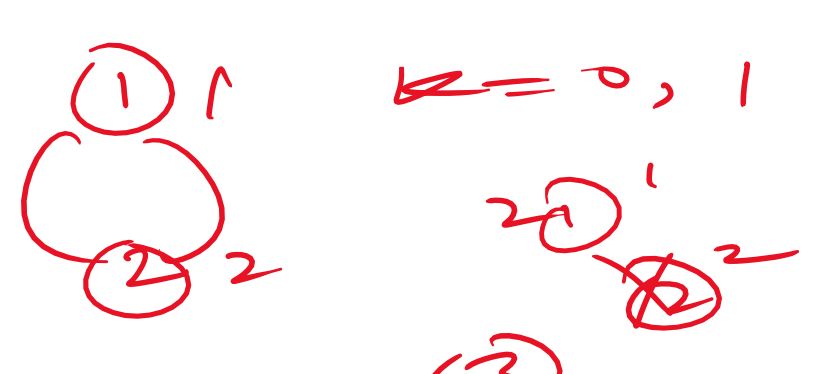


n	w
1	1
2	1
3	3
4	1
5	3
6	5

Fib Series 0, 1, 1, 2, 3, 5, 8, 13
Recursion? $f(n) = f(n-1) + f(n-2)$

solve(n,k) → solve(n-1,k)
solve(2,k) → solve(1,k)
1 1
solve(3,k) → solve(2,k)
3 1+k
1+2 = 3
solve(4,k) → solve(3,k)
1 3+2 = 5 % n → 5 % 4 = 1
solve(5,k) → solve(4,k)
3 1+2 = 3
solve(n,k) = [solve(n-1,k) + k] % n

n	w
1	1
2	1
3	3
4	1
5	3
6	5



k = 0, 1   n↑   k↑
1, 2

Normalization
↓
within Limit

0 % 5 = 0
1 % 5 = 1
2 % 5 = 2
⋮
5 % 5 = 0   6 % 5 = 1

Circular arrange