INTRODUCTION TO SORTING, SEARCHING, AND HASHING

SirJae

"ORGANIZE THE MESS!"

Imagine you are helping set up a library. You have piles of books and need to:

- Sort the books by title.
- Search for a specific book quickly.
- Organize the books on a shelf so you can find any book fast later.

- HOW WOULD YOU MANUALLY ARRANGE THE BOOKS?

- WHAT IF THE BOOKS KEEP ARRIVING IN RANDOM ORDER?

- HOW WOULD YOU SPEED UP FINDING A BOOK AMONG 500 OR 5,000 BOOKS?

SORTING ALGORITHMS

Sorting Algorithms are a class of algorithms that arrange the elements of a list or collection in a specific order — most commonly in ascending or descending order according to a defined comparison operation (such as numerical or lexicographical order).

Sorting is a fundamental operation in computer science because it improves the efficiency of other algorithms (like search algorithms) and organizes data in a meaningful way.

SORTING ALGORITHMS

Key Points:

- Sorting can be based on comparisons (e.g., Insertion Sort, Merge Sort) or non-comparisons (e.g., Counting Sort, Radix Sort).
- Sorting affects the performance of many algorithms that require ordered input (like Binary Search).
- The efficiency of a sorting algorithm is measured by its time complexity, space complexity, and sometimes its stability (whether equal elements maintain their relative order).

SORTING ALGORITHMS

Common Sorting Goals:

- Efficiency: Minimize time and memory usage.
- Stability: Maintain the order of equal elements.
- In-place: Sort without using extra significant memory.

EXAMPLES OF SORTING ALGORITHMS:

Algorithm	Type	Average Time Complexity
Insertion Sort	Comparison-based	$O(n^2)$
Selection Sort	Comparison-based	$O(n^2)$
Heap Sort	Comparison-based	$O(n \log n)$
Counting Sort	Non-comparison-based	$O(n + k)$
Radix Sort	Non-comparison-based	$O(d(n + k))$

INSERTION SORT

DEFINITION:

Insertion Sort is a simple comparison-based sorting algorithm that builds the final sorted array one item at a time. It repeatedly takes the next element and inserts it into the correct position among the previously sorted elements.

HOW IT WORKS:

You take one item at a time and insert it into its correct position in a sorted part of the list.

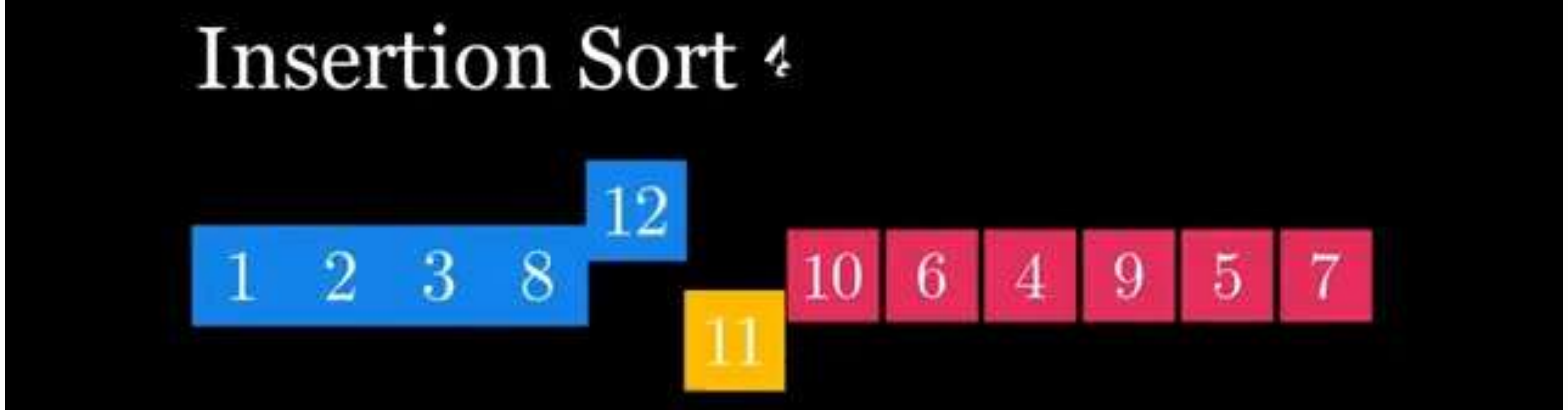
- Real-life Example: Organizing playing cards in your hand.

Time Complexity:

Best: $O(n)$
Average/Worst: $O(n^2)$

Space Complexity: $O(1)$ (in-place)

STEPS:



1. Start with one card.
2. Pick the next card and insert it into its correct position among already sorted cards.
3. Repeat until all cards are sorted.

SELECTION SORT

DEFINITION:

Selection Sort is a comparison-based algorithm that repeatedly selects the minimum (or maximum) element from the unsorted portion of the list and swaps it with the first unsorted element, shrinking the unsorted section by one each time.

HOW IT WORKS:

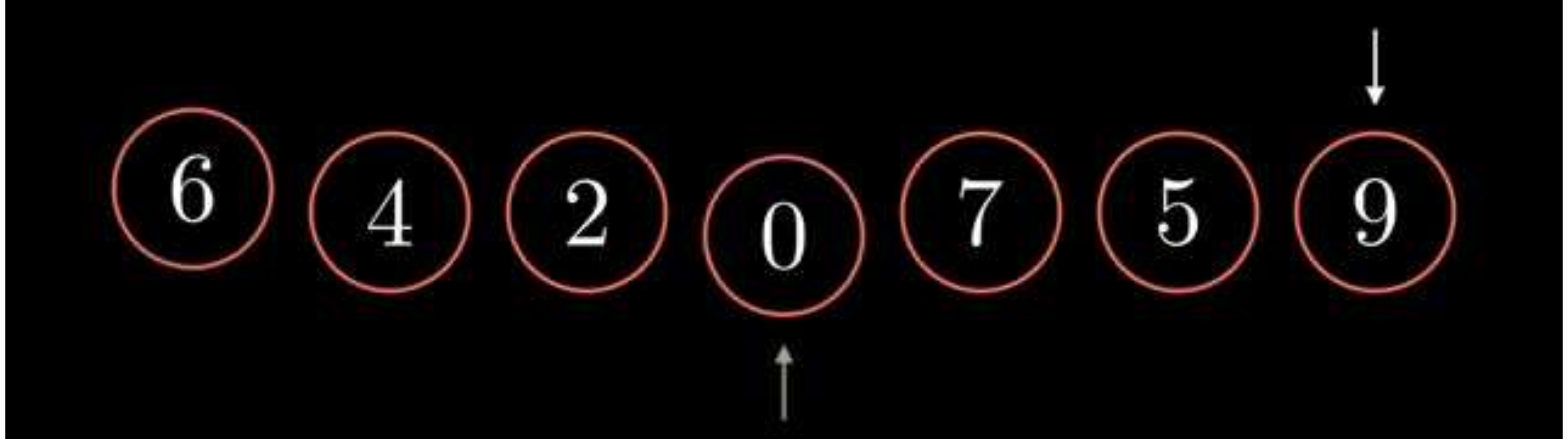
Find the smallest (or largest) item and place it at the beginning. Repeat for the remaining list.

- Real-life Example: Choosing the cheapest item from a shelf first, then the second cheapest, and so on.

Time Complexity: $O(n^2)$ (for all cases)
Space Complexity: $O(1)$ (in-place)

Space Complexity: $O(1)$ (in-place)

STEPS:



1. Look through the list to find the smallest item.
2. Swap it with the first item.
3. Repeat for the rest of the list.

HEAP SORT

DEFINITION:

Heap Sort is a comparison-based sorting technique based on a binary heap data structure. It first builds a Max-Heap or Min-Heap from the input data, then repeatedly extracts the root (largest or smallest element) and rebuilds the heap.

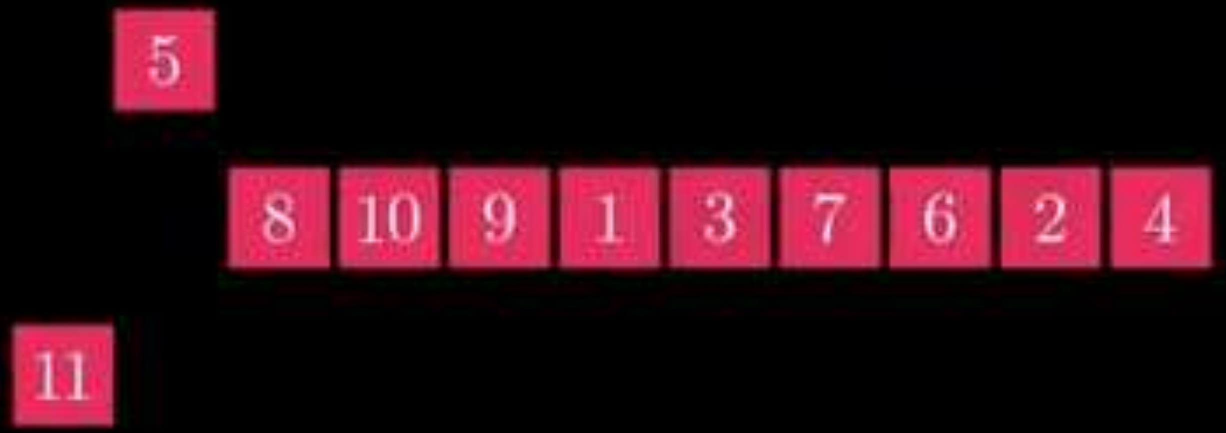
HOW IT WORKS:

How it works: Build a special tree (heap) structure where the highest (or lowest) priority item is always on top, and then repeatedly remove it.

- Real-life Example: Prioritizing the most urgent tasks in a to-do list.

Time Complexity: O(n log n) (for all cases)
Space Complexity: O(1) (in-place, but not stable)

STEPS:



1. Arrange tasks so that the most urgent is always first.
2. Complete the first task.
3. Rearrange the list to find the next most urgent task.

LINEAR TIME SORTING

COUNTING SORT

DEFINITION:

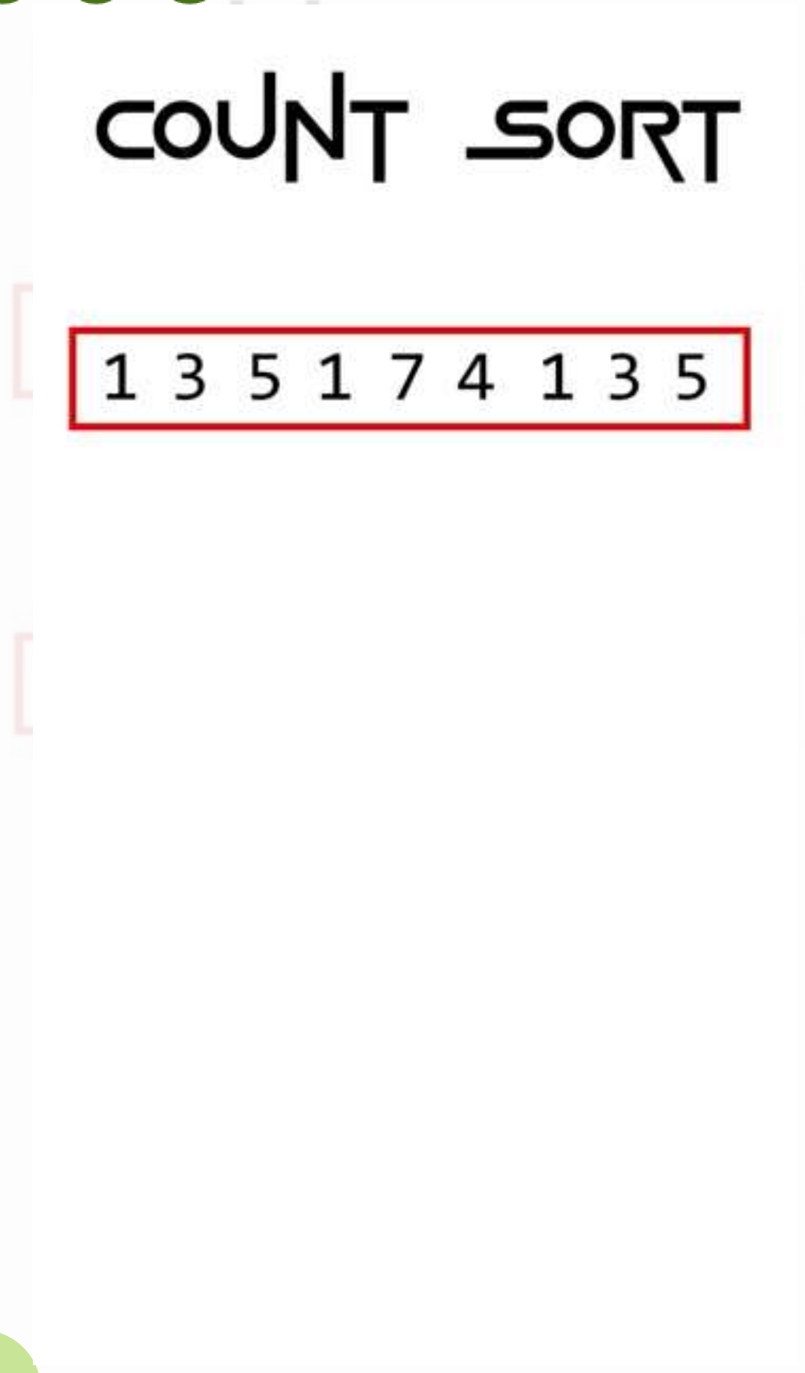
Counting Sort is a non-comparison-based sorting algorithm that sorts integers by counting the number of occurrences of each distinct value, then calculating the position of each element in the output array.

HOW IT WORKS:

Count the number of occurrences of each item, then use this information to place the items in the correct order.

- **Real-life Example:** Counting the number of students in each grade (Grade 1, Grade 2, etc.) and listing them accordingly.

Time Complexity: O(n + k), where k is the range of input
Space Complexity: O(n + k)



Limitation:

Works only for non-negative integers (or requires adaptation for negatives).

LINEAR TIME SORTING

RADIX SORT

DEFINITION:

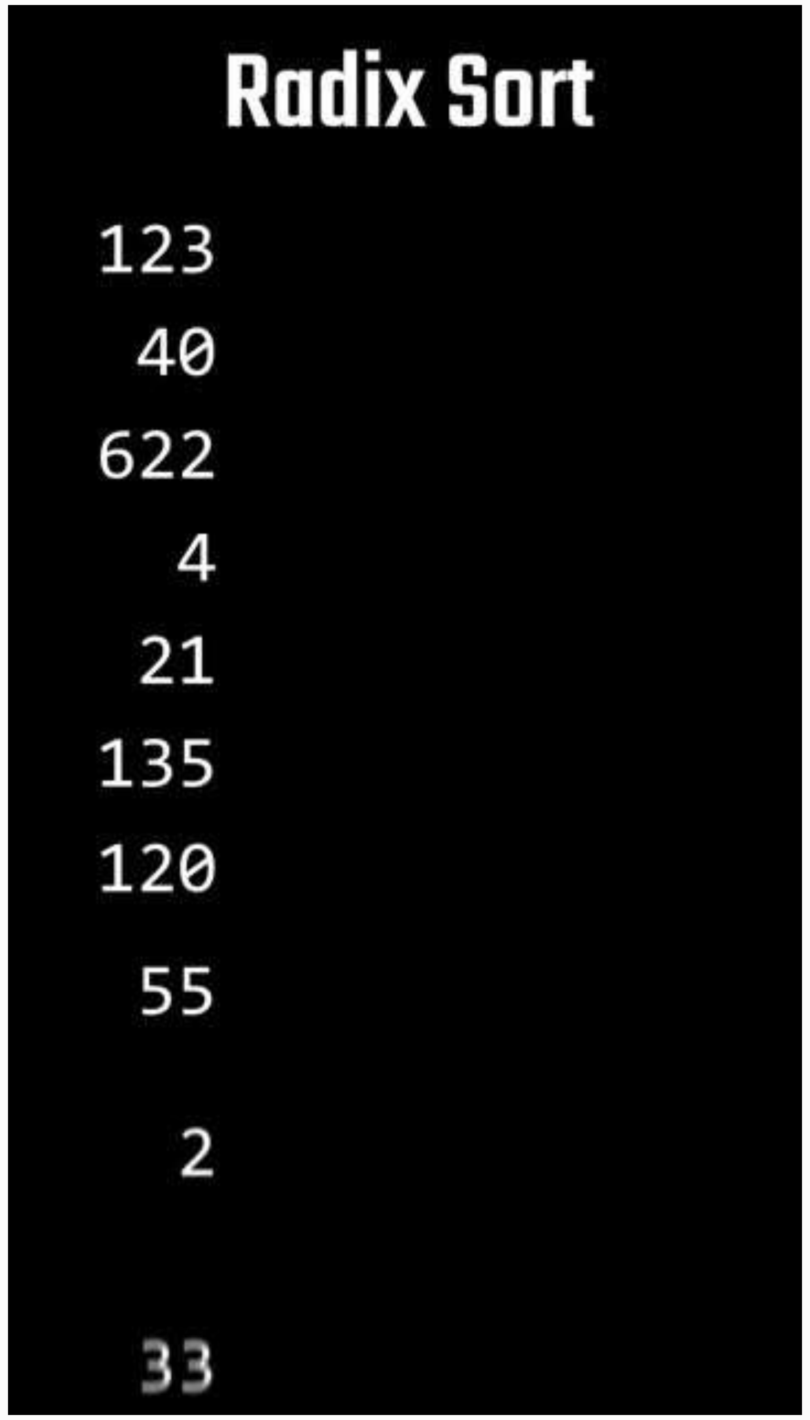
Radix Sort is a non-comparison-based sorting algorithm that sorts numbers by processing individual digits from the least significant digit (LSD) to the most significant digit (MSD), using a stable intermediate sort like Counting Sort at each digit level.

HOW IT WORKS:

How it works: Sort numbers digit by digit, starting from the least significant digit (rightmost).

- Real-life Example: Sorting ID numbers by their last digit, then their second-last, and so on.

- Time Complexity: O(d(n + k)),
- Space Complexity: O(n + k)


Radix Sort

```
123
 40
622
  4
 21
135
120
 55

  2


 33
```

where:

- n = number of elements
- d = number of digits in the largest number
- k = range of digits (usually 0–9)

SEARCHING ALGORITHMS

LINEAR SEARCH

Definition:

Linear Search is a straightforward search algorithm that sequentially checks each element of a list until the target element is found or the list ends.

Time Complexity:
Best: O(1) (target is first element)
Worst: O(n)
Space Complexity: O(1)

SEARCHING ALGORITHMS

LINEAR SEARCH

A. Linear Search

- How it works: Check each item one by one until you find the one you're looking for.
- Real-life Example: Looking through a messy drawer to find a pair of scissors.

SEARCHING ALGORITHMS

BINARY SEARCH

Binary Search is an efficient search algorithm that operates on a sorted list by repeatedly dividing the search interval in half, eliminating half of the remaining elements from consideration each step.

SEARCHING ALGORITHMS

BINARY SEARCH

Time Complexity: O(log n)

Space Complexity:

Iterative: O(l)

Recursive: O(log n) (due to call stack)

Important: Requires the list to be sorted beforehand.

SEARCHING ALGORITHMS

BINARY SEARCH

- How it works: Divide the sorted list into half, check the middle item, and decide whether to look in the left or right half.

- Real-life Example: Looking up a word in a dictionary by starting at the middle.

Important: List must be sorted for Binary Search!

HASH TABLE CONCEPT

Definition:

A Hash Table is a data structure that stores key-value pairs, where the key is mapped to an index in an array using a hash function. This allows for fast retrieval, insertion, and deletion operations on average.

Average Time Complexity: O(1) for insert, search, and delete
Worst-Case Time Complexity: O(n) (if many collisions occur)

HASH TABLES

Hash Table Concept

- How it works: Data is stored using a "key" that maps to a "slot" or "bucket."
- Real-life Example: Assigning mailboxes in a building — each apartment has a unique mailbox based on their apartment number.

COLLISION HANDLING

Collision Handling refers to strategies used in hash tables to deal with two different keys that hash to the same index.

- Common Techniques:
  - Chaining: Store multiple elements at the same index using linked lists.
  - Open Addressing: Find the next available slot based on a probing sequence (e.g., linear probing, quadratic probing, double hashing).

COLLISION HANDLING

- Problem: Sometimes, two keys might map to the same slot.
- Solutions:
  - Chaining: Store multiple items in a list at the same slot.
  - Open Addressing: Find another empty slot based on a rule.

Real-life Example:

If two tenants receive mail but only one mailbox is available, put both mails in the same box (chaining) or redirect one to the next empty mailbox (open addressing).

HASH TABLES AND COLLISION HANDLING

EXAMPLES OF SORTING ALGORITHMS:

Algorithm	Type	Average Time Complexity
Insertion Sort	Comparison-based	$O(n^2)$
Selection Sort	Comparison-based	$O(n^2)$
Heap Sort	Comparison-based	$O(n \log n)$
Counting Sort	Non-comparison-based	$O(n + k)$
Radix Sort	Non-comparison-based	$O(d(n + k))$

THANK YOU