SEARCHING & SORTING ALGORITHMS

Sunday, December 28, 2025 12:26 AM

SEARCHING ALGORITHMS

Goal: Find index of Target T in List L.

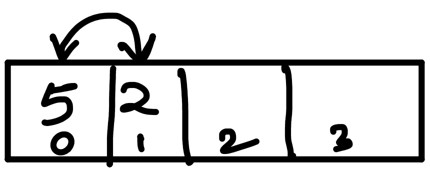
	LINEAR SEARCH	BINARY SEARCH
Concept:	The Brute Force	Divide & Conquer
Requirement:	None (works on unsorted)	MUST BE SORTED
How:	check $i=0, i=1, i=2 \dots$ until found.	Check middle. If $T < Mid$, go left. If $T > Mid$, go right.
Visual:		
Complexity:	$O(n)$ slow	$O(\log n)$ Fast

Key difference: Linear is like reading every page of a book. Binary is like opening a dictionary to the middle, then deciding to go left or right.

ELEMENTARY SORTS ($O(n^2)$)

1. BUBBLE SORT

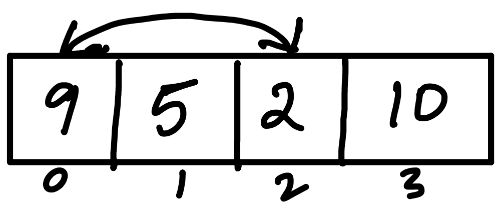
- The Hook: Bubbling up
- Mechanism: Compare adjacent pairs $(i, i+1)$. Swap if wrong order. Largest element moves to the end in each pass.



- Best For: Educational purposes only.
- Complexity: $O(n^2)$

2. SELECTION SORT

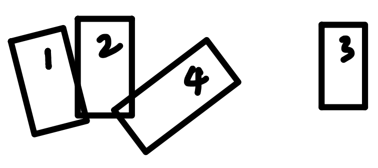
- The Hook: Select the Smallest
- Mechanism: Scan the entire unsorted side to find the absolute minimum. Swap it to the front.



- Best For: When writing to memory (swapping) is expensive.
- Complexity: $O(n^2)$

3. INSERTION SORT

- The hook: Playing Cards
- Mechanism: Pick next item. Shift sorted items to the right to make a gap. Insert item.



- Best For: Small or nearly sorted lists.
- Complexity: $O(n^2)$ (but $O(n)$ if nearly sorted.)

THE "CHEAT SHEET" CARD

Algorithm	Time (Avg)	Space	Stable?
Linear	$O(n)$	$O(1)$	-
Binary	$O(\log n)$	$O(1)$	-
Bubble	$O(n^2)$	$O(1)$	Yes
Selection	$O(n^2)$	$O(1)$	No
Insertion	$O(n^2)$	$O(1)$	Yes