Real Time Complexity and Assumed Time Complexity of Array

Array Operation	*Real Time Complexity*	*Assumed Time Complexity*
$Access\ i-th\ element$	$O(\sqrt{n})$	$O(1)$
$Traverse\ all\ elements$	$O(n+\sqrt{n})$	$O(n)$
$Override\ element\ at\ i-th\ index$	$O(\sqrt{n})$	$O(1)$
$Insert\ Element\ at\ E$	$O(n+\sqrt{n})$	$O(n)$
$Delete\ Element\ at\ E$	$O(n+\sqrt{n})$	$O(n)$

❗ Why This Is NOT Used in Standard DSA Analysis

In typical computer science time complexity:

✅ We assume:

- data is in **main memory (RAM)**
- address arithmetic is O(1)
- basic operations cost constant time

So:

- access by index → O(1)
- traverse → O(n)
- shift for insert/delete → O(n)

No √N terms appear in standard array time complexity.

The √N cost is a **hardware-level overhead**, not a complexity measure used in algorithm design.

🟡 When √N Applies

The √N time idea is relevant when modeling:

- External memory access
- Paging and cache loads
- Disk-to-RAM block transfer

This belongs to **systems-level performance models**, not the common algorithmic time complexity used in DSA courses.

If the chip (the library) is arranged as a square grid of memory cells:

- ***The width of the chip is roughly*** $\sqrt{\{N\}}$.
- ***To reach the farthest memory cell, the signal has to travel physically across the chip***.
- **The bigger the chip** $\sqrt{N}$**, the longer the travel time.**
- ***Time Complexity***: $O(\sqrt{\{N\}})$ (***Time depends on physical distance***).
