**Insertion Sort Steps for [22,27,16,2,18,6]**

Insertion Sort works by building a sorted portion of the array one element at a time. Here's how the array evolves step-by-step:

1. Start with [22] — already sorted.
2. Insert 27: [22,27] — no change needed.
3. Insert 16: compare with 27 and 22, insert before both → [16,22,27]
4. Insert 2: insert before all → [2,16,22,27]
5. Insert 18: compare with 27, 22, 16 → insert between 16 and 22 → [2,16,18,22,27]
6. Insert 6: insert between 2 and 16 → [2,6,16,18,22,27]

Final sorted array: [2,6,16,18,22,27]

**Big-O Notation**

* **Best Case (already sorted)**: O(n)O(n)
* **Worst Case (reverse order)**: O(n2)O(n^2)
* **Average Case**: O(n2)O(n^2)

Insertion Sort is efficient for small or nearly sorted datasets but scales poorly with large, unordered data.

**Time Complexity Case for Number 18**

After sorting, 18 is in the middle of the array: [2,6,16,18,22,27]

* **Position**: Index 3 (0-based), which is central.
* **Case**: **Average Case** — because the number is in the middle.