

7

Recursive Insertion Sort

EASY

■ What You Will Learn

- ✓ The recursive formulation of insertion sort
- ✓ How to think: 'sort n-1 elements, then insert the nth'
- ✓ Clean recursive code vs iterative

■ Intuition First

Instead of a loop that grows the sorted section one element at a time, we think recursively: 'Sort the first n-1 elements first, then insert the nth element into its correct position in that sorted section.' The recursive call handles sorting, the current call handles the insertion.

■ Recursive Thinking

- Base case: $n == 1$ (one element is sorted)
- Recursive case: sort first n-1 elements, then insert `arr[n-1]` into sorted portion
- Insertion = shift elements right until correct spot found

■ Recursion Unwind Visualization

```
rInsert([4,3,2,1], n=4):  
→ rInsert(arr, 3) first...  
→ rInsert(arr, 2)...  
→ rInsert(arr, 1): BASE CASE ✓ [4]  
← Insert arr[1]=3 into [4]: [3,4]  
← Insert arr[2]=2 into [3,4]: [2,3,4]  
← Insert arr[3]=1 into [2,3,4]: [1,2,3,4] ✓
```

■ Java Implementation

```
public static void recursiveInsertionSort(int[] arr, int n) {  
    // Base case: 1 element is already sorted  
    if (n <= 1) return;  
    // Recursively sort first n-1 elements  
    recursiveInsertionSort(arr, n - 1);  
    // Insert the nth element (arr[n-1]) into sorted portion  
    int last = arr[n - 1];  
    int j = n - 2;
```

```
// Shift elements right until correct position found
while (j >= 0 && arr[j] > last) {
    arr[j + 1] = arr[j];
    j--;
}
arr[j + 1] = last; // Place element in correct spot
}

// Caller: recursiveInsertionSort(arr, arr.length);
```

■ Time & Space Complexity

Case	Time	Space	What happens?
Best	$O(n)$	$O(n)$	Sorted input — while loop doesn't run, but recursion stack still $O(n)$
Average	$O(n^2)$	$O(n)$	Recursion depth n + inner while loop work
Worst	$O(n^2)$	$O(n)$	Reverse sorted — max shifts + max recursion depth

■ Compare to Iterative

- Time: same $O(n^2)$ — recursion doesn't help
- Space: $O(n)$ recursion stack vs $O(1)$ iterative — recursion is worse
- Code: some find recursive version more intuitive to reason about
- Interview tip: be ready to explain the trade-off clearly

■ Common Mistake

- Calling `recursiveInsertionSort(arr, n-1)` BEFORE doing the insertion — order matters!
- The recursive call must come first, insertion happens on the way back up the call stack

■ Practice Problems

[Easy] Trace recursive insertion sort on [5,4,3,2,1]

[Easy] Count how many shifts happen for [3,1,2]

[Med] Convert to sort only a subarray: sort `arr[l..r]` recursively

■ Quick Revision

- Recurse on $n-1$ first, then insert n th element

- Base case: $n == 1$
- Same $O(n^2)$ time as iterative, but $O(n)$ space
- Good exercise in recursive thinking