

6

Recursive Bubble Sort

EASY

■ What You Will Learn

- ✓ How to convert an iterative algorithm to recursive
- ✓ The base case and recursive case for bubble sort
- ✓ Why recursion doesn't improve performance here
- ✓ How to think recursively about sorting

■ Intuition First

Same idea as regular bubble sort — but instead of an outer for loop, you use recursion. One pass bubbles the largest element to end, then you recursively call the same function on the first $n-1$ elements. Base case: array of size 1 is already sorted.

■ Recursive Thinking

- If $n == 1$: return (base case — 1 element is always sorted)
- Do one full pass of bubble sort on $\text{arr}[0..n-1]$
- Recursively call on $\text{arr}[0..n-2]$ (last element is now sorted)

■ Recursion Stack Visualization

```
Call: rBubble([5,3,1,4], n=4)
→ Pass: [3,1,4,5] → Call rBubble(arr, n=3)
→ Pass: [1,3,4,5] → Call rBubble(arr, n=2)
→ Pass: [1,3,4,5] → Call rBubble(arr, n=1)
→ n==1: BASE CASE, return ✓
→ Unwind: [1,3,4,5] fully sorted ✓
```

■ Java Implementation

```
public static void recursiveBubbleSort(int[] arr, int n) {
    // Base case: array of size 1 is sorted
    if (n == 1) return;

    // One pass: bubble largest to end
    for (int i = 0; i < n - 1; i++) {
        if (arr[i] > arr[i + 1]) {
            int temp = arr[i];
```

```
arr[i] = arr[i + 1];
arr[i + 1] = temp;
}
}

// Recursively sort first n-1 elements
recursiveBubbleSort(arr, n - 1);
}

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

■ Time & Space Complexity

Case	Time	Space	What happens?
Best	$O(n^2)$	$O(n)$	Recursion stack depth = n (unlike iterative's $O(1)$)
Average	$O(n^2)$	$O(n)$	Same comparisons as iterative version
Worst	$O(n^2)$	$O(n)$	Stack depth grows to n — stack overflow risk!

■ Key Difference from Iterative

- Iterative: $O(1)$ space (just a few variables)
- Recursive: $O(n)$ space for the call stack
- Same time complexity — recursion doesn't help here
- Taught to practice recursive thinking, not for production use

■ Watch Out

- No easy way to add the 'swapped' optimization recursively
- Deep recursion on large arrays will cause `StackOverflowError` in Java
- Always pass n (not `arr.length`) — the range shrinks each call

■ Practice Problems

[Easy] Trace recursive bubble sort on `[4,2,1,3]` — draw call stack

[Easy] What is the recursion depth for an array of size 8?

[Med] Add the 'swapped' optimization to recursive bubble sort

■ Quick Revision

- Base case: $n == 1$, return
- One pass bubbles largest to end, recurse on $n-1$
- $O(n^2)$ time, $O(n)$ space — worse than iterative
- Good for practicing recursion, not production code