Merge sort

Merge sort is a
divide and conquer algorithm.

Conceptually, a merge sort works as follows:

1. Divide the unsorted list into n sublists, each containing one element (a list of one element is considered sorted).
2. Repeatedly merge sublists to produce new sorted sublists until there is only one sublist remaining. This will be the sorted list.

Two ways to implement

Top-down implementation

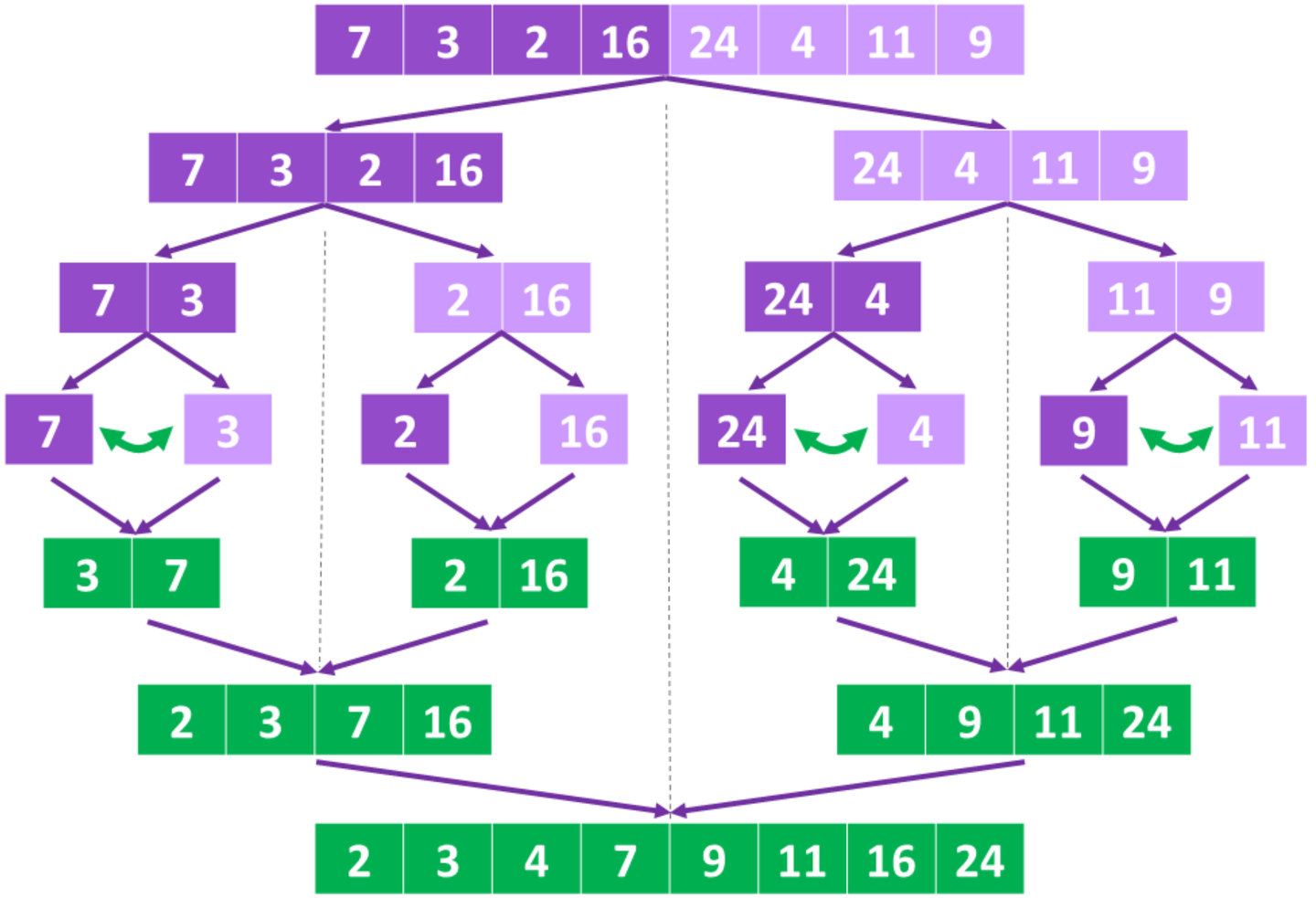
recursive

Bottom-up implementation

non-recursive

Top-down implementation

Merge Sort



Step 1:
Split sub-lists in two until you reach pair of values.

Step 3:
Sort/swap pair of values if needed.

Step 4:
Merge and sort sub-lists and repeat process till you merge to the full list.

Merge operation

```
function merge(left, right) is
    var result := empty list

    while left is not empty and right is not empty do
        if first(left) ≤ first(right) then
            append first(left) to result
            left := rest(left)
        else
            append first(right) to result
            right := rest(right)

    // Either left or right may have elements left; consume them.
    // (Only one of the following loops will actually be entered.)
    while left is not empty do
        append first(left) to result
        left := rest(left)
    while right is not empty do
        append first(right) to result
        right := rest(right)
    return result
```

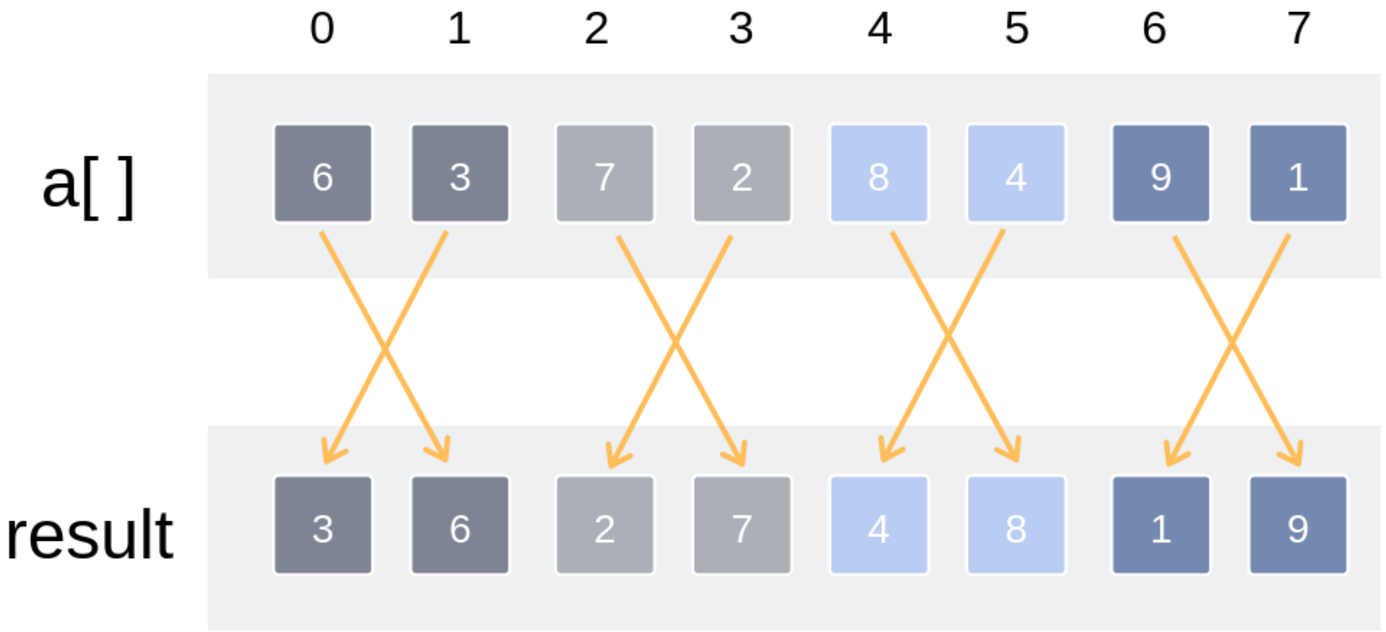
Top-down implementation

```
function merge_sort(list m) is
    // Base case. A list of zero or one elements is sorted, by definition.
    if length of m ≤ 1 then
        return m

    // Recursive case. First, divide the list into equal-sized sublists
    // consisting of the first half and second half of the list.
    // This assumes lists start at index 0.
    var left := empty list
    var right := empty list
    for each x with index i in m do
        if i < (length of m)/2 then
            add x to left
        else
            add x to right

    // Recursively sort both sublists.
    left := merge_sort(left)
    right := merge_sort(right)

    // Then merge the now-sorted sublists.
    return merge(left, right)
```

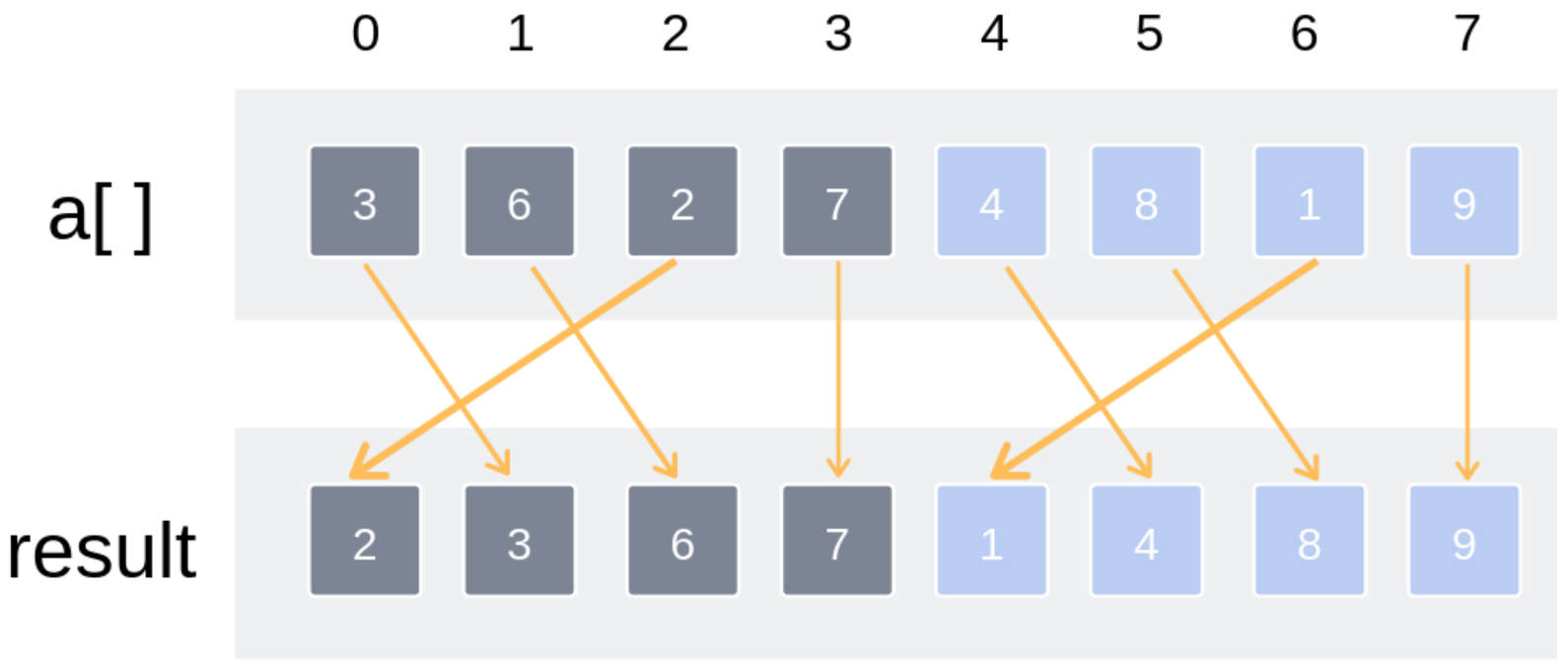
The Bottom-Up merge sort approach uses iterative methodology.

It starts with the “single-element” array, and combines two adjacent elements and also sorting the two at the same time. The combined-sorted arrays are again combined and sorted with each other until one single unit of sorted array is achieved.

Merge pairs of arrays of size 1



Merge pairs of arrays of size 2



Merge pairs of arrays of size 4