MERGE SORT

MergeSort Algorithm

- MergeSort is a *divide and conquer* method of sorting
- MergeSort is a recursive sorting procedure that uses at most **O(n lg(n))** comparisons.
- To sort an array of **n** elements, we perform the following steps in sequence:
- If **n < 2** then the array is already sorted.
- Otherwise, **n > 1**, and we perform the following three steps in sequence:
  1. **Sort** the **left half** of the the array using MergeSort.
  2. **Sort** the **right half** of the the array using MergeSort.
  3. **Merge** the sorted left and right halves.

How to Merge

Here are two lists to be merged:

First: (12, 16, 17, 20, 21, 27)

Second: (9, 10, 11, 12, 19)

Compare **12** and **9**

First: (12, 16, 17, 20, 21, 27)

Second: (10, 11, 12, 19)

New: (9)

Compare **12** and **10**

First: (12, 16, 17, 20, 21, 27)

Second: (11, 12, 19)

New: (9, 10)

Merge Example

Compare **12** and **11**

- **First: (12, 16, 17, 20, 21, 27)**
- **Second: (12, 19)**
- **New: (9, 10, 11)**

Compare **12** and **12**

- **First: (16, 17, 20, 21, 27)**
- **Second: (12, 19)**
- **New: (9, 10, 11, 12)**

Merge Example

Compare **16** and **12**

- **First: (16, 17, 20, 21, 27)**
- **Second: (19)**
- **New: (9, 10, 11, 12, 12)**

Compare **16** and **19**

- **First: (17, 20, 21, 27)**
- **Second: (19)**
- **New: (9, 10, 11, 12, 12, 16)**

Merge Example

Compare **17** and **19**

First: (20, 21, 27)

Second: (19)

New: (9, 10, 11, 12, 12, 16, 17)

Compare **20** and **19**

First: (20, 21, 27)

Second: ( )

New: (9, 10, 11, 12, 12, 16, 17, 19)

Merge Example

Checkout **20** and **empty list**

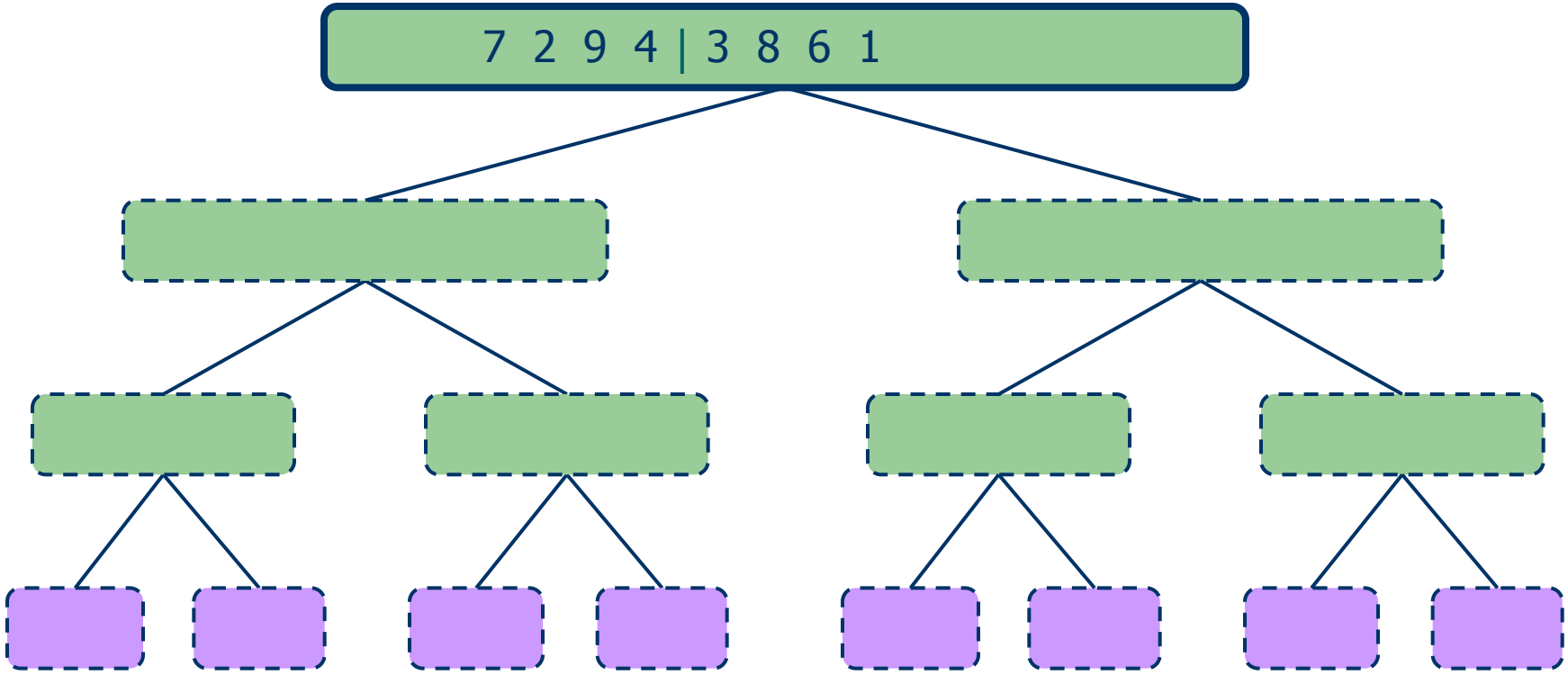
First: ( )

Second: ( )

New: (9, 10, 11, 12, 12, 16, 17, 19, 20, 21, 27)

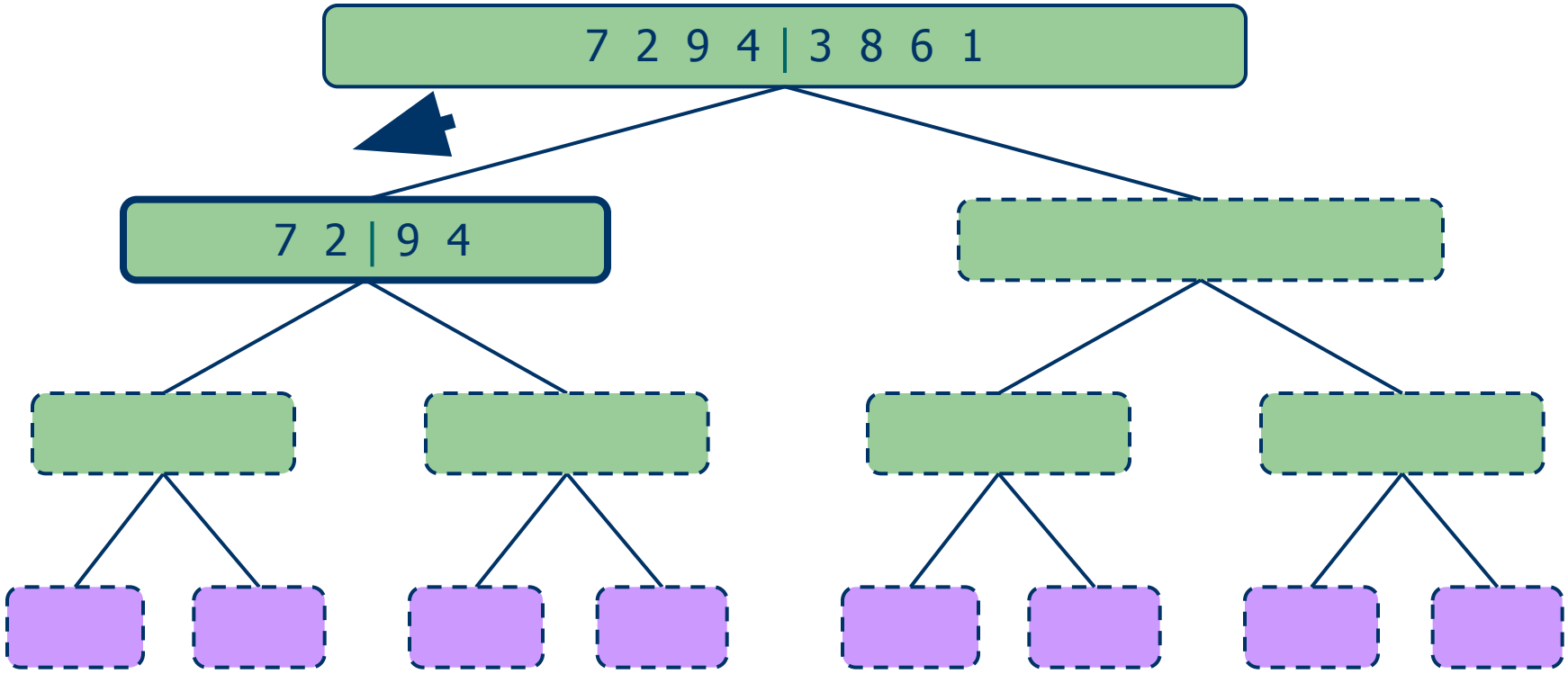
Execution Example

- Partition



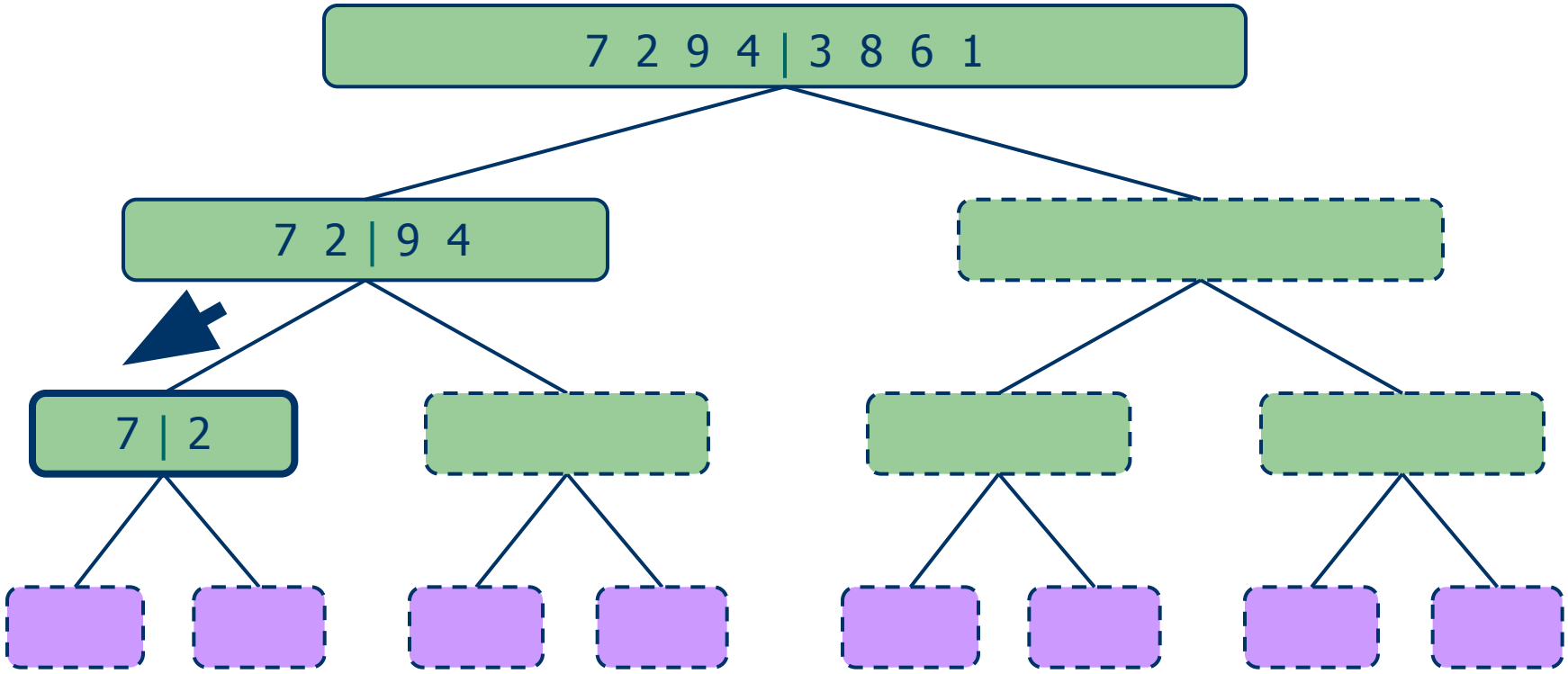
Execution Example (cont.)

- Recursive call, partition



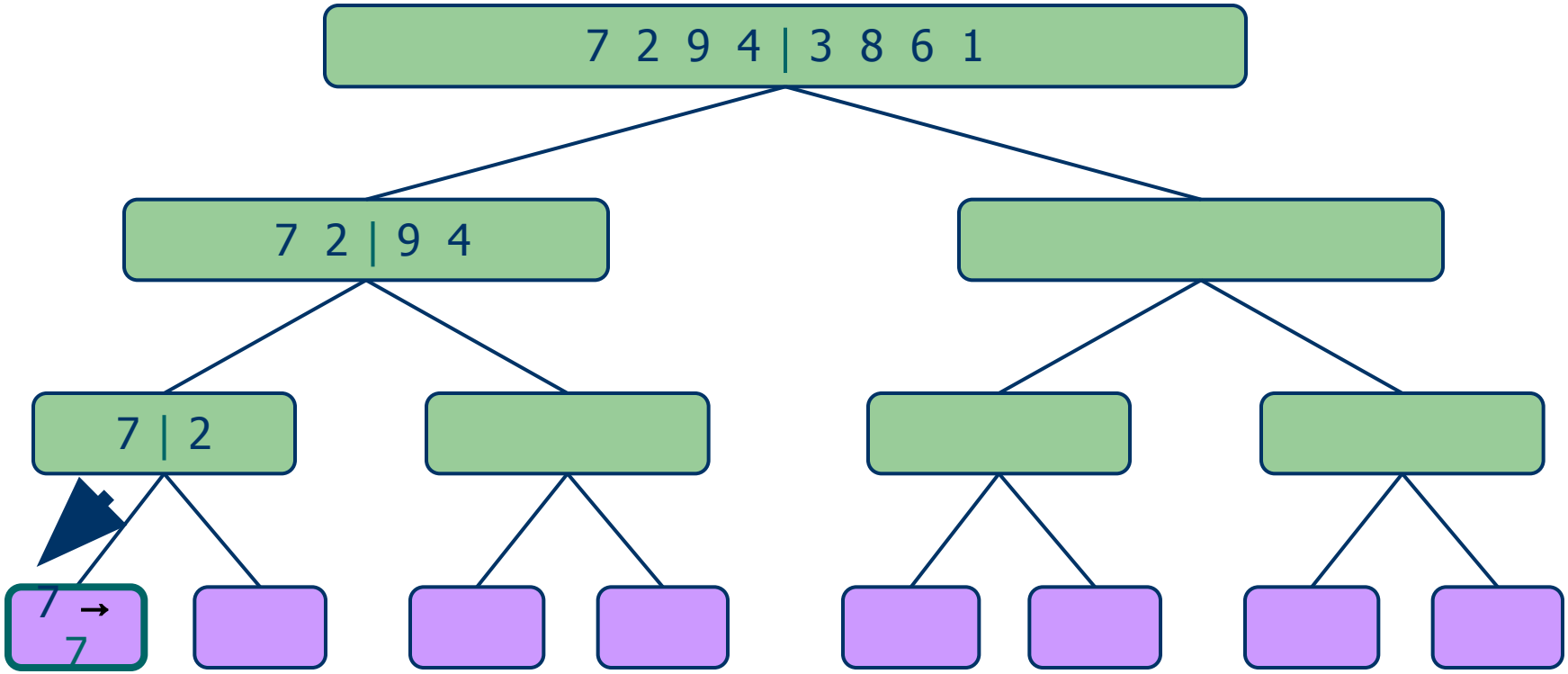
Execution Example (cont.)

- Recursive call, partition



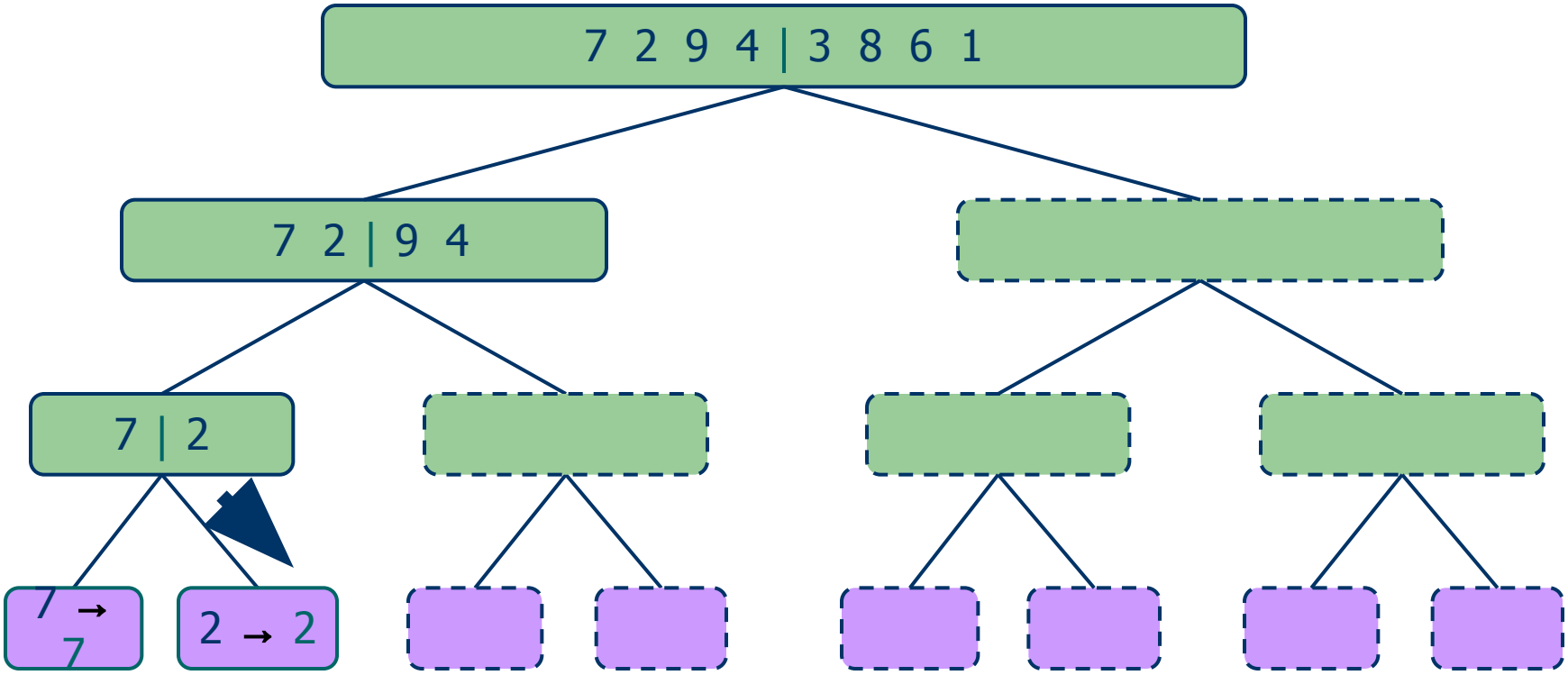
Execution Example (cont.)

- Recursive call, base case



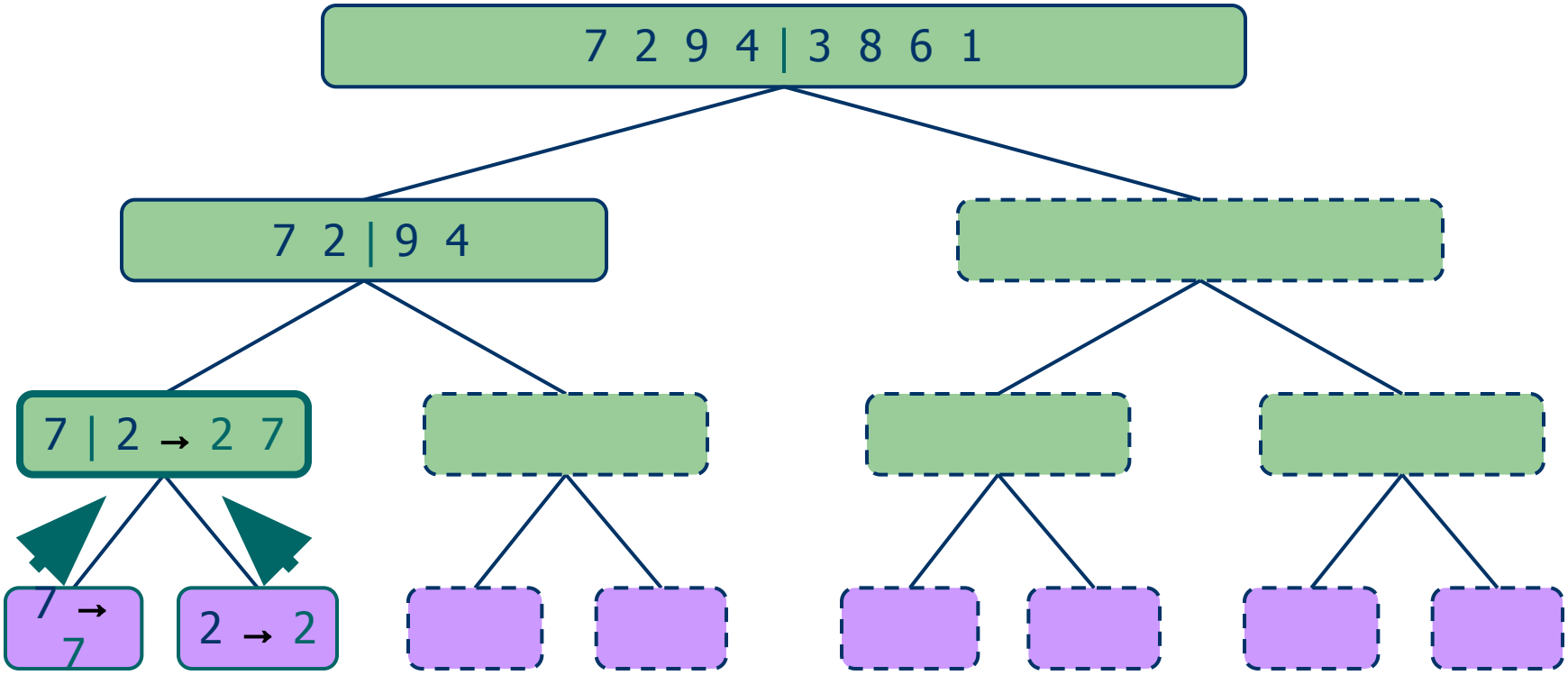
Execution Example (cont.)

- Recursive call, base case



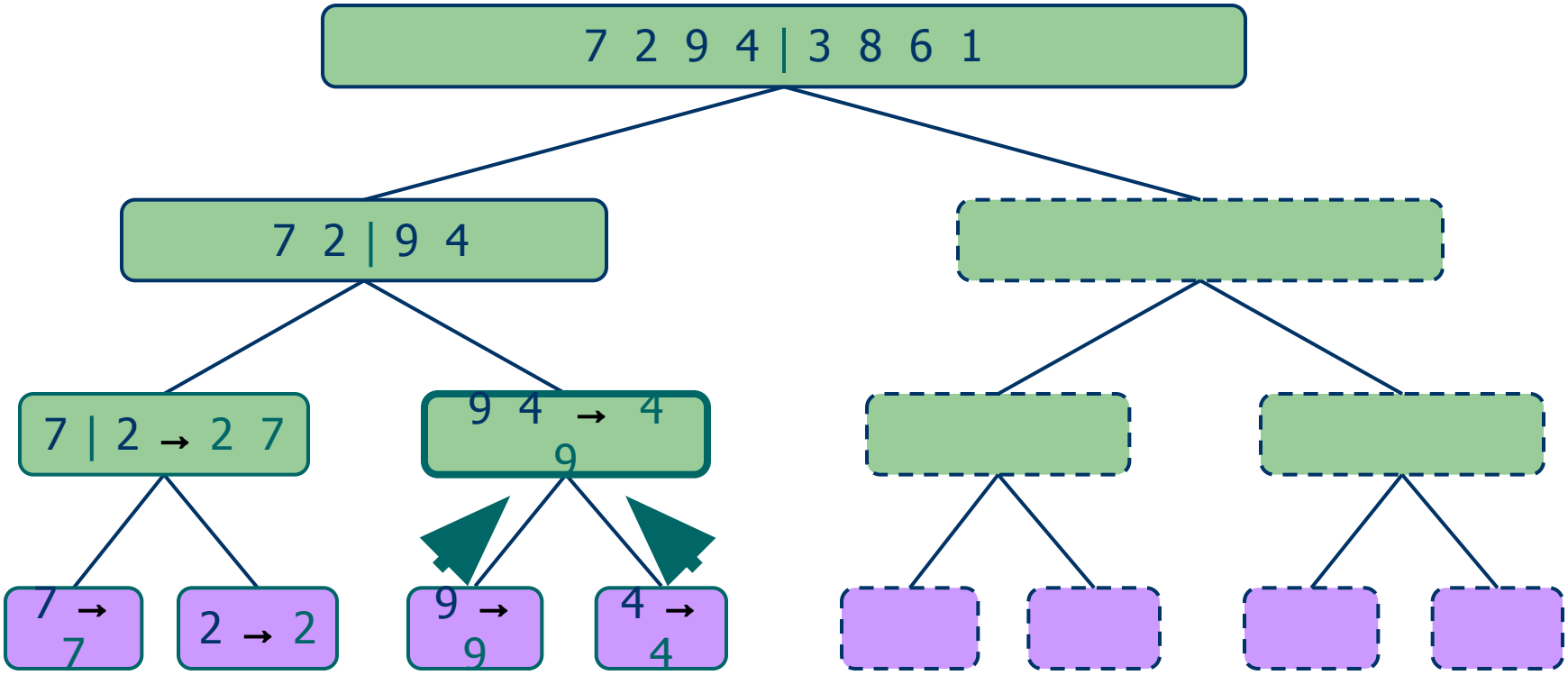
Execution Example (cont.)

- Merge



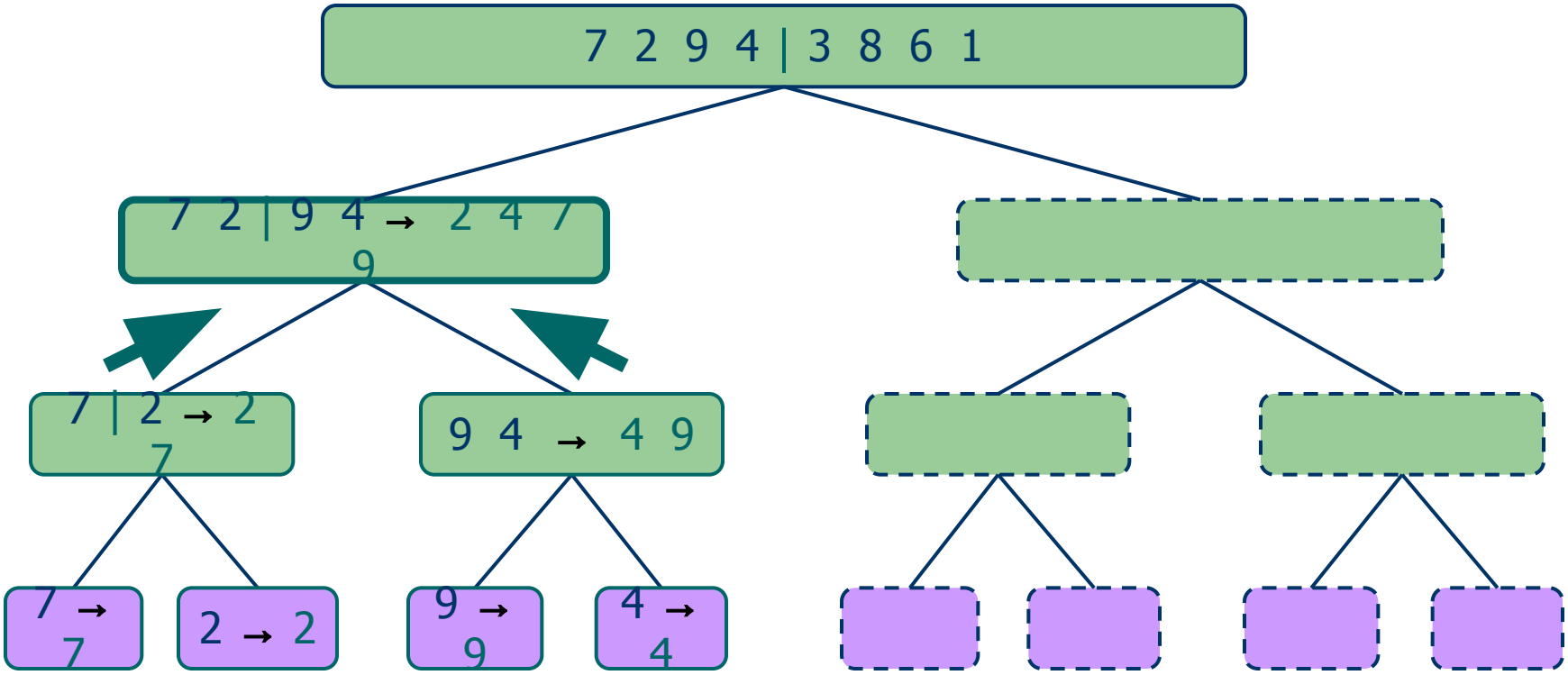
Execution Example (cont.)

- Recursive call, …, base case, merge



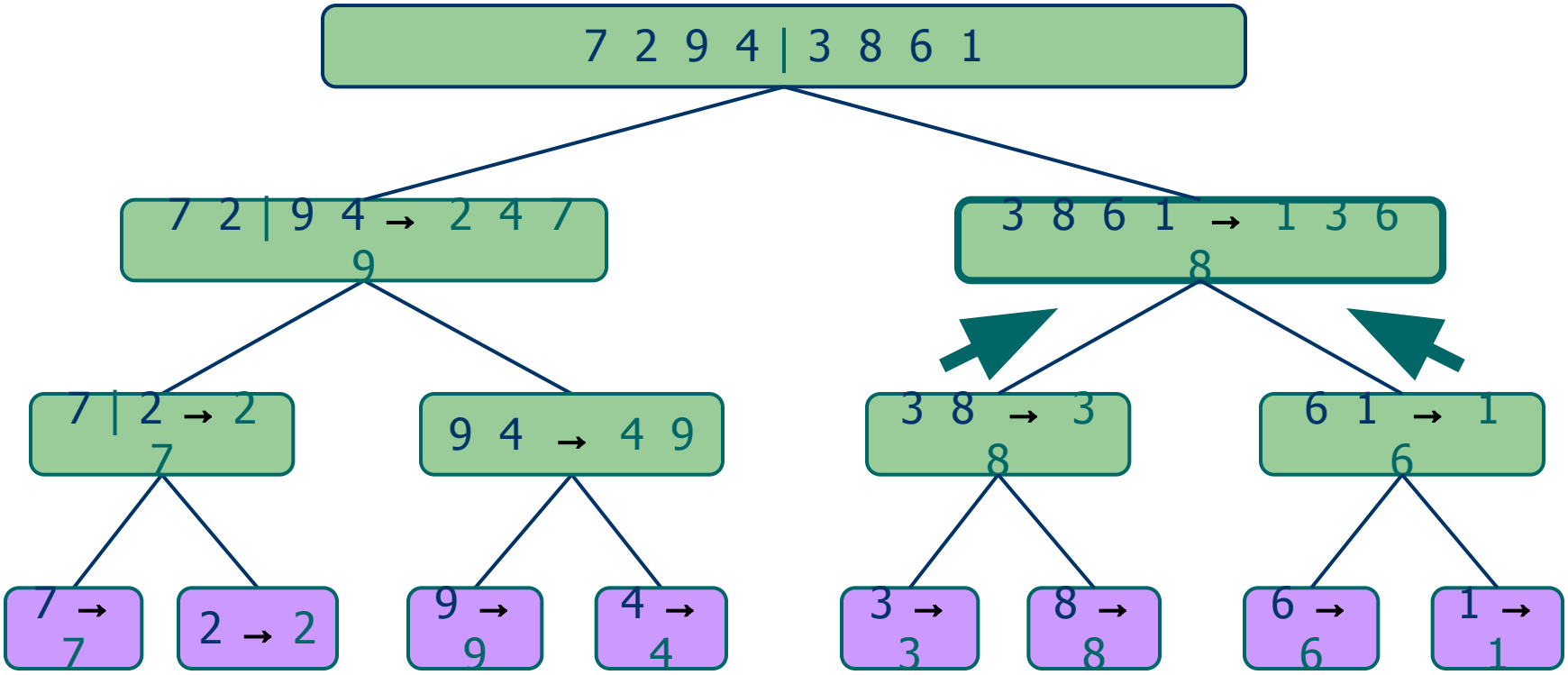
Execution Example (cont.)

- Merge



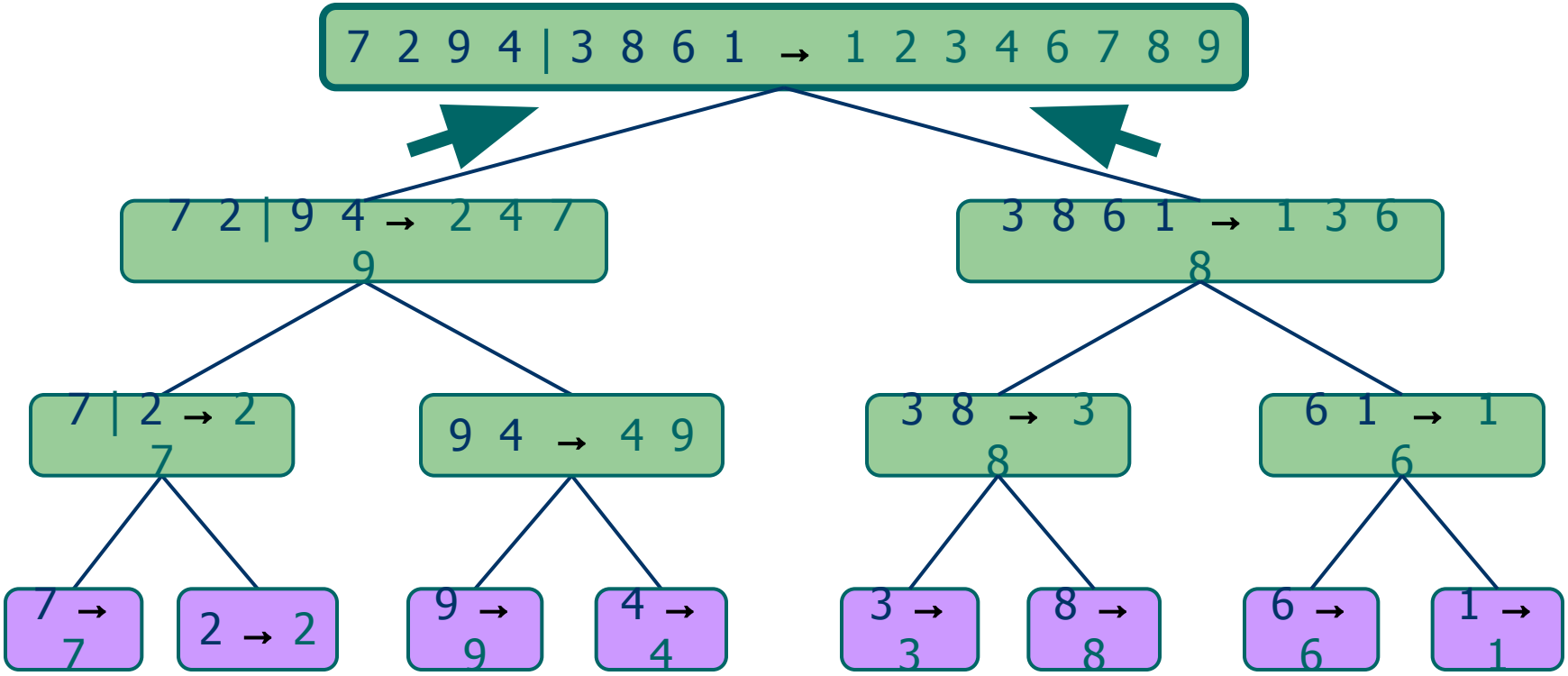
Execution Example (cont.)

- Recursive call, …, merge, merge



Execution Example (cont.)

- Merge



Time Complexity

- Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation.

$$T(n) = 2T(n/2) + \theta(n)$$

- Time complexity of Merge Sort is $\theta(nLogn)$ in all 3 cases (worst, average and best) as merge sort always divides the array into two halves and takes linear time to merge two halves.