Data Structures and Algorithms

Merge Sorting Algorithm

Amilcar Aponte

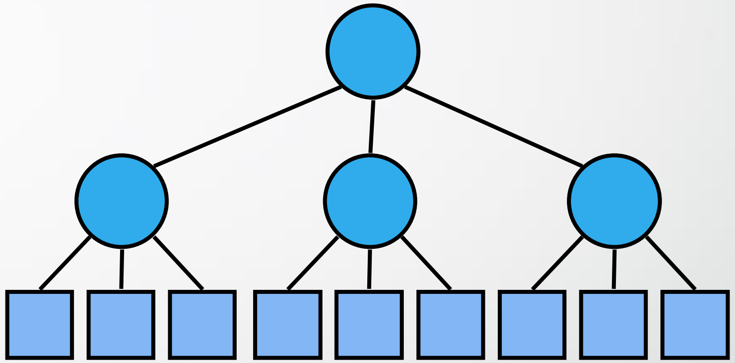
amilcar@cct.ie

Today’s Plan

- Merge Sort Algorithm

Divide-and-Conquer

- Divide-and conquer is a general algorithm design paradigm:
  - Divide: divide the input data $\boldsymbol{S}$ in two or more disjoint subsets $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …
  - Conquer: solve the subproblems recursively
  - Combine: combine the solutions for $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …, into a solution for $\boldsymbol{S}$
- The base case for the recursion are subproblems of constant size
- Analysis can be done using recurrence equations



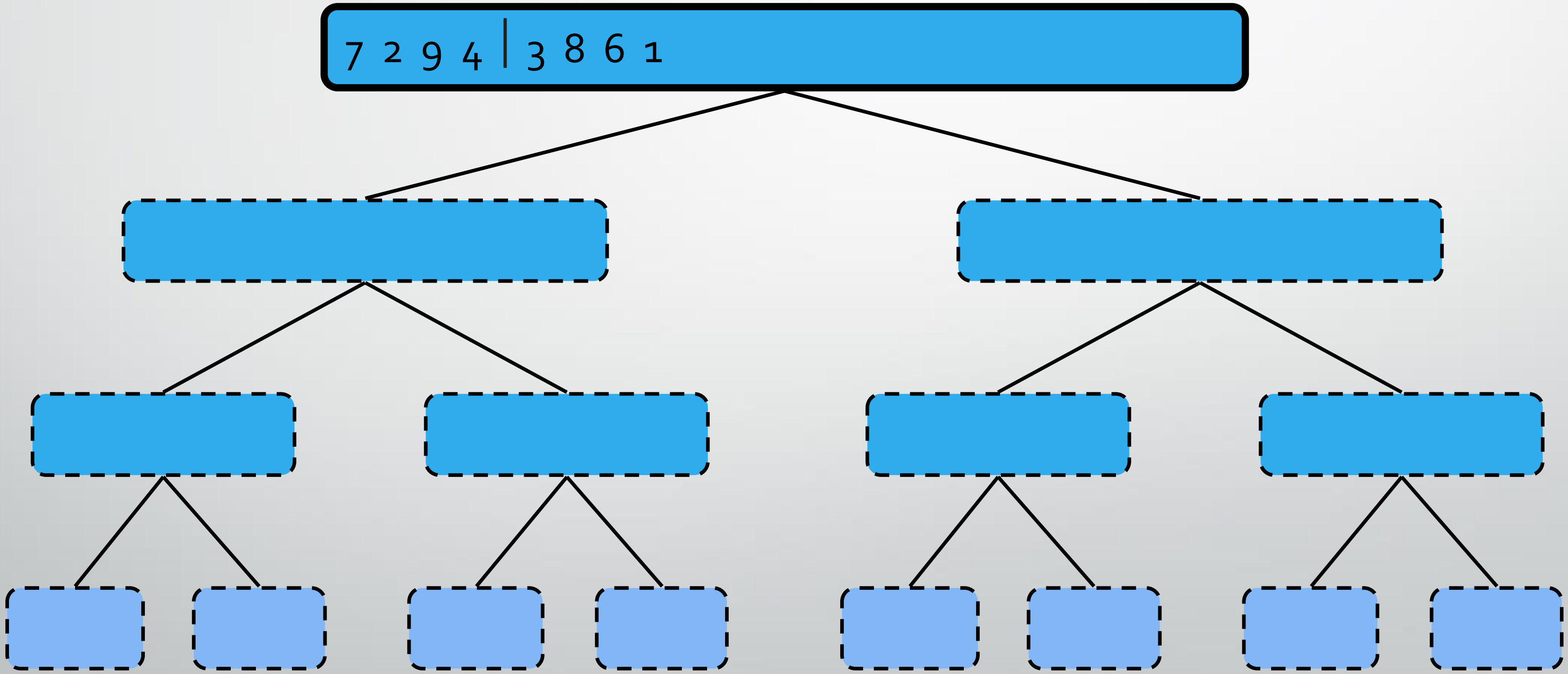
- Merge-sort is a sorting algorithm based on the divide-and-conquer paradigm

Merge Sort

- ***Merge sort*** orders values by recursively dividing the list in half until each sub-list has one element, then recombining
- More specifically:
  - divide the list into two roughly equal parts
  - recursively divide each part in half, continuing until a part contains only one element
  - merge the two parts into one sorted list
  - continue to merge parts as the recursion unfolds

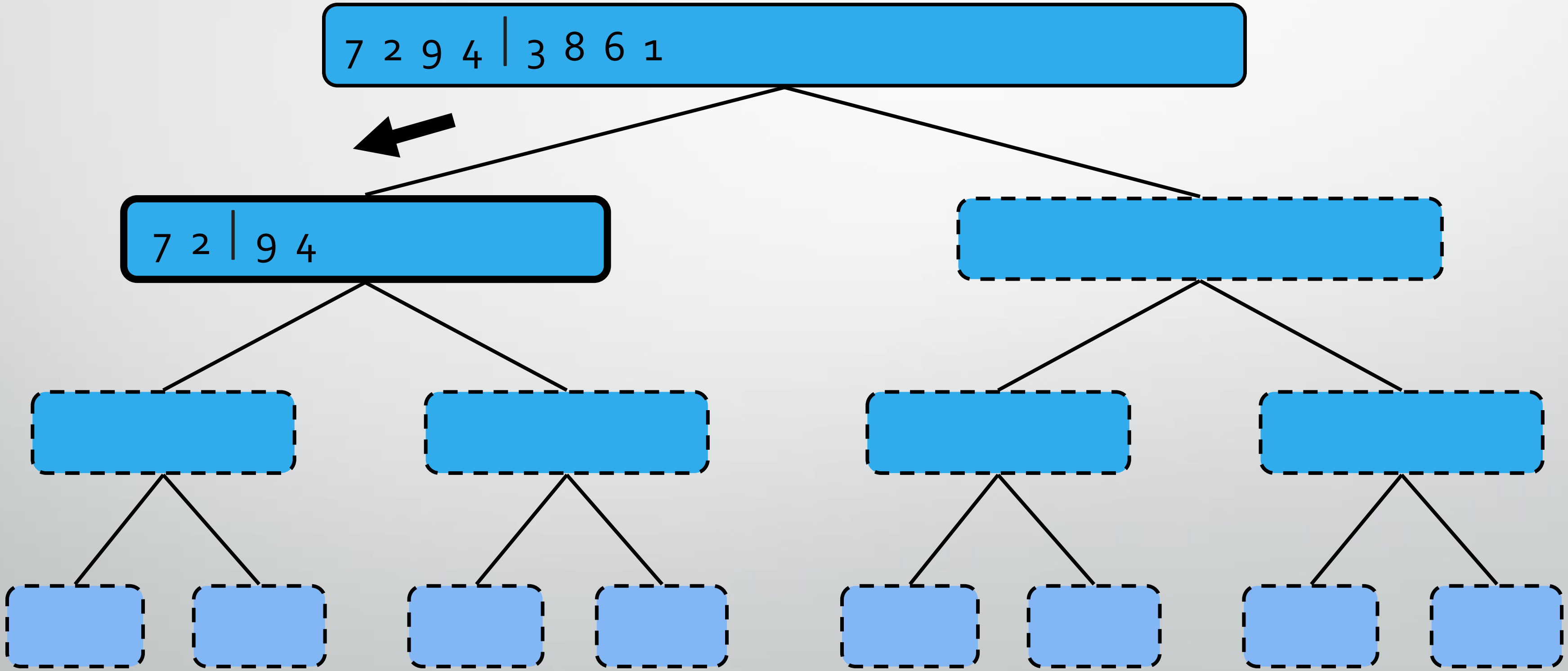
Execution Example

- Partition



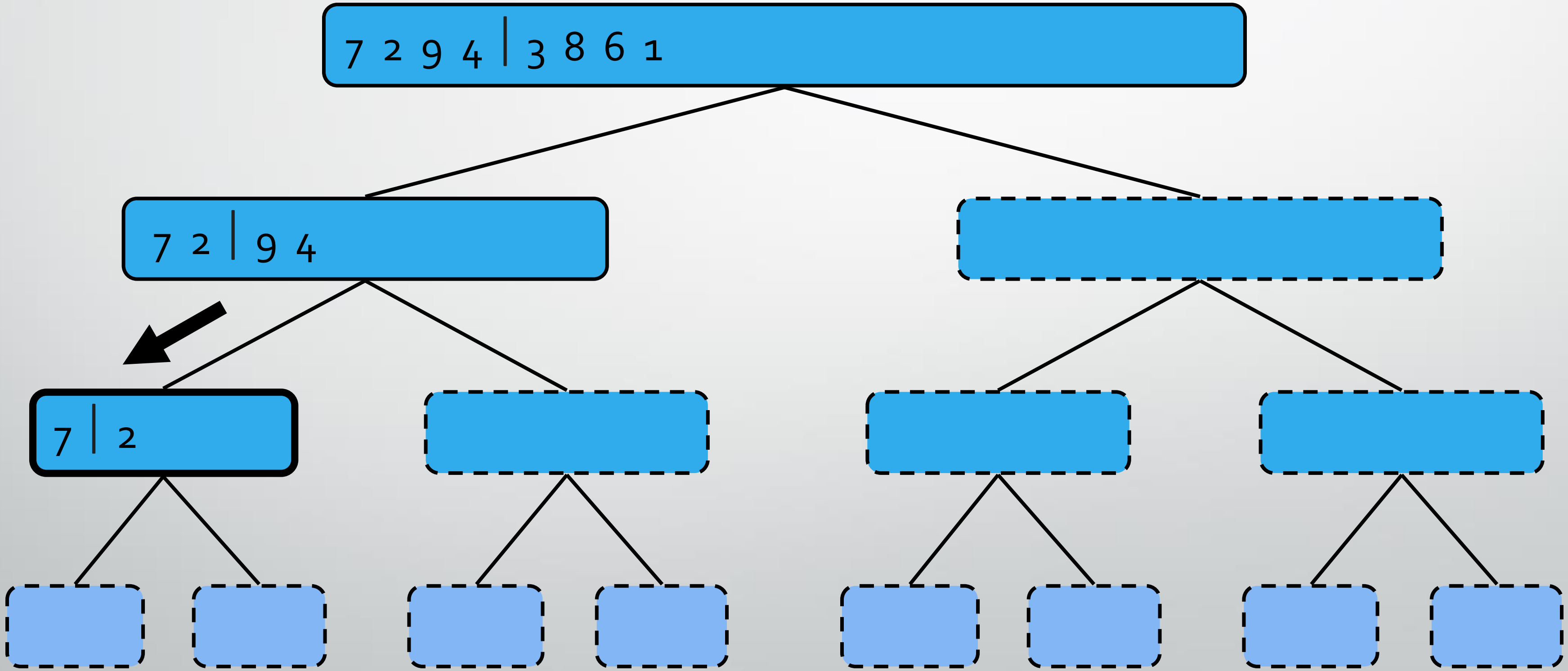
Execution Example

- Recursive call, partition



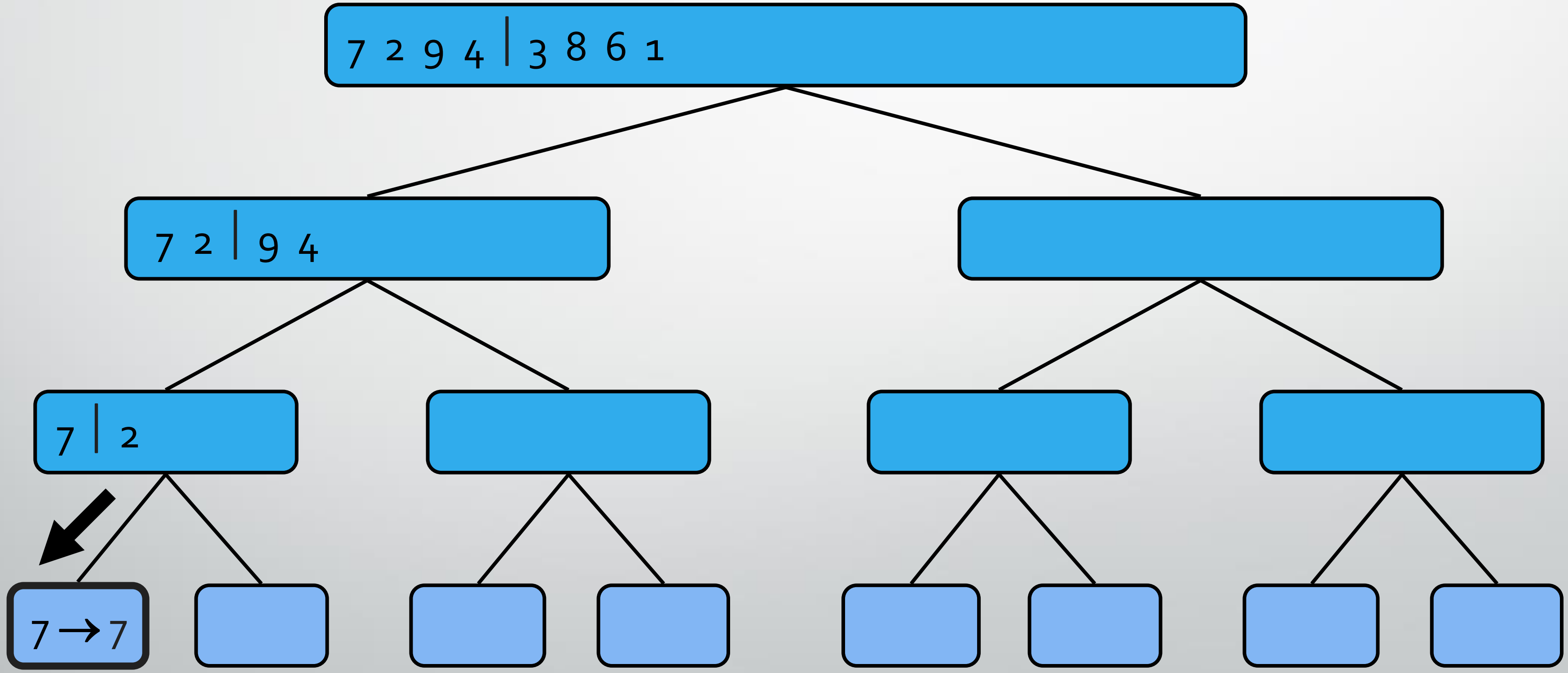
Execution Example

- Recursive call, partition



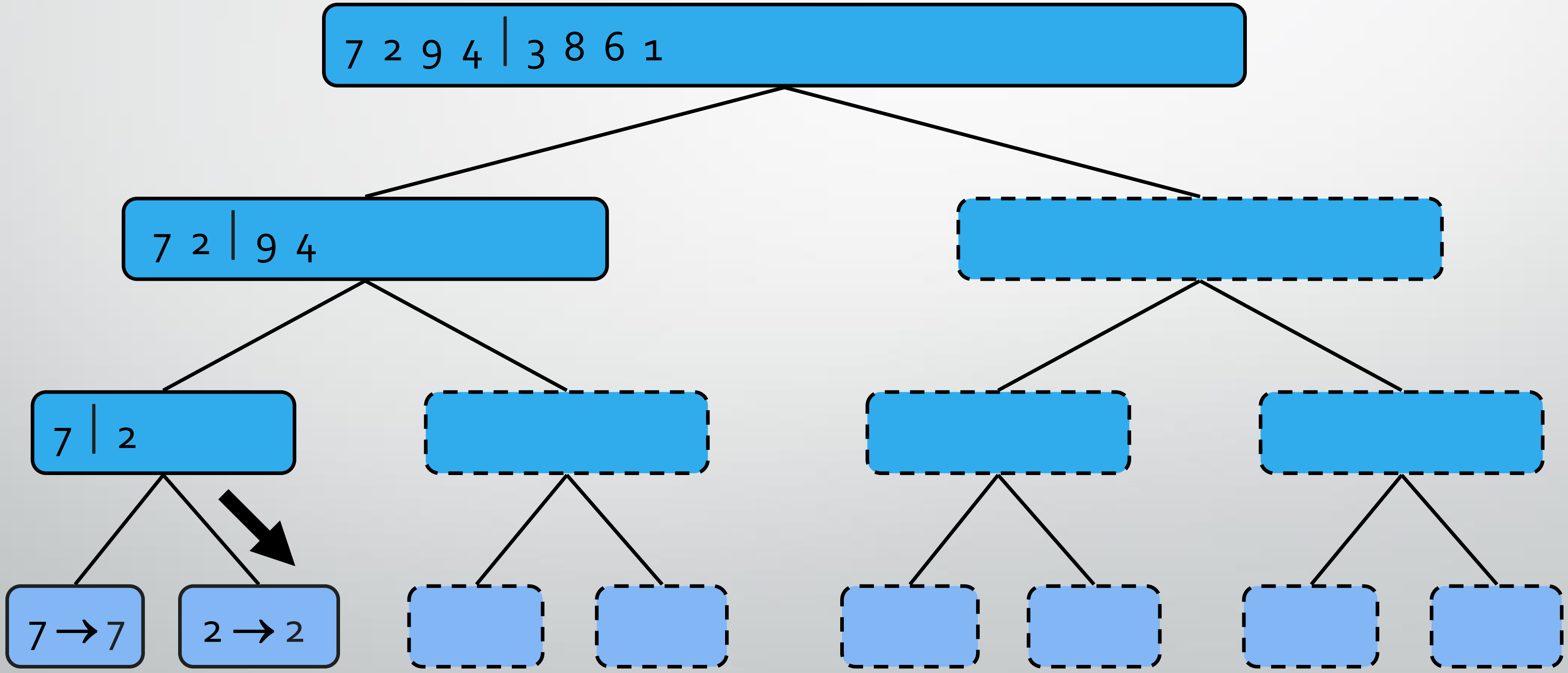
Execution Example

- Recursive call, base case



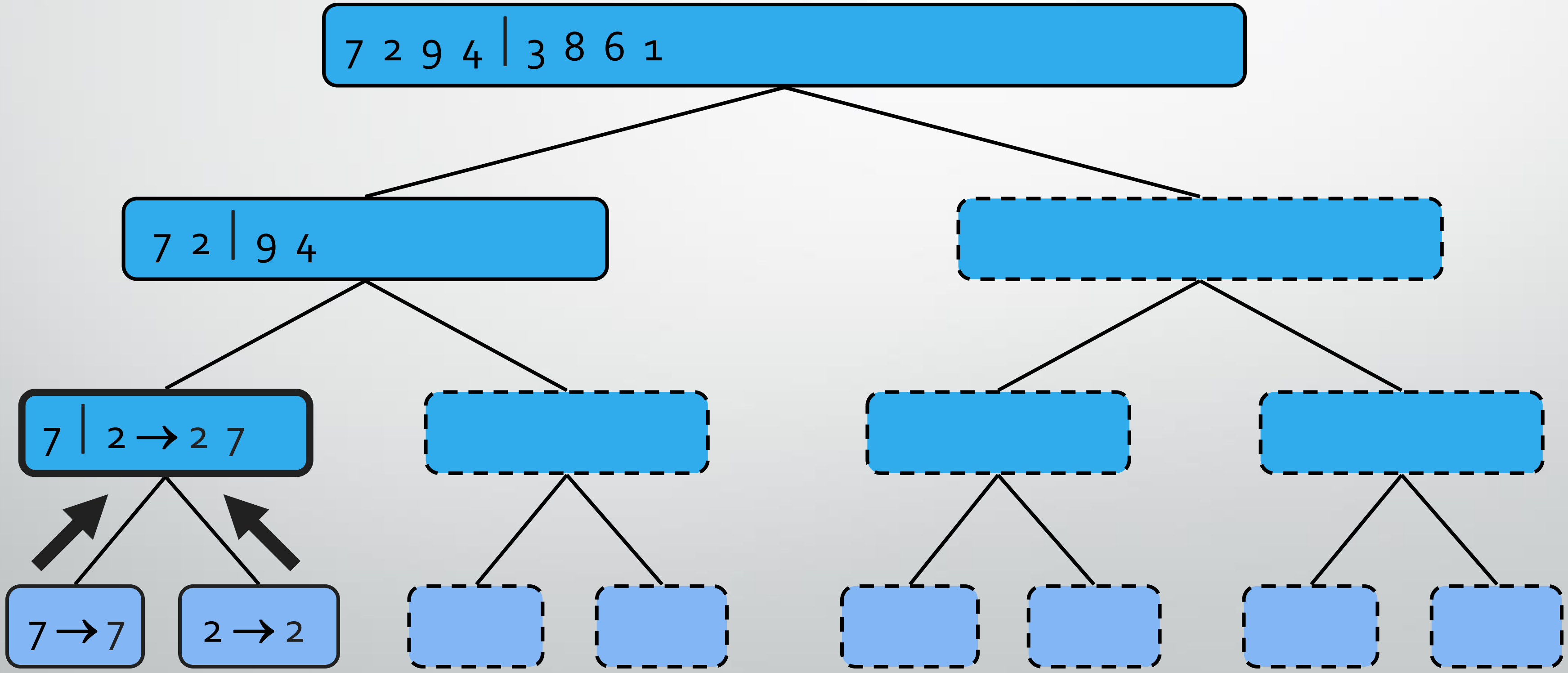
Execution Example

- Recursive call, base case



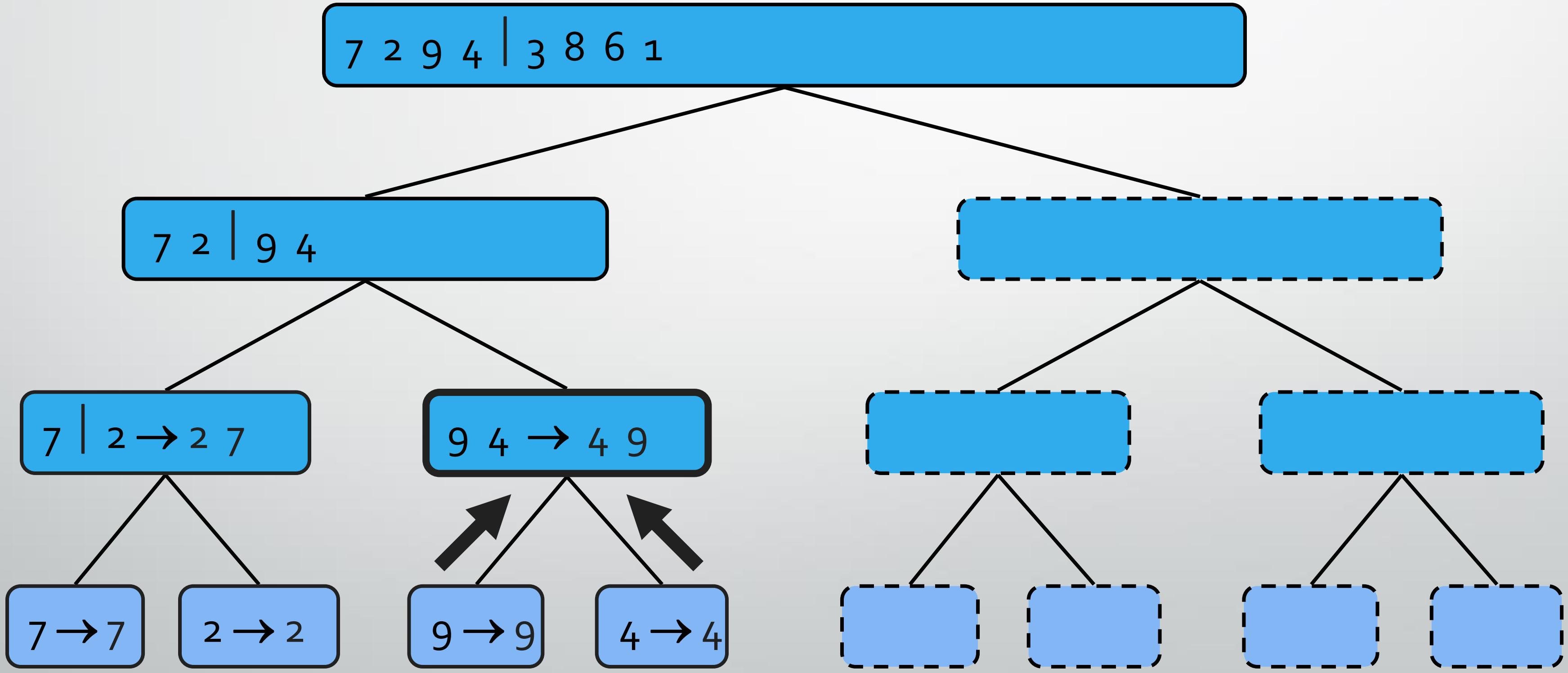
Execution Example

- Merge



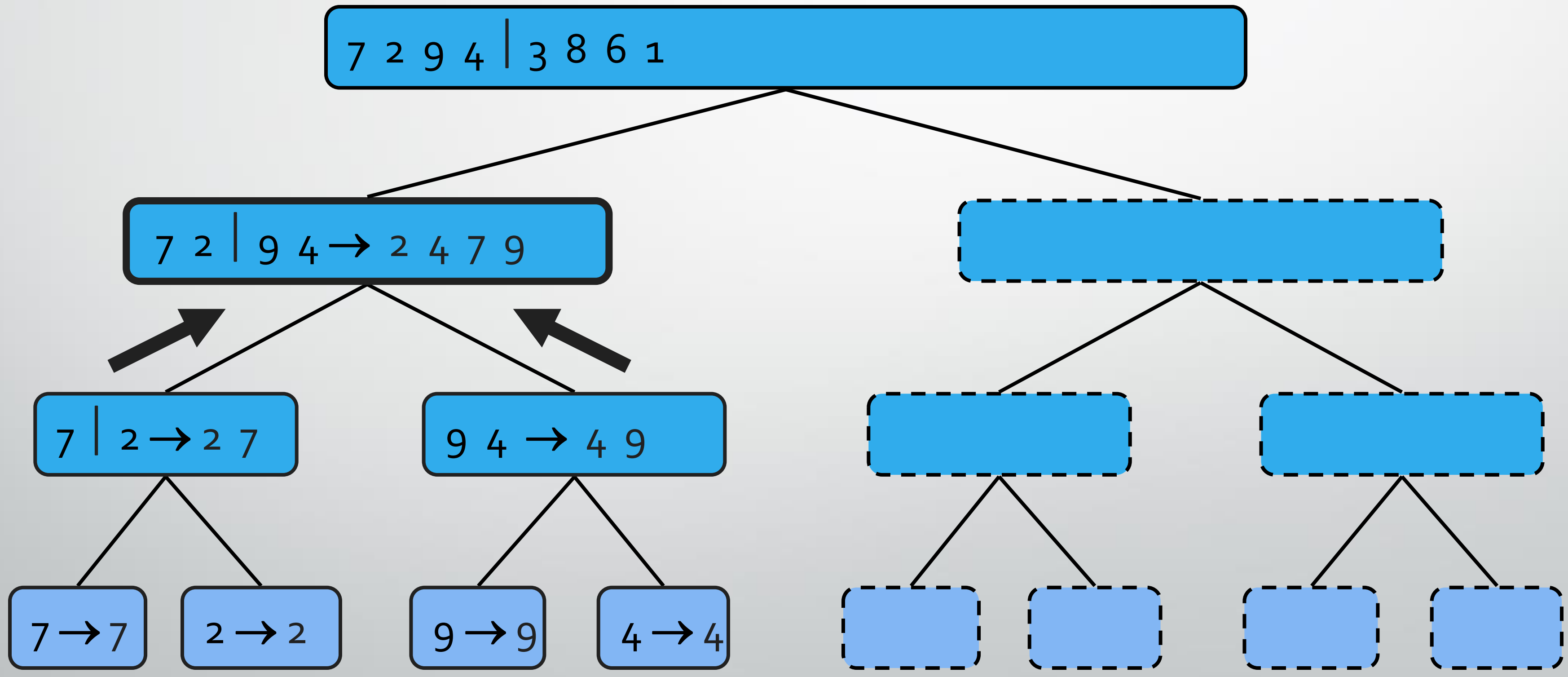
Execution Example

- Recursive call, …, base case, merge



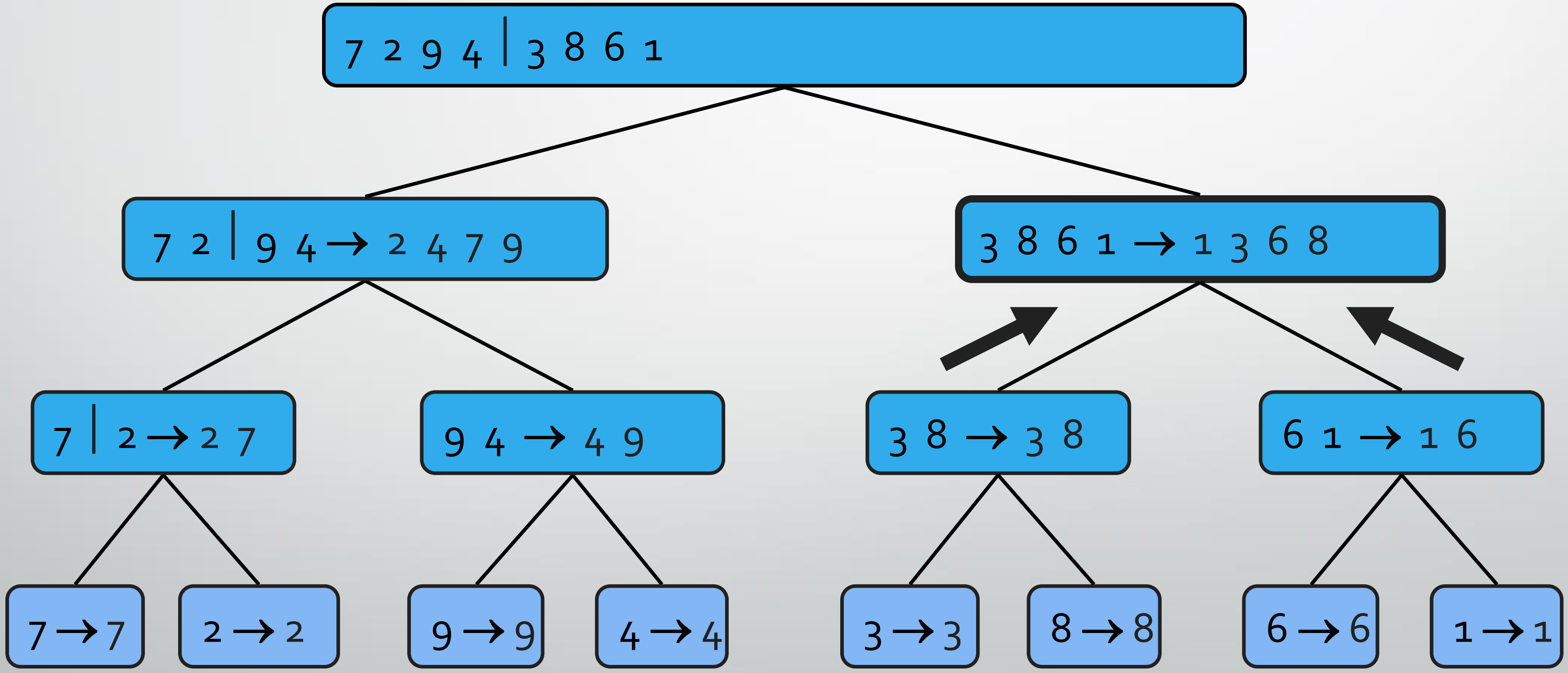
Execution Example (cont.)

- Merge



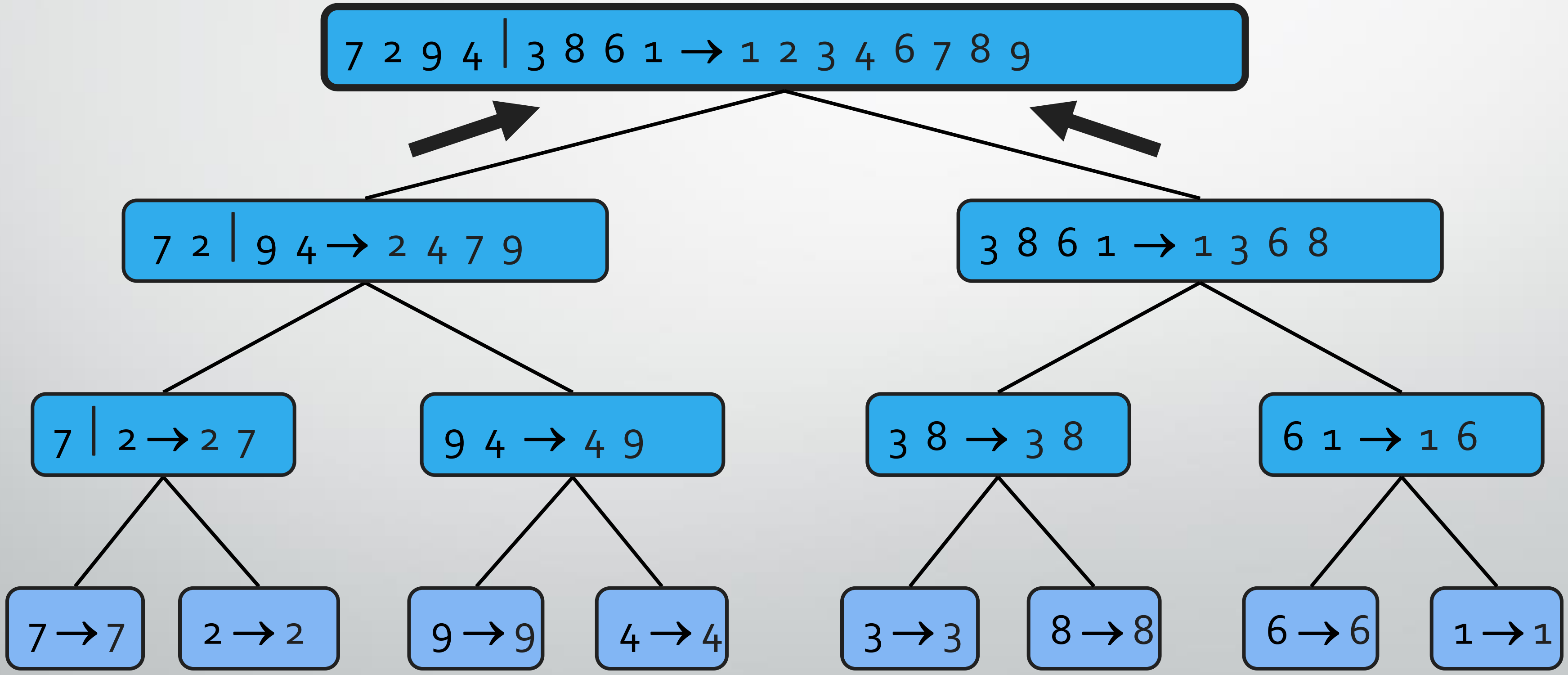
Execution Example

- Recursive call, …, merge, merge



Execution Example

- Merge



7 2 9 4 | 3 8 6 1 → 1 2 3 4 6 7 8 9

7 2 | 9 4 → 2 4 7 9

3 8 6 1 → 1 3 6 8

7 | 2 → 2 7

9 4 → 4 9

3 8 → 3 8

6 1 → 1 6

7 → 7

2 → 2

9 → 9

4 → 4

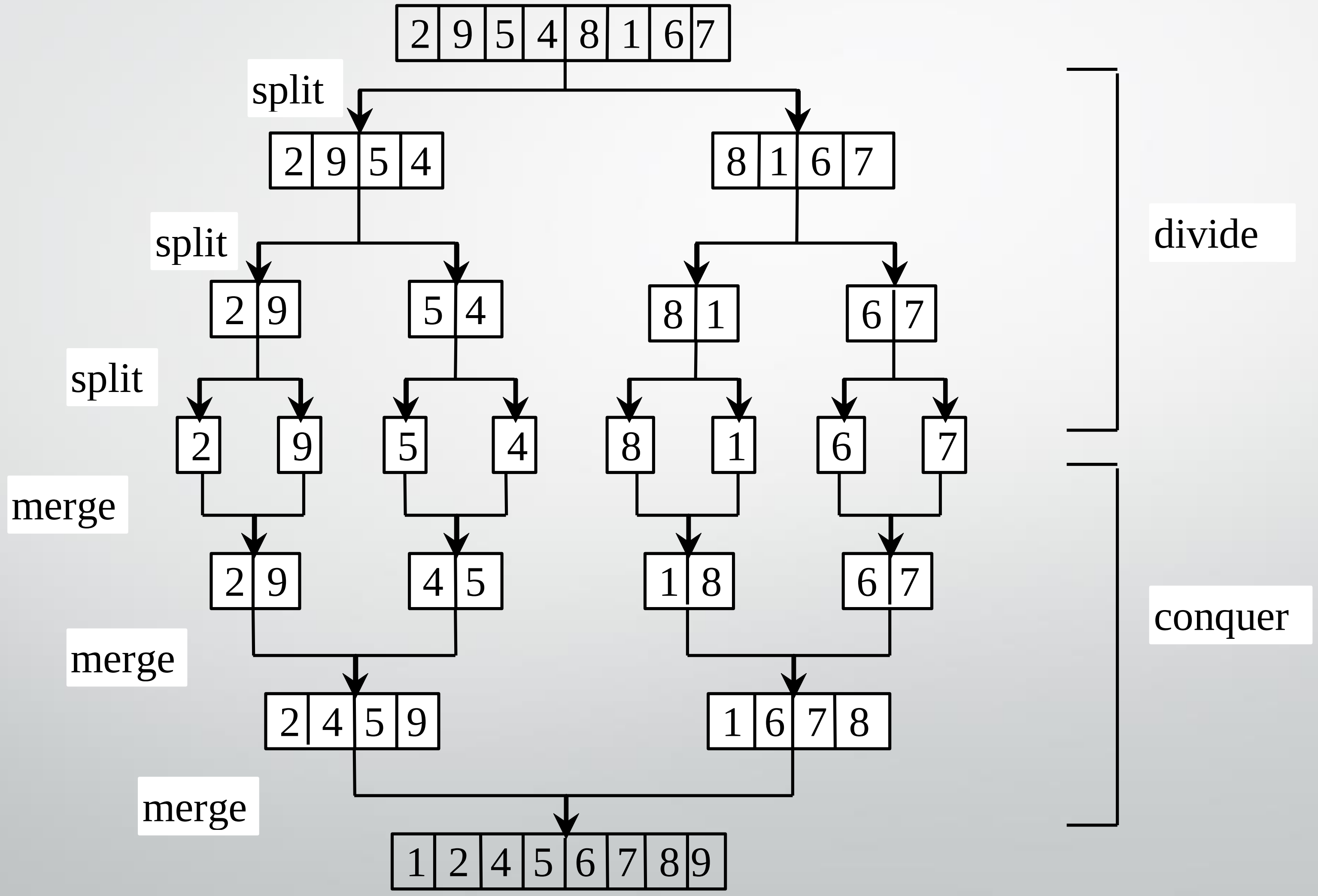
3 → 3

8 → 8

6 → 6

1 → 1

Merge Sort


2 9 5 4 8 1 6 7
split
2 9 5 4
8 1 6 7
split
2 9
5 4
8 1
6 7
split
2
9
5
4
8
1
6
7
merge
2 9
4 5
1 8
6 7
merge
2 4 5 9
1 6 7 8
merge
1 2 4 5 6 7 8 9
divide
conquer

Coding the Merge Sort

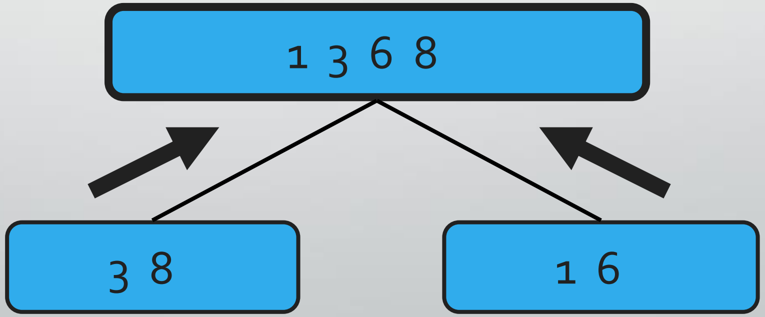
- We have several challenges here
  - Split the array between in two subarrays
  - Organise them / Merge them back together
  - Code this in a recursive way

Coding the Merge Sort

- Let’s start splitting the array into two sub arrays.
- How do we do it?
- Give it a go!

Coding the Merge Sort Algorithm

- Let’s now code the merging part / sorting part.
- We’ll be merging two arrays that are already sorted
- So when merging, we need to keep in mind that the new array must be sorted.



Merging Two Sorted Sequences

```
Algorithm merge(A, B)
    Input sequences A and B
    S = empty sequence
    while !A.isEmpty() & !B.isEmpty()
        if A.first().element() < B.first().element()
            S.addLast(A.first())
            A.remove(A.first())
        else
            S.addLast(B.remove(B.first()))
    while !A.isEmpty()
        S.addLast(A.first())
        A.remove(A.first())
    while !B.isEmpty()
        S.addLast(B.first())
        B.remove(B.first())
    Output  S sorted sequence of $A \cup B$
```

Merging two sorted sequences

- Give it a go!

Merge Sort Pseudocode

```
Algorithm mergeSort(S)
    Input sequence S with n elements
    if S.size() > 1
        [S_1, S_2] = partition(S, n/2)
        mergeSort(S_1)
        mergeSort(S_2)
        S = merge(S_1, S_2)
        Output sequence S sorted
    else
        Output sequence S
```

- Merge-sort on an input sequence ***S*** with ***n*** elements consists of three steps:
  1. Divide: partition ***S*** into two sequences S_1 and S_2 of about ***n***/2 elements each
  2. Recur: recursively sort S_1 and S_2
  3. Conquer: merge S_1 and S_2 into a unique sorted sequence

That’s all folks

- Any question?