# Session 52: Recursive Sorting

- Merge Sort
- Complexity of Merge Sort

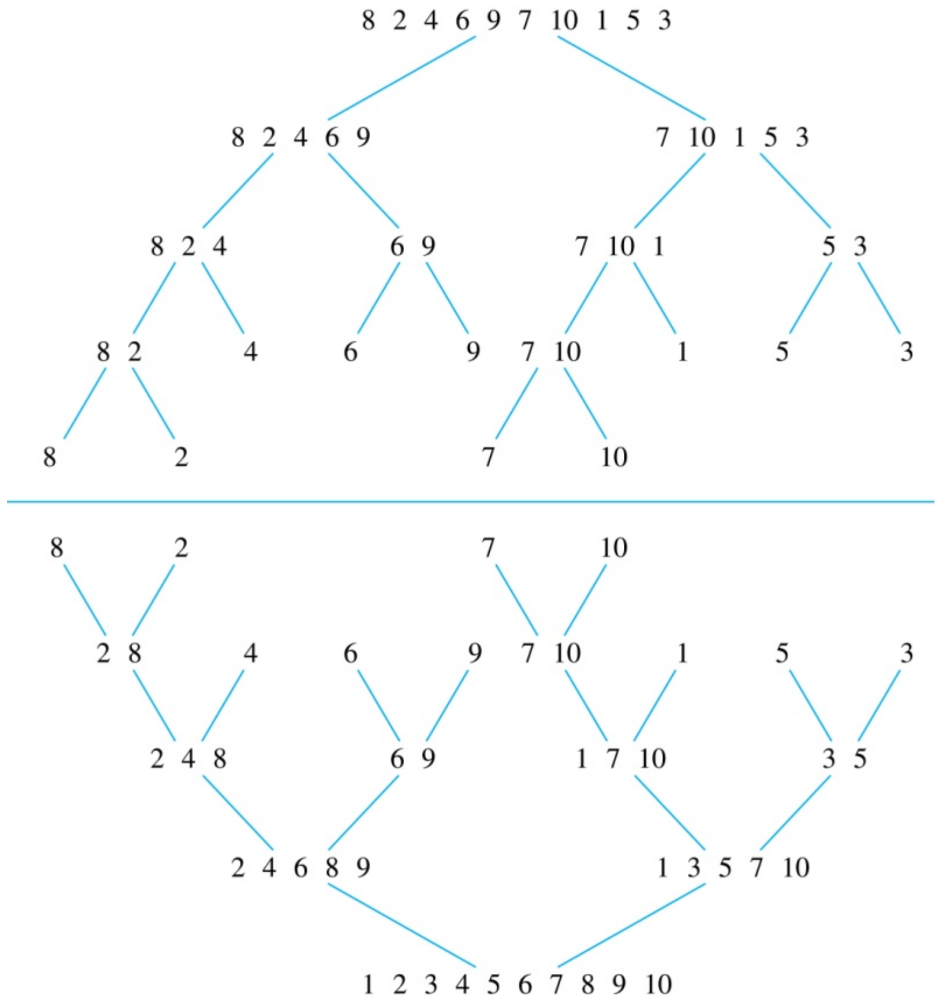
# Recursive Sorting

Sorting algorithms, like Bubble Sort, had complexity $\Theta(n^2)$

**Merge Sort** is a recursive sorting algorithm that performs significantly better

- Merge Sort works by iteratively splitting a list into two sublists of equal length until each sublist has one element.
- At each step a pair of sublists is successively merged into a list with the elements in increasing order. The process ends when all the sublists have been merged.

# Illustration Merge Sort



# Illustration Merging Two Lists

**Example**: merging the two sorted lists

Traverse the two lists in parallel from left to right

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 6 | 8 | 9 | 1 | 3 | 5 | 7 | 10 |

1 2 3 4 5 6 7 8 9 10

Always take the smaller element at the left of the two lists

# Algorithm for Merging Two Sorted lists

**procedure** *merge*($L_1$, $L_2$ : sorted lists)
$L$ := empty list
**while** $L_1$ and $L_2$ are both nonempty
remove smaller of first elements of $L_1$ and $L_2$ from its list;
put it at the end of $L$
**if** this removal makes one list empty
**then** remove all elements from the other list and append them to L
**return** $L$

**Complexity of Merge**: at most $|L_1| + |L_2| - 1$ comparisons

# Recursive Merge Sort

**procedure** *mergesort*($L = a_1, a_2, \ldots, a_n$ )

**if** $n > 1$ **then**

$m := \lfloor n/2 \rfloor$

$L_1 := a_1, a_2, \ldots, a_m$

$L_2 := a_{m+1}, a_{m+2}, \ldots, a_n$

$L := merge(mergesort(L_1), mergesort(L_2))$

When mergesort terminates $L$ is sorted into elements in increasing order

# Complexity of Merge Sort

For simplicity, assume that *n* is a power of 2, say $2^m$.

- At each invocation of procedure *mergesort*
  - The number of lists doubles and the length of the lists half
- After *m* invocations there are $n = 2^m$ lists of length 1.
- The merge procedure is now executed *m* times; at each execution
  - The length of lists doubles and their number halves
  - The cost of the merge procedure is the sum of the length of the lists minus 1
- The total cost is thus at most

$$(2^0 * (2^m - 1)) + (2^1 * (2^{m-1} - 1)) + \ldots + (2^{m-1} * (2^1 - 1)) = \sum_{k=1}^{m} \boxed{2^{k-1}}\boxed{(2^{m-k+1} - 1)}$$

Number of merges Cost of merge

# Complexity of Merge Sort

Using

$$\sum_{k=1}^{m} 2^{k-1} = 2^m - 1$$

we obtain

$$\sum_{k=1}^{m} 2^{k-1}(2^{m-k+1} - 1) =$$

Therefore the algorithm has complexity O(n log n), i.e. linearithmic complexity, which is the best complexity that can be obtained for sorting.

# Summary

- Merge Sort is a recursive sorting algorithm
- Complexity of Merge Sort is linearithmic