Sorting Techniques

- **Merge Sort**: it is also called external sorting technique because required extra memory to sort the elements.

Merge Sort

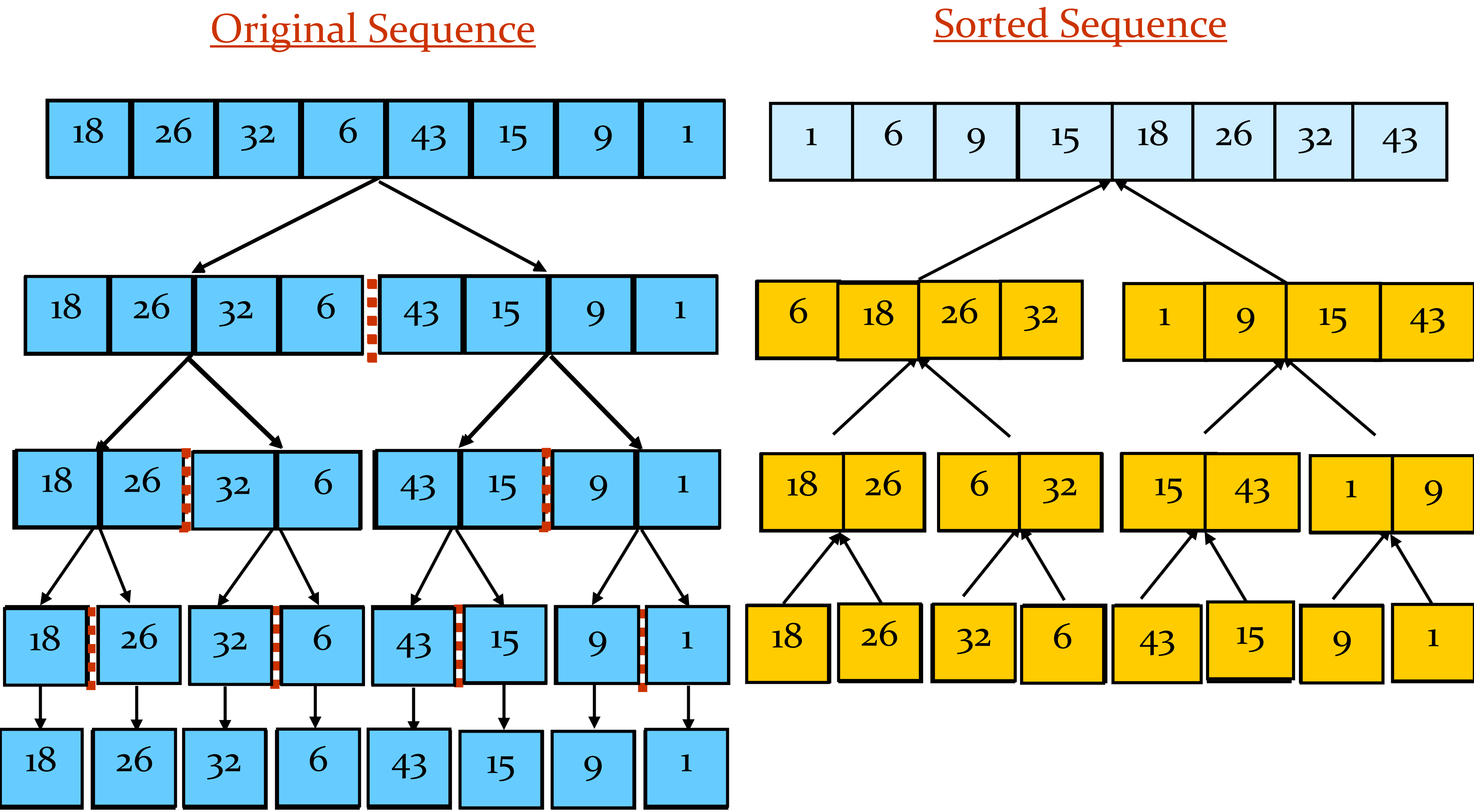
- Divide and Conquer
- Recursive in structure
  - ***Divide*** the problem into sub-problems that are similar to the original but smaller in size
  - ***Conquer*** the sub-problems by solving them recursively. If they are small enough, just solve them in a straightforward manner.
  - ***Combine*** the solutions to create a solution to the original problem

An Example: Merge Sort

Sorting Problem: Sort a sequence of n elements into non-decreasing order.

- ***Divide:*** Divide the n-element sequence to be sorted into two subsequences of $n/2$ elements each
- ***Conquer:*** Sort the two subsequences recursively using merge sort.
- ***Combine:*** Merge the two sorted subsequences to produce the sorted answer.

Merge Sort – Example



Merge-Sort (A, p, r)

INPUT: a sequence of *n* numbers stored in array A

OUTPUT: an ordered sequence of *n* numbers

***MergeSort* (*A*, *p*, *r*)** // sort $A[p..r]$ by divide & conquer

if $p < r$

  then $q \leftarrow \lfloor (p+r)/2 \rfloor$

    MergeSort (A, p, q)

    MergeSort (A, q+1, r)

    Merge (A, p, q, r) // merges $A[p..q]$ with $A[q+1..r]$

Initial Call: MergeSort(A, 1, n)

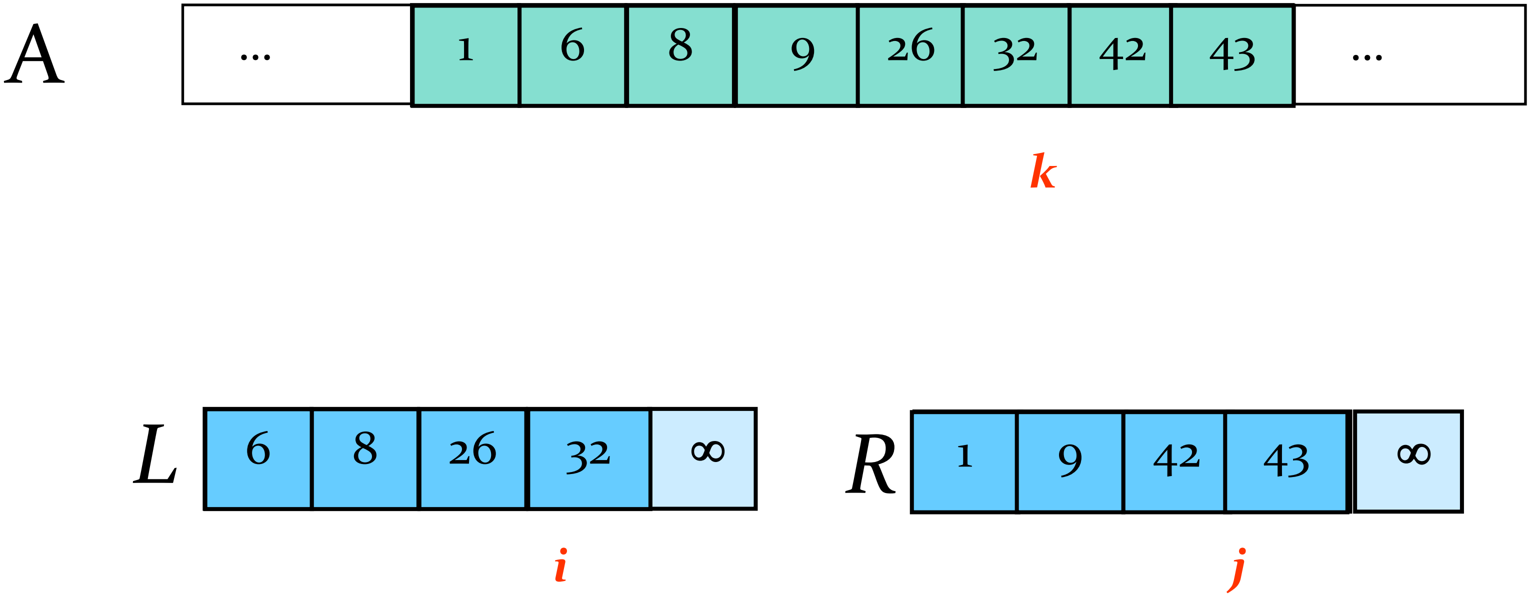
Procedure Merge

```
Merge(A, p, q, r)
n1 ← q − p + 1
n2 ← r − q
    for i ← 1 to n1
        do L[i] ← A[p + i − 1]
    for j ← 1 to n2
        do R[j] ← A[q + j]
    L[n1+1] ← ∞
    R[n2+1] ← ∞
    i ← 1
    j ← 1
    for k ← p to r
        do if L[i] ≤ R[j]
            then A[k] ← L[i]
                 i ← i + 1
            else A[k] ← R[j]
                 j ← j + 1
```

Input: Array containing sorted subarrays $A[p..q]$ and $A[q+1..r]$.

Output: Merged sorted subarray in $A[p..r]$.

Merge – Example



Analysis of Merge Sort

- Running time $T(n)$ of Merge Sort:
- Divide: computing the middle takes $\Theta(1)$
- Conquer: solving 2 sub-problems takes $2T(n/2)$
- Combine: merging n elements takes $\Theta(n)$
- Total:

$$T(n) = \Theta(1) \quad \text{if } n = 1$$
$$T(n) = 2T(n/2) + \Theta(n) \quad \text{if } n > 1$$

$$
\begin{aligned}
T(n) &= 2\ T(n/2) + n \\
&= 2\ ((n/2)\log(n/2) + (n/2)) + n \\
&= n\ (\log(n/2)) + 2n \\
&= n \log n - n + 2n \\
&= n \log n + n \\
&= O(n \log n\ )
\end{aligned}
$$

Comparing the Algorithms

	Best Case	**Average Case**	**Worst Case**
Bubble Sort	O(n)	O(n^2)	O(n^2)
Insertion Sort	O(n)	O(n^2)	O(n^2)
Selection Sort	O(n^2)	O(n^2)	O(n^2)
Merge Sort	O($n \log n$)	O($n \log n$)	O($n \log n$)
Quick Sort	O($n \log n$)	O($n \log n$)	O(n^2)
Heap Sort	O($n \log n$)	O($n \log n$)	O($n \log n$)

Thank You