# Divide-and-Conquer (D&C) Paradigm

Given a problem $\mathcal{A}$, if the D&C strategy is applicable, the problem is then solved using the strategy. The detailed steps are as follows.

1. **Divide** $\mathcal{A}$ into a number of subproblems with equal size roughly.

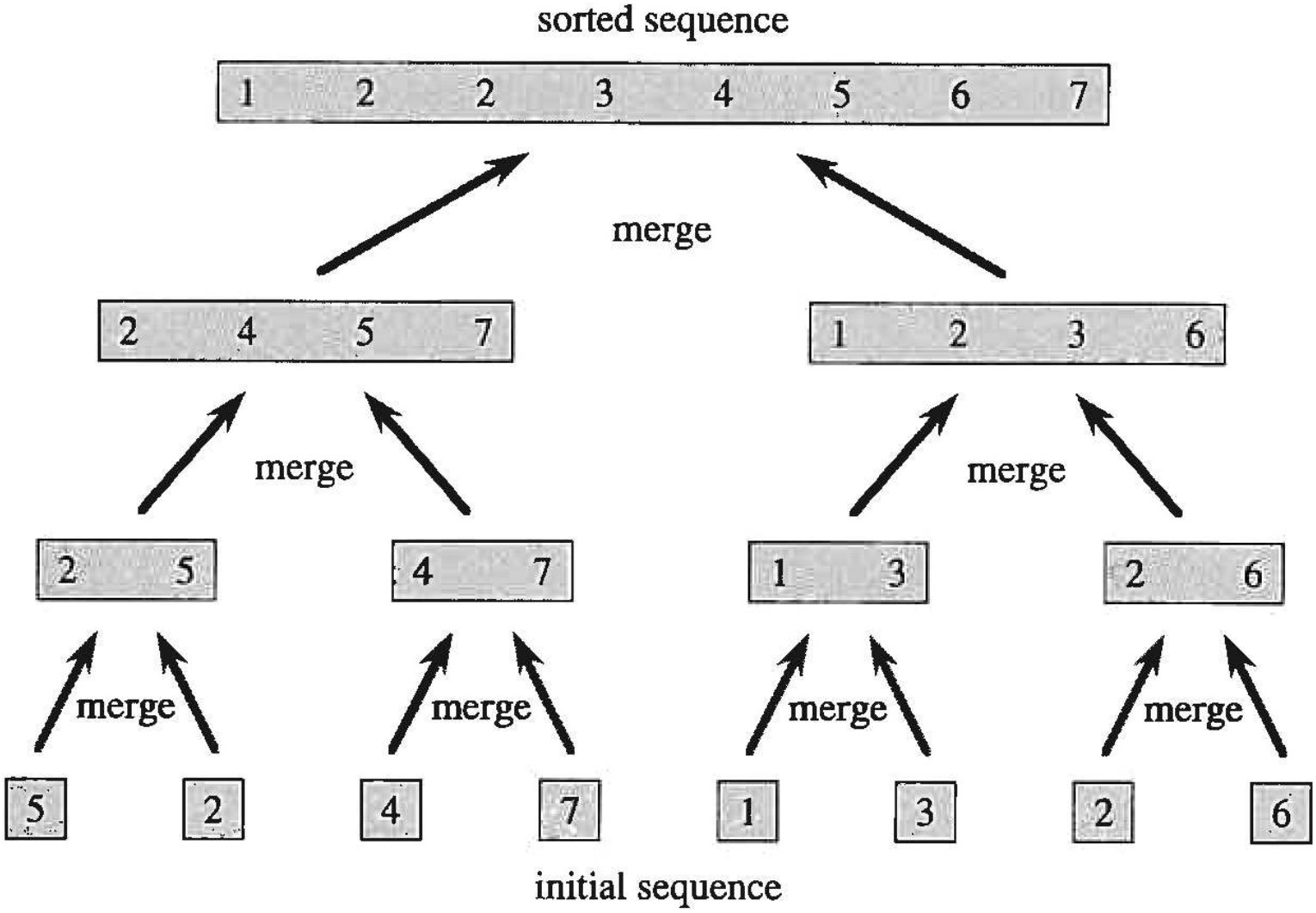
2. **Conquer** the subproblems by solving them recursively. For small enough subproblems, solve them directly without dividing them further.

3. **Combine** the solutions of the subproblems into the solution of the original problem.

# Divide-and-Conquer (D&C) Paradigm (cont.)

An Example of D&C Strategy: Consider the merge-sort algorithm in the very first slide of lecture 5.

- **Divide:** Divide the $n$-element sequence into two subsequences of $n/2$ elements each (or as close as possible if $n$ is odd)
- **Conquer:** Sort the two subsequences recursively, invoking the merge-sort algorithm
- **Combine:** Merge the two sorted subsequences into a sorted sequence


sorted sequence
1 2 2 3 4 5 6 7
merge
2 4 5 7
1 2 3 6
merge
merge
2 5
4 7
1 3
2 6
merge
merge
merge
merge
5
2
4
7
1
3
2
6
initial sequence

Cormen, p35

# Chapter 9. Medians and Order Statistics

**Question:** There are $n$ homogeneous servers in a data center. Let $a_i$ be the workload of server $i$, $1 \leq i \leq n$. Assuming that only comparison operations like $=, <, >$ are allowed, the questions are

- how many comparisons required to identify the heaviest loaded server
- how many comparisons required to identify the lightest loaded server
- how many comparisons required to find the median workload (used for balancing the load among the servers)

Notice that the communication cost among the servers is proportional to the number of messages exchanged between them, which is closely related to the number of comparisons performed.

# Chapter 9. Medians and Order Statistics

The $i$-th order statistic of a set of $n$ elements $A$ is the $i$-th smallest element, where $A = \{a_i \mid 1 \leq i \leq n\}$. E.g.,

- The *minimum* of a set of $n$ elements is the 1st order statistic
- The *maximum* of a set of $n$ elements is the $n$-th order statistic
- A *median*, informally, is the middle-valued element of the set. About half the elements are larger than the median and about half are smaller.
  - When $n$ is odd, the median is the $i$-th order statistic, where $i = (n+1)/2$.

    $a'_1, a'_2, \ldots, a'_{\frac{n+1}{2}}, \ldots, a'_n$, where $a'_i \leq a'_j$ if $i < j$.
  - When $n$ is even, there are two medians, which are the $i$-th and $(i+1)$-th order statistics, where $i = n/2$.

    $a'_1, a'_2, \ldots, a'_{\frac{n}{2}}, a'_{\frac{n}{2}+1}, \ldots, a'_n$, where $a'_i \leq a'_j$ if $i < j$

# 9.1 Finding the Minimum and Maximum Elements Simultaneously

Given a set $A$ of $n$ elements $A = \{a_1, a_2, \ldots, a_n\}$, what is the minimum number of comparisons required to find both the maximum and minimum elements in $A$?

It is easy to achieve that with $2n - 3$ comparisons:

- Use $n - 1$ comparisons in the original sequence to find the minimum element
- Use $n - 2$ comparisons in the rest of the sequence to find the maximum element

Is the best possible that we can do?
Can we do better? (Yes or No)

# 9.1 Finding the Minimum and Maximum Elements (continued)

If $n$ is even, a simple algorithm is proposed as follows.

- Divide the elements into $n/2$ pairs, and compare each pair ($n/2$ comparisons).
- Find the minimum of the $n/2$ smallest elements ($n/2-1$ comparisons).
- Find the maximum of the $n/2$ largest elements ($n/2-1$ comparisons).

In total, $3n/2-2$ comparisons.

If $n$ is odd, we can find the maximum and minimum values of the first $n-1$ elements $a_1, a_2, \ldots, a_{n-1}$, and then compare them with the last element $a_n$.

It thus takes $3(n-1)/2-2+2 = 3(n-1)/2$.

In general, including odd $n$, no more than $3\lfloor n/2 \rfloor$ comparisons. It can be proved that this is the best possible, based on the comparison model.

# 9.1 Finding the Minimum and Maximum Elements (continued)

**An example:**

Consider a set $\{3,7,5,4,8,2,9,6,1\}$ of 9 elements.

- Divide the first 8 elements into 4 pairs, and sort each pair $\{(3,7),(4,5),(2,8),(6,9)\}$ (4 comparisons).
- Find the minimum of the 4 smallest elements $\{3,4,2,6\}$, 2, (3 comparisons).
- compare the smallest element 2 with the last element 1, the minimum element is 1.
- Find the maximum of the 4 largest elements $\{7,5,8,9\}$, 9, (3 comparisons).
- compare the largest element 9 with the last element 1, the maximum element is 9.

The total number of comparisons is $\frac{n-1}{2}+2(\frac{n-1}{2}-1)+2 = 3\frac{n-1}{2} = 3\lfloor n/2 \rfloor$.

**Exercise:** What is the minimum number comparisons needed to find the median of a sequence of $n$ elements?

Given a set $A$ of $n$ (distinct) elements and an integer $i$, with $1 \leq i \leq |A|$, the selection problem is to find an element $x \in A$ that is larger than exactly $(i-1)$ other elements in the set, assuming that $n = |A|$. Notice that the median of set $A$ is the $\lfloor |A|/2 \rfloor$-order statistic of $A$.

# 9.2 The Selection Problem

The selection problem: Given a set of $n$ elements, find the $i$th smallest element from the set, where $1 \leq i \leq n$.

Several variants of the selection problem are as follows.

1. Find the $i$-th largest element in $A$
   = find the $(n-i+1)$-th smallest element in $A$

2. Find the first $i$ smallest elements in $A$
   find the $i$-th smallest element $x$ in $A$, followed by scanning the sequence and picking any element $y \leq x$ when the number of picked elements is no larger than $i$

3. Find the first $i$ largest elements in $A$

# An algorithm for general selection

Suppose we have an array $A[1..n]$ and we want to find the $i$-th smallest element.

1. (PIVOT ELEMENT)
   Choose an element $x$ from $A[1..n]$, where $x$ is referred to as the **the pivot element** of the sequence.

2. (PARTITION)
   Rearrange $A[1..n]$ so that for $\exists j$,
   $A_{1,1} = \{A[1], \ldots, A[j-1] \mid A[i] < x\}$, $A[j] = x$, and
   $A_{1,2} = \{A[j+1], \ldots, A[n] \mid A[i'] > x\}$.

3. (RECURSE)
   **If** $j = i$, then $x$ is the answer (WHY?).
   **Else** (**If** $i > j$, then find **the** $(i-j)$**-th smallest element** in $A[j+1..n]$).
   **Else** find the $i$-th smallest element in $A[1..j-1]$.

# An algorithm for general selection (continued)

- The partitioning step can be done in $O(n)$ time.
- The overall performance of the algorithm depends on the choice of the pivot element $x$.
  In the worst case, the total time might be $\Theta(n^2)$. (Consider if $i = 1$ and $x$ is always the largest element.)
- It can be shown that if $x$ is chosen at random, then the average time is $O(n)$.