Overview

- Aims to familiarize us with framework used throughout text
- Examines alternate solutions to the sorting problem presented in Ch. 1
- Specify algorithms to solve problem
- Argue for their correctness
- Analyze running time, introducing notation for asymptotic behavior
- Introduce divide-and-conquer algorithm technique
The Sorting Problem

**Input:** A sequence of \( n \) numbers \([a_1, a_2, \ldots, a_n]\).

**Output:** A permutation or reordering \([a'_1, a'_2, \ldots, a'_n]\) of the input sequence such that \( a'_1 \leq a'_2 \leq \ldots \leq a'_n \).

An instance of the Sorting Problem:

**Input:** A sequence of 6 numbers \([31, 41, 59, 26, 41, 58]\).

**Expected output for given instance:**

**Expected Output:** The permutation of the input \([26, 31, 41, 41, 58, 59]\).

Insertion Sort

The main idea ...
Insertion Sort (cont.)

(a) 1 2 3 4 5 6
    5 2 4 6 1 3

(b) 1 2 3 4 5 6
    2 5 4 6 1 3

(c) 1 2 3 4 5 6
    2 4 5 6 1 3

(d) 1 2 3 4 5 6
    2 4 5 6 1 3

(e) 1 2 3 4 5 6
    1 2 4 5 6 3

(f) 1 2 3 4 5 6
    1 2 3 4 5 6

3
Insertion Sort (cont.)

The algorithm ...

\begin{algorithm}
\caption{Insertion-Sort($A$)}
\begin{algorithmic}[1]
\For{$j \leftarrow 2$ to $\text{length}[A]$}
    \State $key \leftarrow A[j]$
    \State $i \leftarrow j - 1$
    \While{$i > 0$ and $A[i] > key$}
        \State $A[i+1] \leftarrow A[i]$
        \State $i \leftarrow i - 1$
    \EndWhile
    \State $A[i+1] \leftarrow key$
\EndFor
\end{algorithmic}
\end{algorithm}

Loop Invariant

- Property of $A[1 \ldots j - 1]$
  - At the start of each iteration of the for loop of lines 1–8, the subarray $A[1 \ldots j - 1]$ consists of the elements originally in $A[1 \ldots j - 1]$ but in sorted order.

- Need to establish the following re invariant:
  - **Initialization**: true prior to first iteration
  - **Maintenance**: if true before iteration, remains true after iteration
  - **Termination**: at loop termination, invariant implies correctness of algorithm
Analyzing Algorithms

- Has come to mean predicting the resources that the algorithm requires
- Usually computational time is resource of primary importance
- Aims to identify best choice among several alternate algorithms
- Requires an agreed-upon “model” of computation
- Shall use a generic, one-processor, random-access machine (RAM) model of computation

Random-Access Machine

- Instructions are executed one after another (no concurrency)
- Admits commonly found instructions in “real” computers, data movement operations, control mechanism
- Uses common data types (integer and float)
- Other properties discussed as needed
- Care must be taken since model of computation has great implications on resulting analysis
Analysis of Insertion Sort

- Time resource requirement depends on **input size**
- **Input size** depends on problem being studied; frequently, this is the number of items in the input
- Running time: number of primitive operations or “steps” executed for an input
- Assume constant amount of time for each line of pseudocode

---

Analysis of Insertion Sort

Time efficiency analysis ...

```
INSERTION-SORT(A)
1   for j ← 2 to length[A]
2     do key ← A[j]
3       ▷ Insert A[j] into the sorted sequence A[1..j−1].
4       i ← j − 1
5       while i > 0 and A[i] > key
6           do A[i + 1] ← A[i]
7           i ← i − 1
8       A[i + 1] ← key
```

| \( n \) | \( n-1 \) | \( n-1 \) | \( n-1 \) | \( t_j \) | \( t_j - 1 \) | \( n-1 \) | \( n-1 \) | \( n-1 \) | \( n-1 \) |