Design & Analysis of Algorithms

Algorithm: Well defined computational procedure that takes some inputs & returns some outputs.

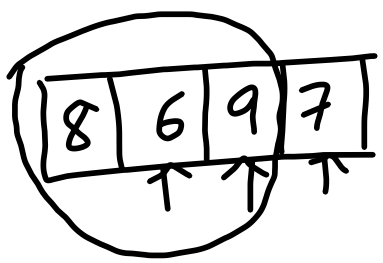
Objective: Solve computational problems efficiently.

Example: Sorting

I/P: $\langle a_1, a_2 \ldots, a_n \rangle$

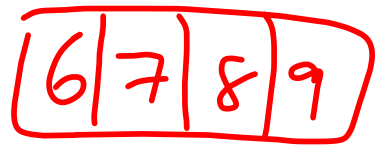
O/P: $\langle a_1', a_2', \ldots, a_n' \rangle \Rightarrow \boxed{a_1' \leq a_2' \leq \ldots \leq a_n'}$

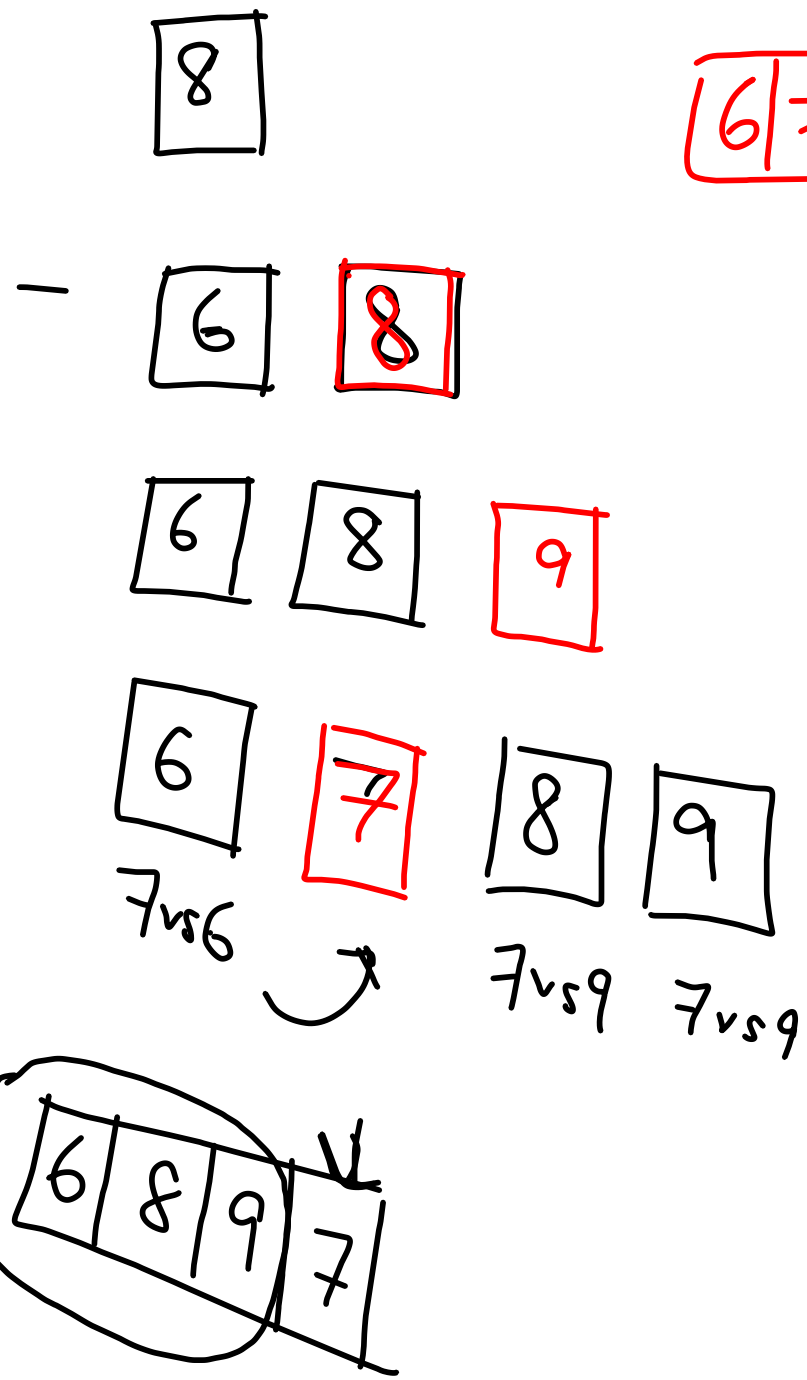
Correctness:
- halts after finite time.
- Correct output.



Insertion Sort

$|A| = n$
↳ size of the array

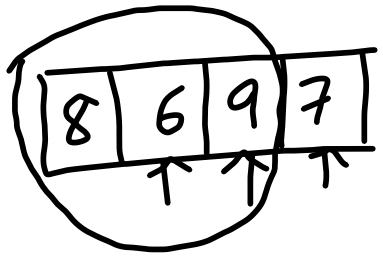




Insertion Sort (A)

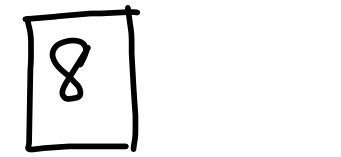
For $j = 2$ (1) n

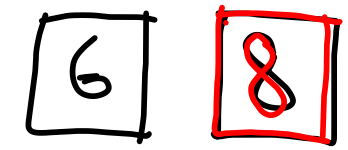
- Key = A[j]
- i = j - 1
- while (key < A[i] && i ≥ 1)
  - A[i+1] = A[i]
  - i = i - 1
- A[i+1] = key

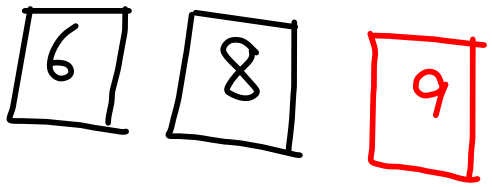


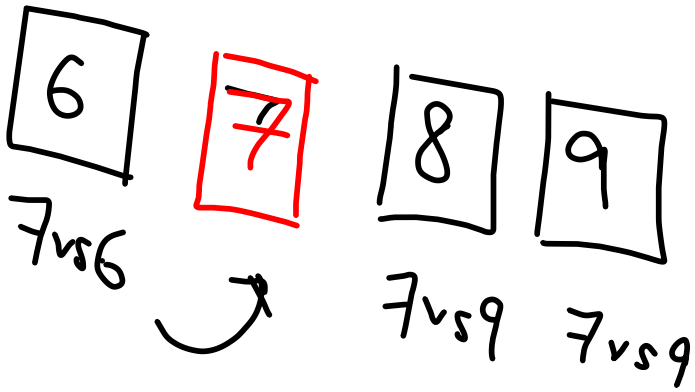
Insertion Sort

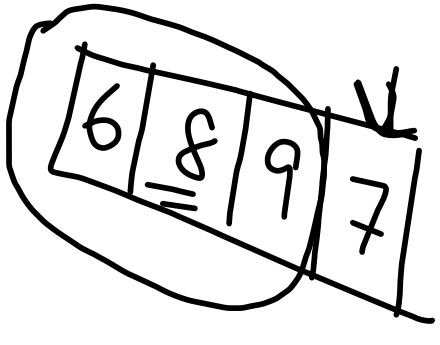
$|A| = n$
↳ size of the array

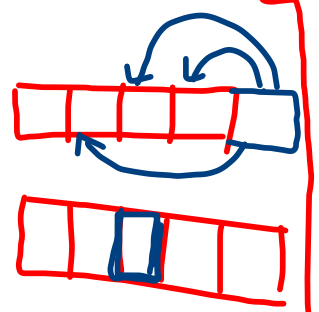












```
Insertion Sort (A)
n      For j = 2 (1) n
(n-1)      Key = A[j]
(n-1)      i = j - 1
Σ_{j=2}^{n} t_j      while (key < A[i] && i ≥ 1)
Σ_{j=2}^{n} (t_j - 1)      |  A[i+1] = A[i]
Σ_{j=2}^{n} (t_j - 1)      |  i = i - 1
(n-1)      A[i+1] = key
```

Correctness

$A[a..b]$

↳ $A[a]\ A[a+1] \cdots A[b]$

Loop Invariant

j^{th} Step: - The elements of $A[1..j-1]$ are same as i/p.
- $A[1..j-1]$ is sorted

- Initialization
- Maintenance
- Termination

} Induction

→ terminate from loop
the property gives you correctness.

} Terminate at $j = n+1$
$\Rightarrow A[1..n]$ Sorted

Analysis

Time → size of the input.

- RAM (Random Access Machine) Model
  - Simple Instructions (ADD, SUBTRACT, MLTPY, LOAD, STORE, COPY, BRANCH)
  - Sequential (no concurrent execution)
  - Each instruction caries some cost (constant)
- Consider Pseudocode, assign some cost for each line.

Analysis

Time → size of the input.

- RAM (Random Access Machine) Model
  - Simple Instructions (ADD, SUBTRACT, MLTPY, LOAD, STORE, COPY, BRANCH)
  - Sequential (no concurrent execution)
  - Each instruction caries some cost (constant)
- Consider Pseudocode, assign some cost for each line.

Time Complexity

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1) + c_3 \cdot (n-1) + c_4 \cdot \sum_{n=2}^{n} t_j + c_5 \cdot \sum_{n=2}^{n} (t_j - 1) + c_6 \cdot \sum_{n=2}^{n} (t_j - 1) + c_7 \cdot (n-1)$$

Best Case :

Worst Case :

Time Complexity

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1) + c_3 \cdot (n-1) + c_4 \cdot \sum_{j=2}^{n} t_j + c_5 \cdot \sum_{j=2}^{n} (t_j - 1) + c_6 \cdot \sum_{j=2}^{n} (t_j - 1) + c_7 \cdot (n-1)$$

Best Case:

Array already Sorted. $t_j = 1$

$\simeq c \cdot n + d$

Worst Case:

Array sorted in reverse order $t_j = j$

$\simeq c \cdot n^2 + d \cdot n + e$

Simplification

- Major term (only)
- Ignore the Constant.

- Worst Case Analysis (*)
- Average Case Analysis

Ex

Selection Sort

- At step i, find the minimum among $A[i, \ldots, n]$
- Let $min = A[k]$
- Swap $(A[i], A[k])$

- Pseudocode
- Loop Invariant – Correctness
- Time Complexity