Data Structures and Algorithm Analysis

13

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

Insertion sort

Insertion sort

- A good algorithm for sorting a small number of elements
- It works the way you might sort a hand of playing cards
  - Start with an empty left hand and the cards face down on the table
  - Then remove one card at a time from the table, and insert it into the correct position in the left hand.
  - To find the correct position for a card, compare it with each of the cards already in the hand, from right to left.

- At all times, the cards held in the left hand are sorted, and these cards were originally the top cards of the pile on the table

- Divide array into two lists <u>logically</u>
- One List contains 1 element
  - 1 element is always sorted
- Insert other elements into that at its correct position in that list.

Insertion Sort

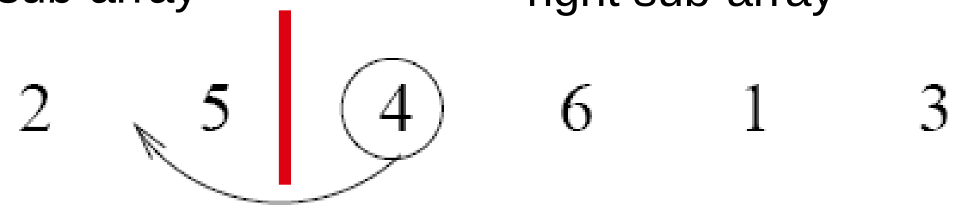
input array

5 2 4 6 1 3

at each iteration, the array is divided in two sub-arrays:

left sub-array

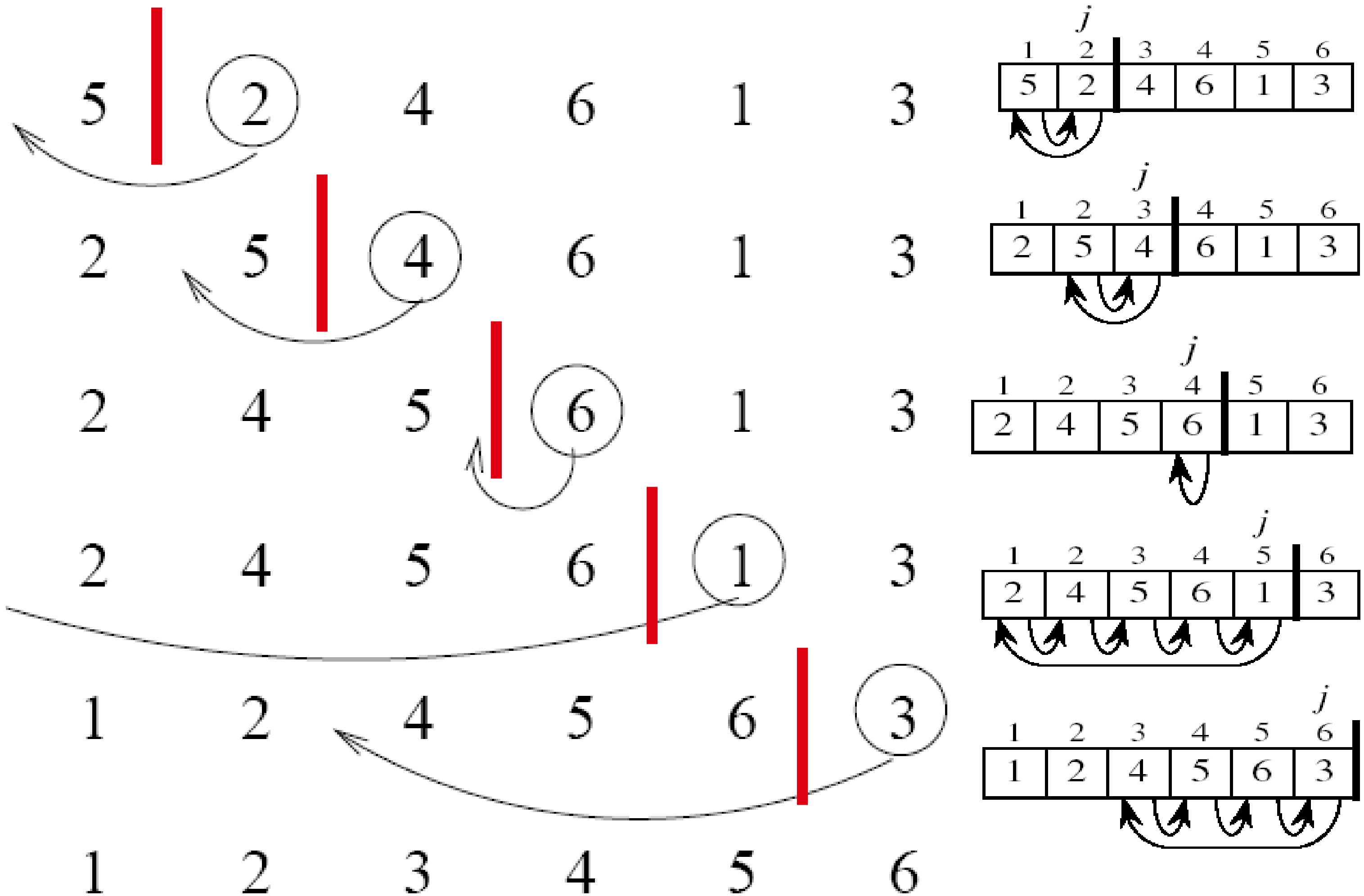
right sub-array



sorted

unsorted

Insertion Sort



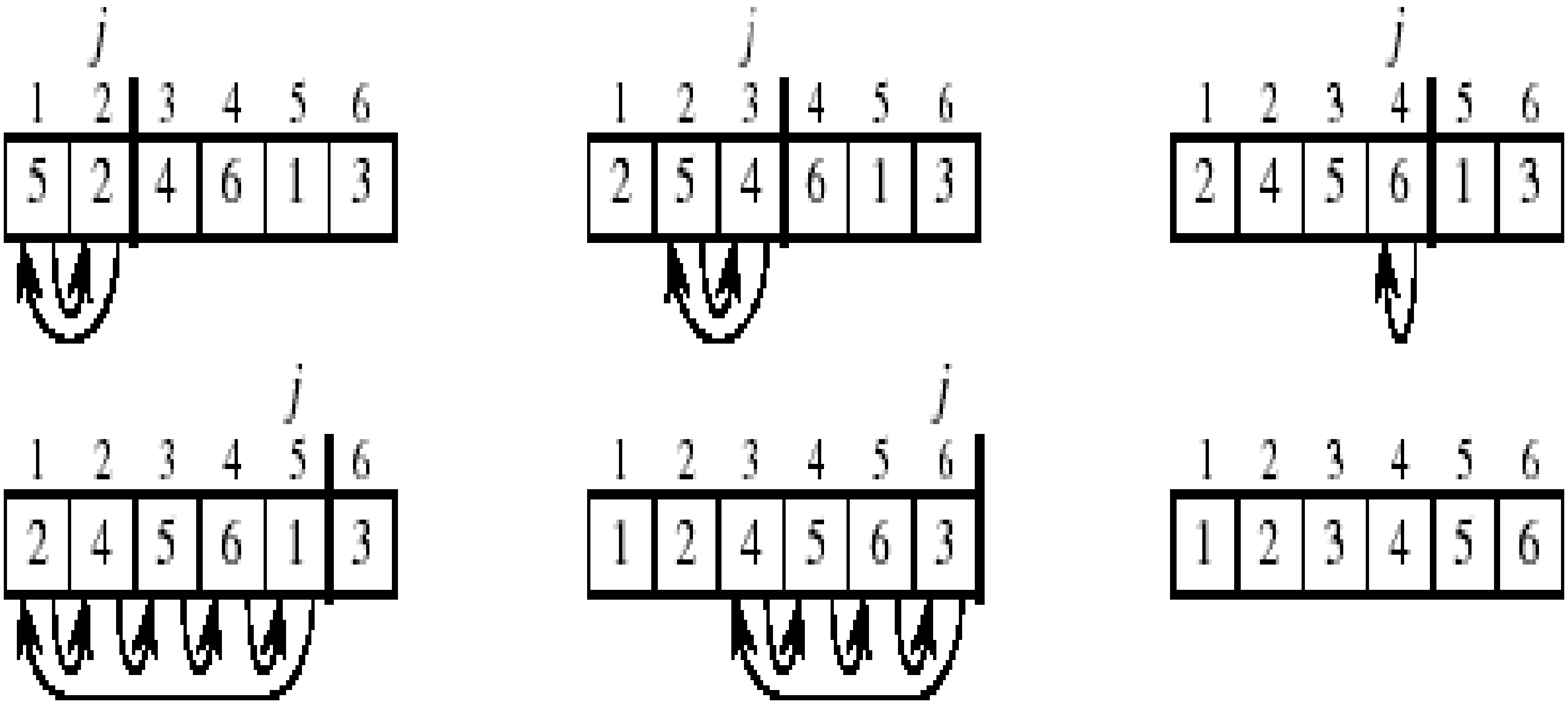
Insertion Sort Algorithm

Assume Array index starts from 1 to n

INSERTION-SORT(A)
for $j \leftarrow 2$ **to** n
  do $key \leftarrow A[j]$

    $i \leftarrow j - 1$
    while $i > 0$ and $A[i] > key$
      do $A[i+1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    $A[i+1] \leftarrow key$

Example:



j
1 2 3 4 5 6
5 2 4 6 1 3
j
1 2 3 4 5 6
2 5 4 6 1 3
j
1 2 3 4 5 6
2 4 5 6 1 3
j
1 2 3 4 5 6
2 4 5 6 1 3
j
1 2 3 4 5 6
1 2 4 5 6 3
1 2 3 4 5 6
1 2 3 4 5 6

Insertion sort - Example

WHILE (i > 0) AND (A[i] > key)

FOR j:=2,n

j	i				
j=2	key= 5	i=1	i=0		
	8 5 9 8 1	8 8 9 8 1	5 8 9 8 1		
j=3	key = 9	i=2			
	5 8 9 8 1	5 8 9 8 1			
j=4	key = 8	i=3	i=2		
	5 8 9 8 1	5 8 9 9 1	5 8 8 9 1		
j=5	key = 1	i=4	i=3	i=2	i=1
	5 8 8 9 1	5 8 8 9 9	5 8 8 8 9	5 8 8 8 9	5 5 8 8 9
					i=0
					1 5 8 8 9

Analysis of Insertion Sort

```
INSERTION-SORT(A)
for j ← 2 to n
    do key ← A[j]

       i ← j − 1
       while i > 0 and A[i] > key
           do A[i + 1] ← A[i]
              i ← i − 1
       A[i + 1] ← key
```

For each ***j*** **=** 2,3.. *n,* where *n* = *length[A],* we let t_j be the number of times the While loop test in line 5 is executed for that value of ***j***.

Analysis of Best case

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ to n	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	n - 1
do $A[i+1] \leftarrow A[i]$	c_6	0
$i \leftarrow i-1$	c_7	0
$A[i+1] \leftarrow key$	c_8	$n-1$

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\
&= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)\,.
\end{aligned}
$$

Best case: The array is already sorted.

- Always find that $A[i] \le key$ upon the first time the **while** loop test is run (when $i = j - 1$).
- All t_j are 1.
- Running time is

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$
$$= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) .$$

- Can express $T(n)$ as $an + b$ for constants a and b (that depend on the statement costs c_i) $\Rightarrow T(n)$ is a *linear function* of n.

Analysis of Worst Case

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** n	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

Worst Case

- *If the array is in reverse sorted order-that is, in decreasing order-the worst case happens.*
- *We must compare each element A[j] with each element in the entire sorted subarray A[l . .j - 1), and so tj= j for j = 2,3,..., n.*

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

- Running time is

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\
&\quad - (c_2 + c_4 + c_5 + c_8)\ .
\end{aligned}
$$

- Can express $T(n)$ as $an^2 + bn + c$ for constants a, b, c (that again depend on statement costs) $\Rightarrow T(n)$ is a *quadratic function* of n.