Lecture 07

CSE 1201 : Data Structure

Insertion Sort

Md. Shahid Uz Zaman
Dept. of CSE, RUET

Lecture 02 : Insertion Sort

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 number [31, 41, 59, 26, 41, 58].

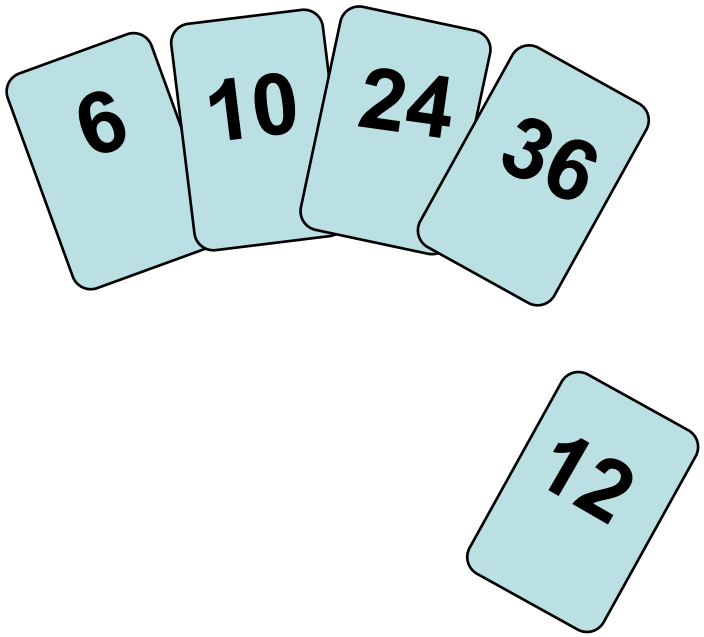
Expected output for given instance:

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

Insertion Sort

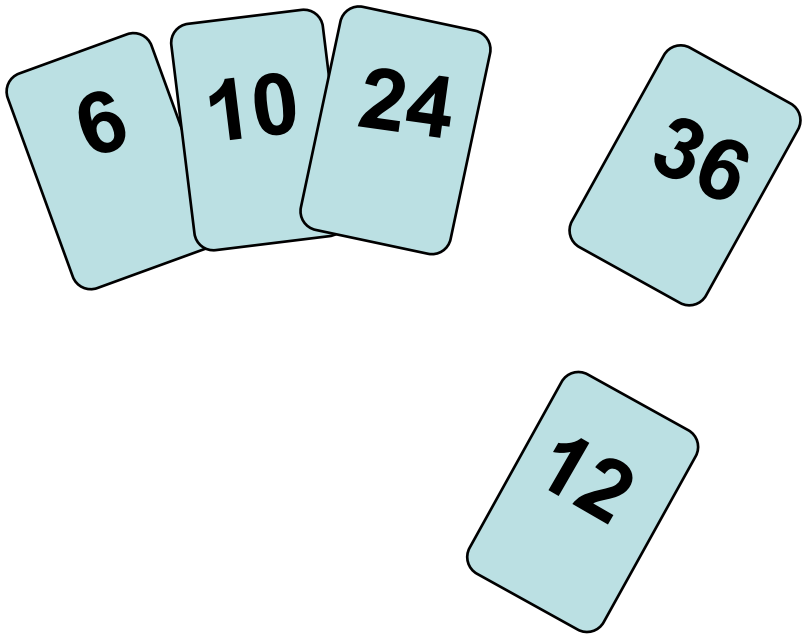
- Idea: like sorting a hand of playing cards
  - Start with an empty left hand and the cards facing down on the table.
  - Remove one card at a time from the table, and insert it into the correct position in the left hand
    - compare it with each of the cards already in the hand, from right to left
  - The cards held in the left hand are sorted
    - these cards were originally the top cards of the pile on the table

Insertion Sort

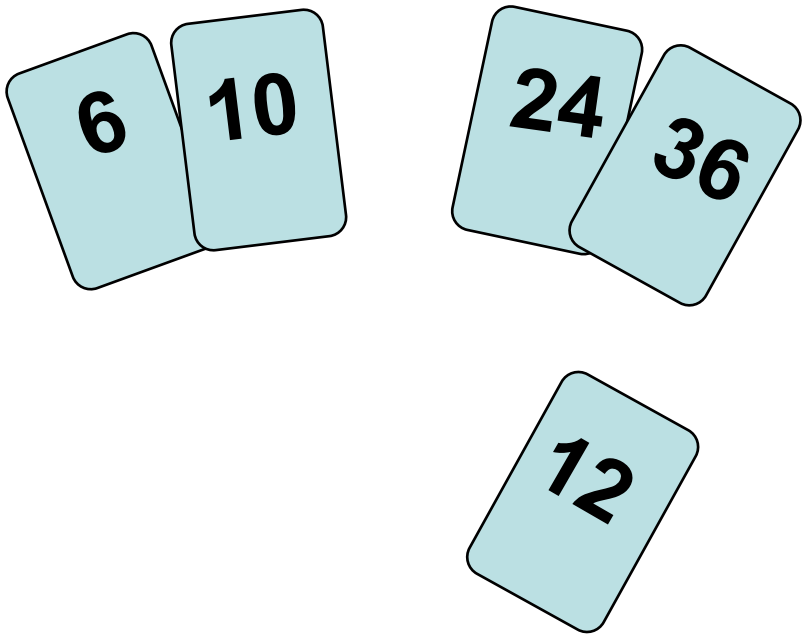


To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion Sort



Insertion Sort



Insertion Sort

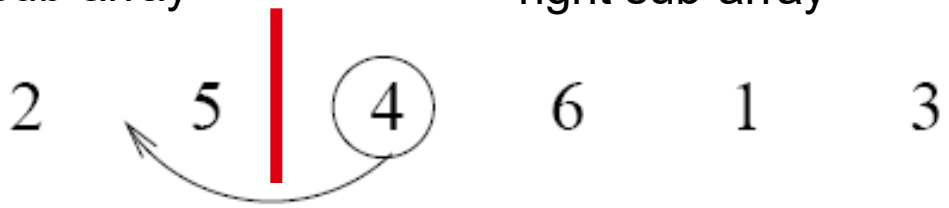
input array

5 2 4 6 1 3

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

left sub-array

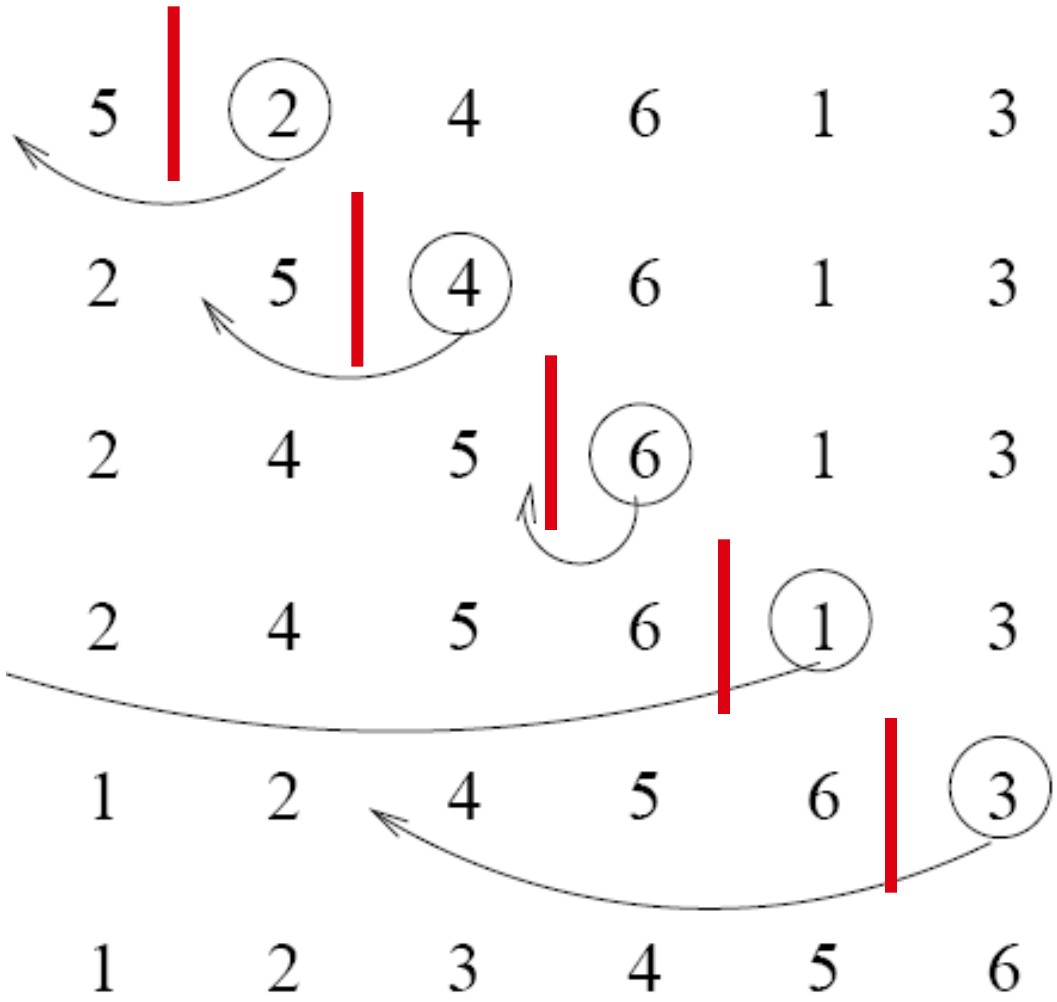
right sub-array

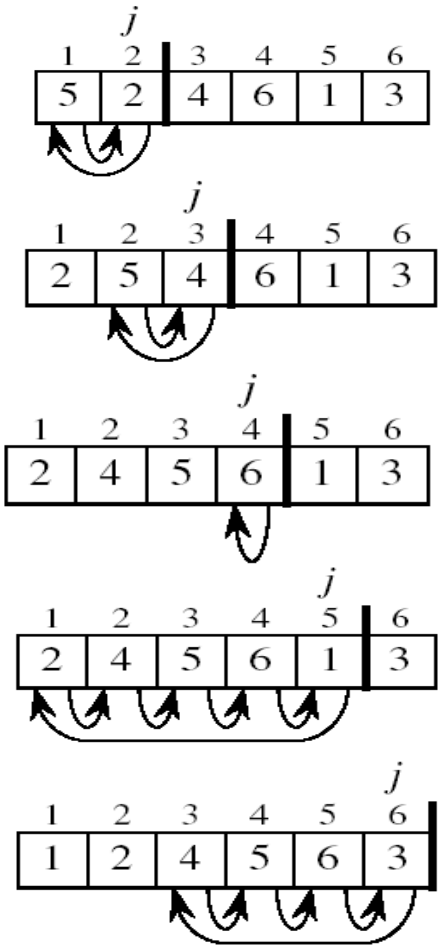


sorted

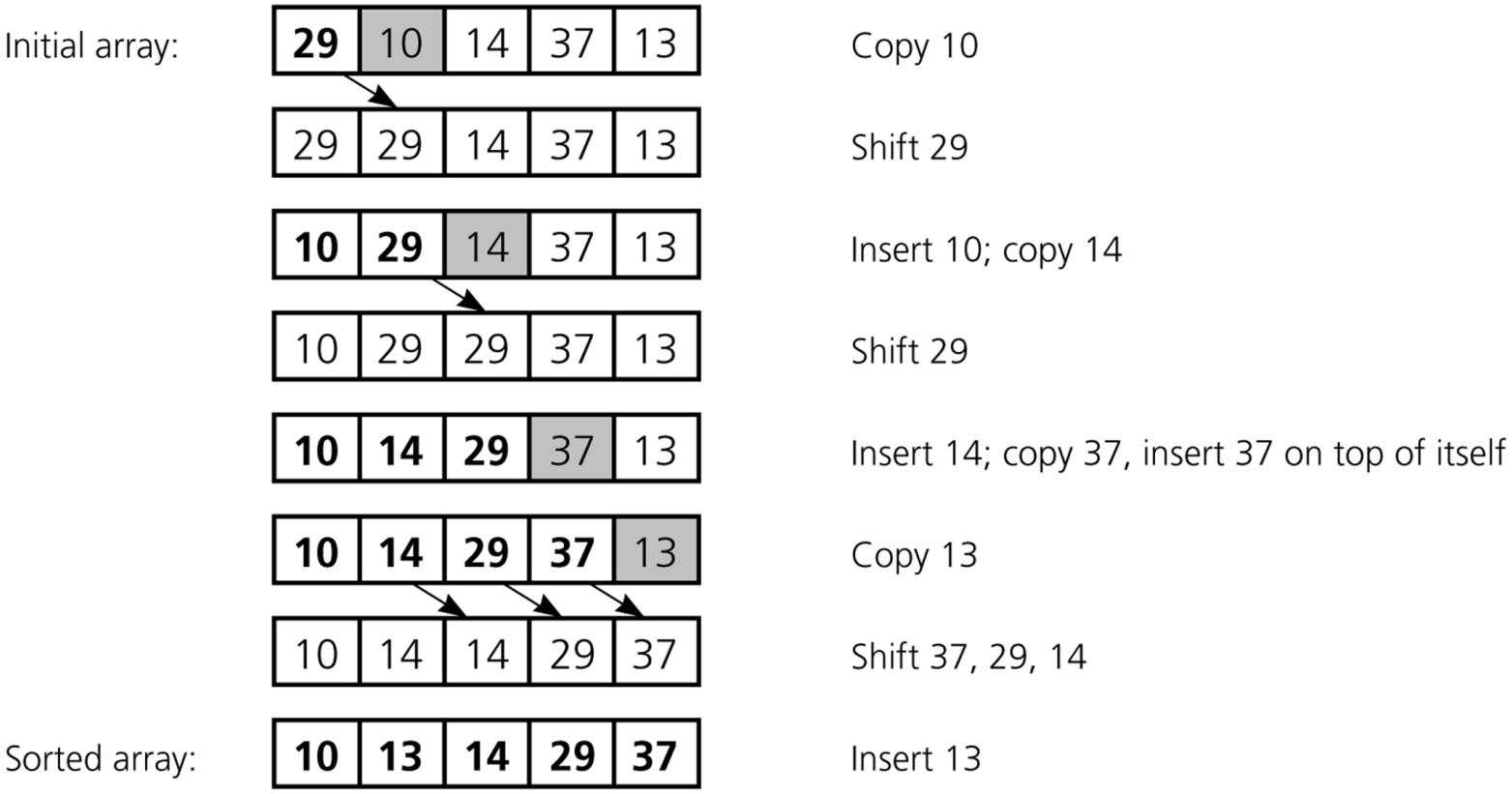
unsorted

Insertion Sort



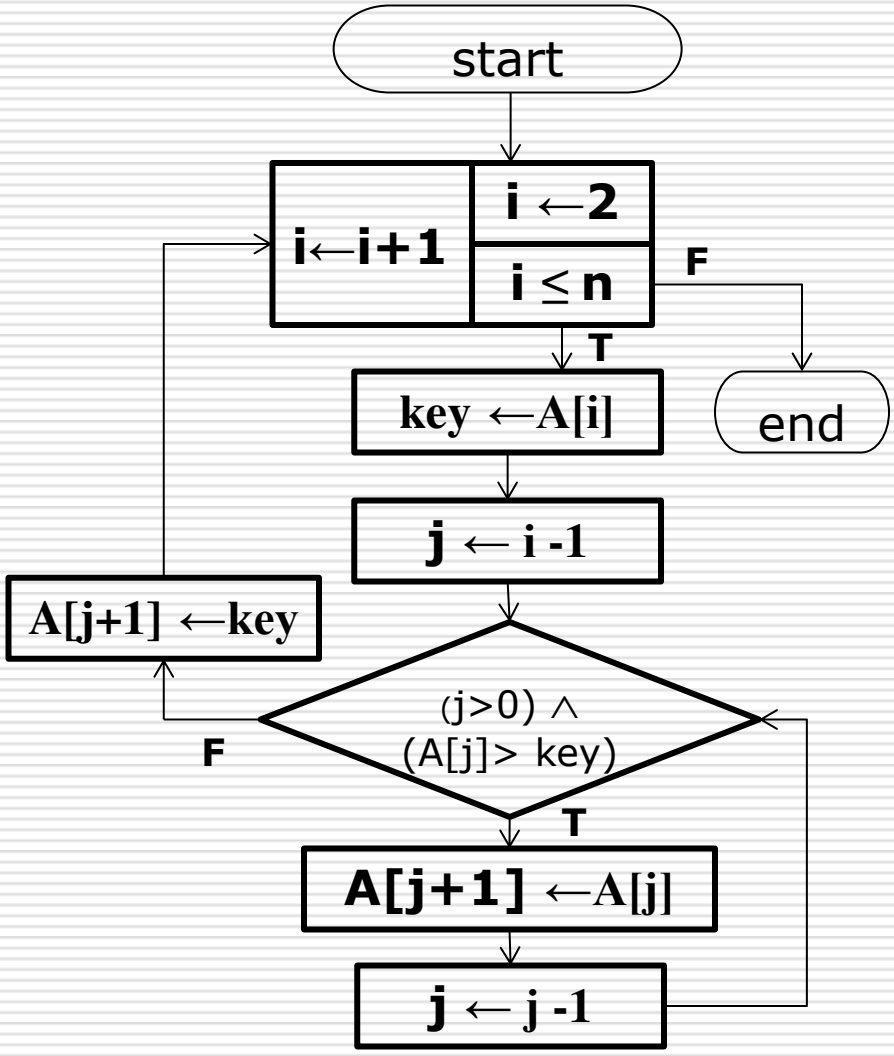


Lecture 02 : Insertion Sort



An insertion sort of an array of five integers

Lecture 02 : Insertion Sort



Flowchart

```
InsertionSort(A, n) {
   for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and
   (A[j] > key) {
         A[j+1] = A[j]
         j = j - 1
      }
      A[j+1] = key
   }
}
```

Coding

Lecture 02 : Insertion Sort

An Example: Insertion Sort

```
InsertionSort(A, n) {
   for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
      }
      A[j+1] = key
   }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort

30	10	40	20
1	2	3	4

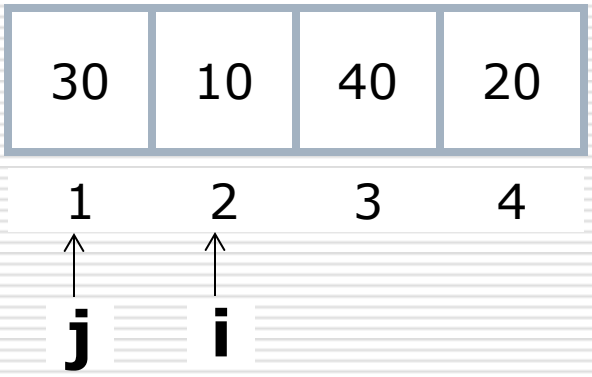
$i = \varnothing \quad j = \varnothing \quad key = \varnothing$

$A[j] = \varnothing \qquad A[j+1] = \varnothing$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort

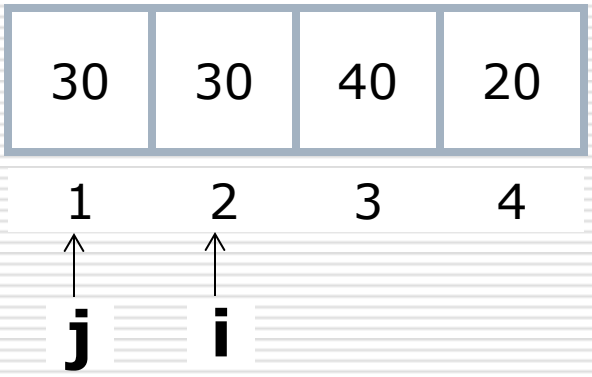


i = 2	j = 1	key = 10
A[j] = 30		A[j+1] = 10

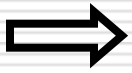
```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort



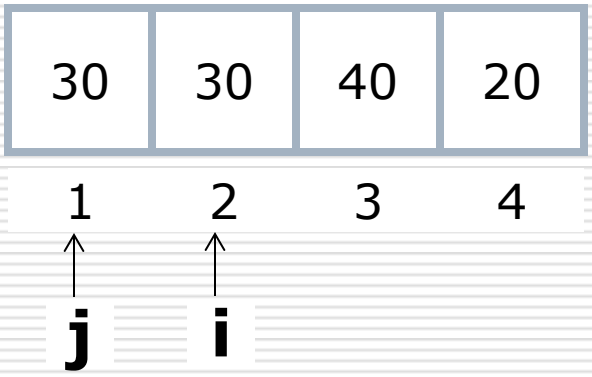
i = 2	j = 1	key = 10
A[j] = 30		A[j+1] = 30



```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort

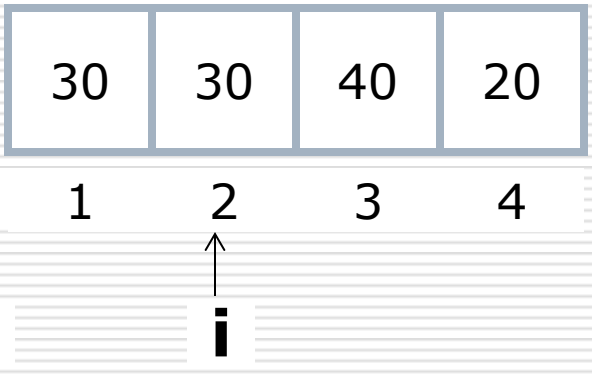


i = 2	j = 1	key = 10
A[j] = 30		A[j+1] = 30

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

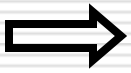
Lecture 02 : Insertion Sort

An Example: Insertion Sort



$i = 2 \quad j = 0 \quad key = 10$

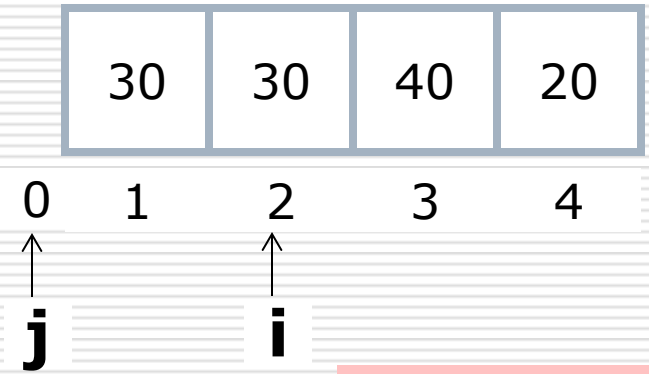
$A[j] = \varnothing \quad A[j+1] = 30$



```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort



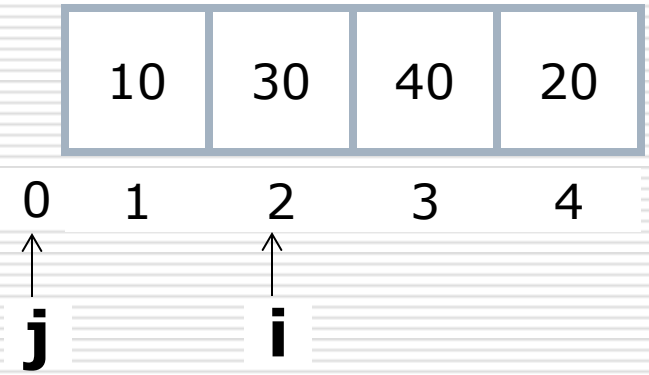
$i = 2 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \quad A[j+1] = 30$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort



$i = 2 \quad j = 0 \quad key = 10$

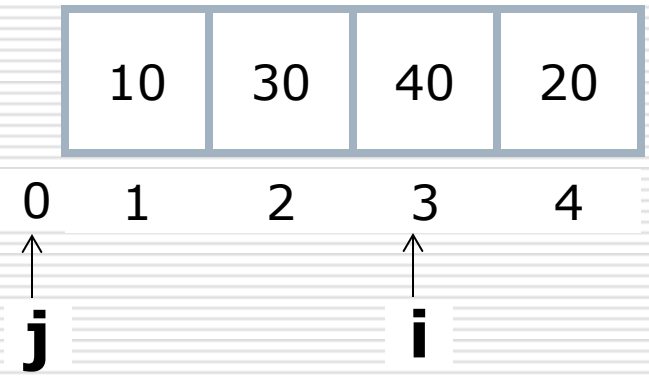
$A[j] = \varnothing \qquad A[j+1] = 10$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Lecture 02 : Insertion Sort

An Example: Insertion Sort

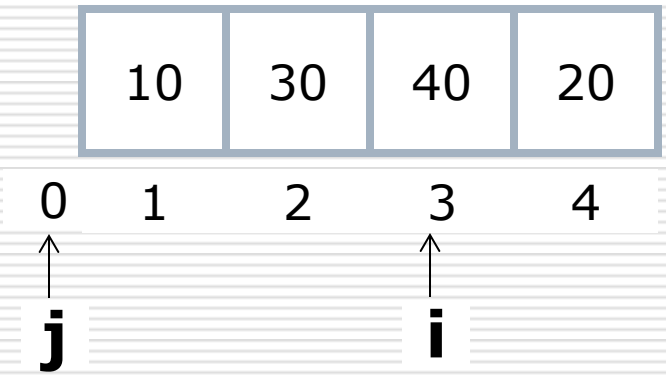


$i = 3 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \qquad A[j+1] = 10$

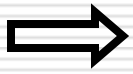
```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



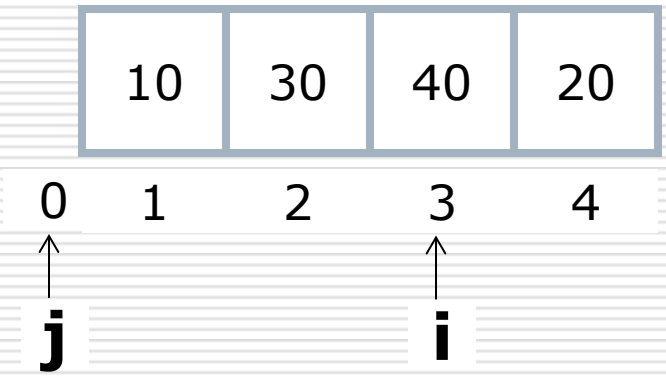
$i = 3 \quad j = 0 \quad key = 40$

$A[j] = \varnothing \qquad A[j+1] = 10$



```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

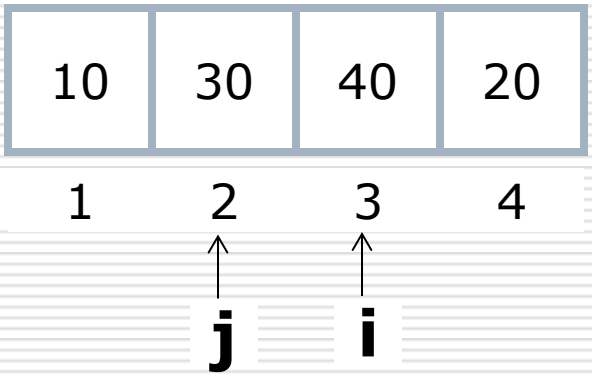


$i = 3 \quad j = 0 \quad key = 40$

$A[j] = \varnothing \qquad A[j+1] = 10$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



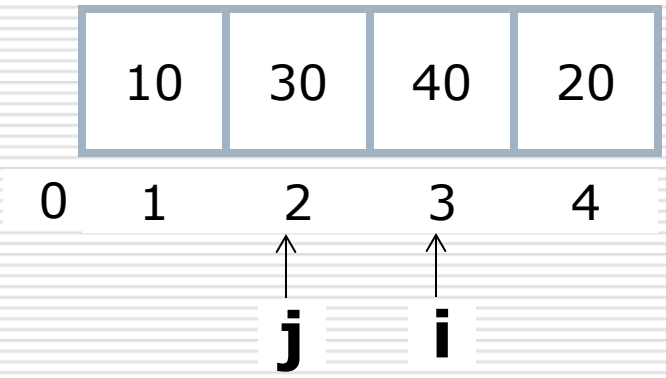
$i = 3 \quad j = 2 \quad key = 40$

$A[j] = 30 \quad A[j+1] = 40$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

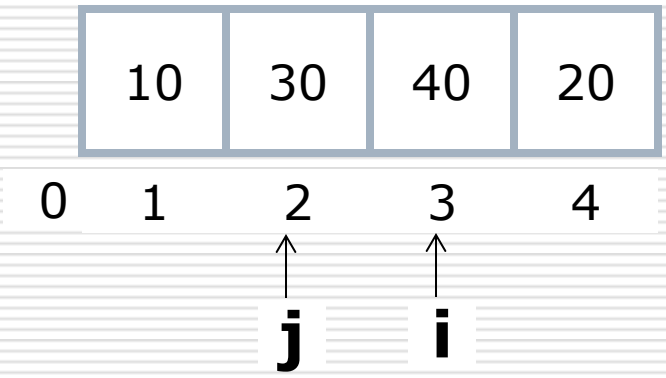


$i = 3 \quad j = 2 \quad key = 40$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

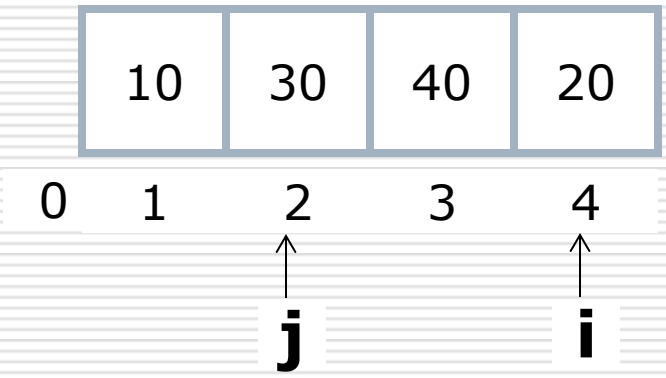


$i = 3 \quad j = 2 \quad key = 40$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

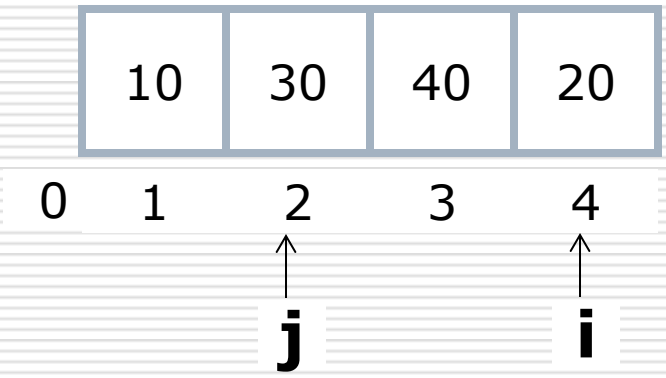


$i = 4 \quad j = 2 \quad key = 40$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



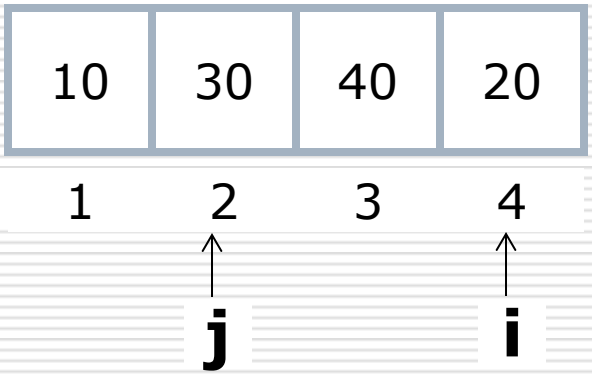
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

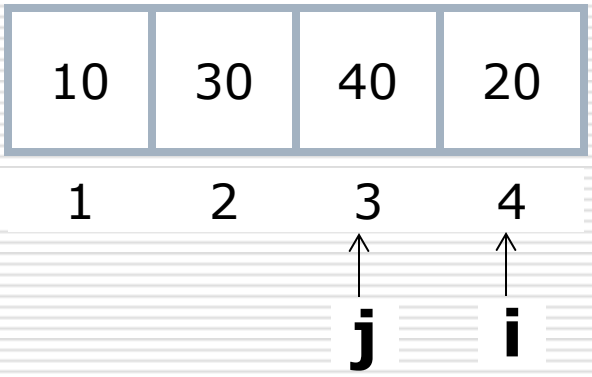


$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



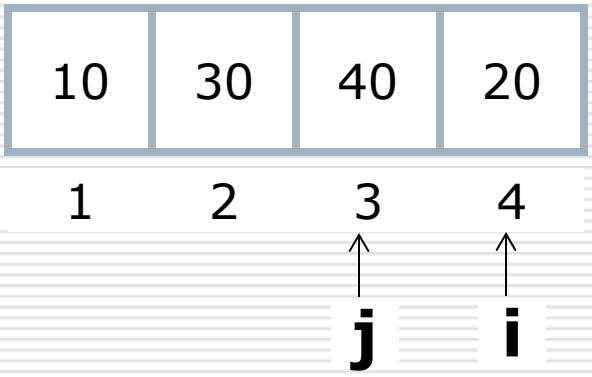
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 20$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

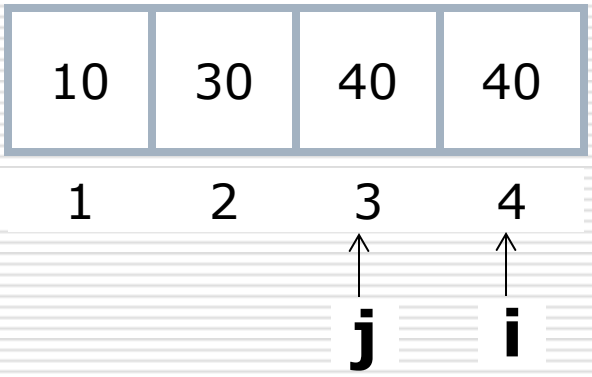


$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 20$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



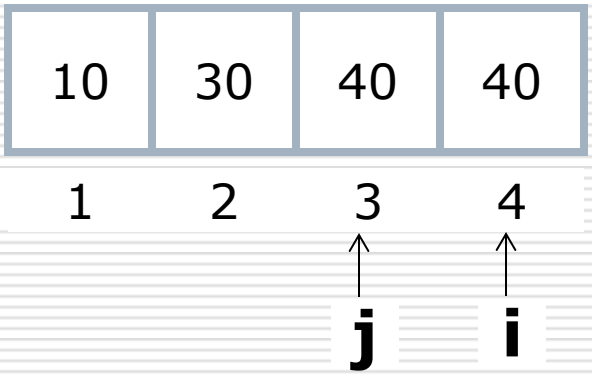
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



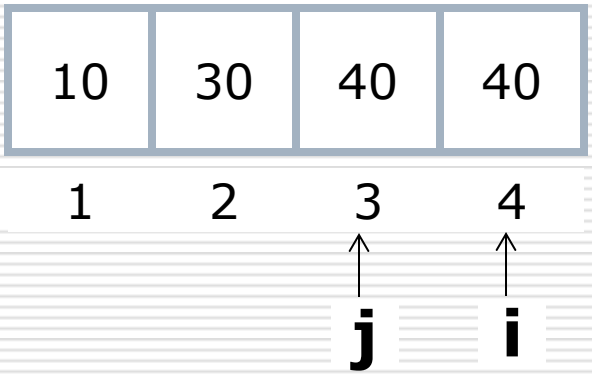
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

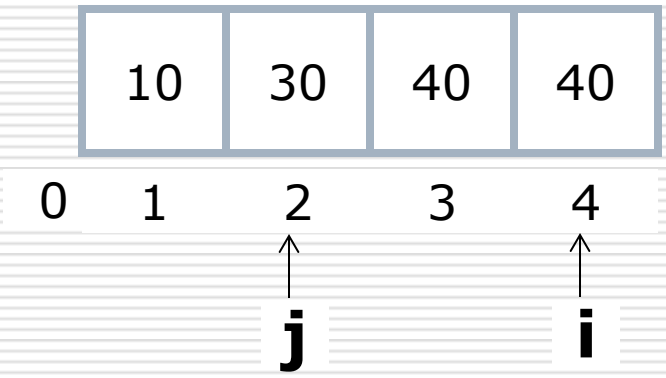


$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



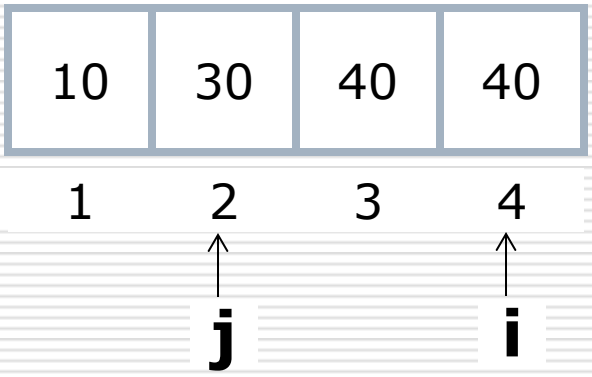
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

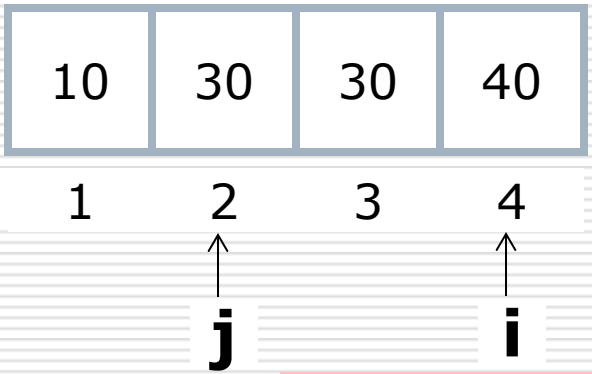


$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



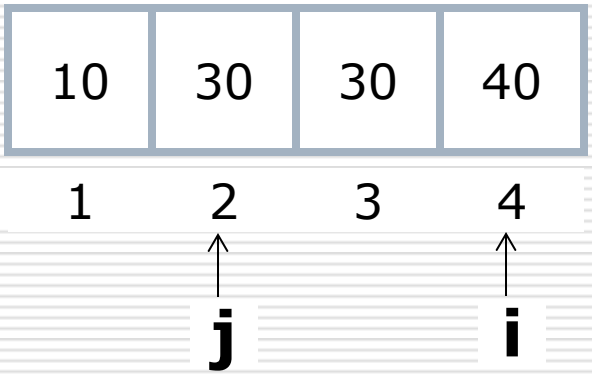
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \qquad A[j+1] = 30$

⇨

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

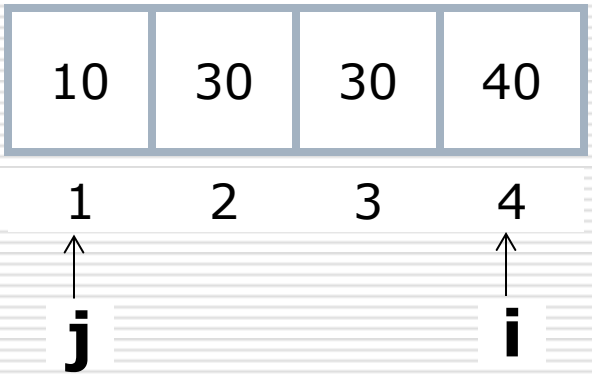


$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
      }
      A[j+1] = key
   }
}
```

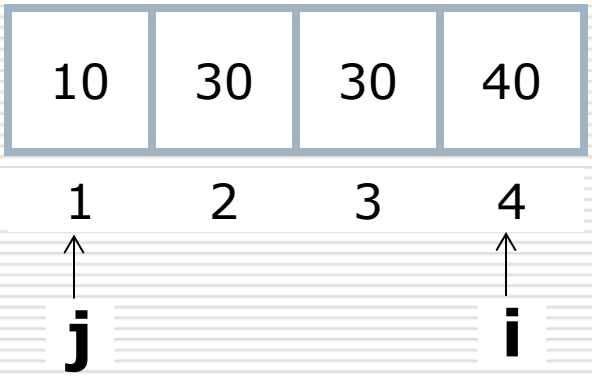
An Example: Insertion Sort



i = 4	j = 1	key = 20
A[j] = 10		A[j+1] = 30

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

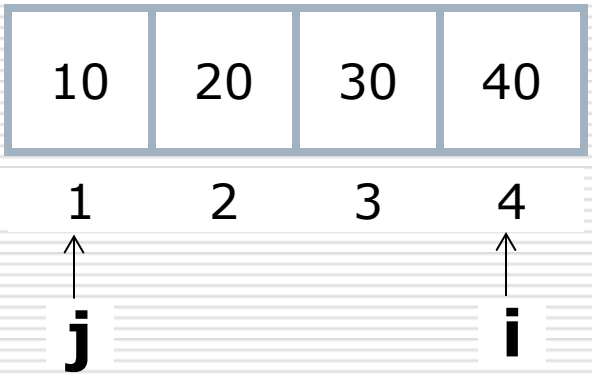


$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort

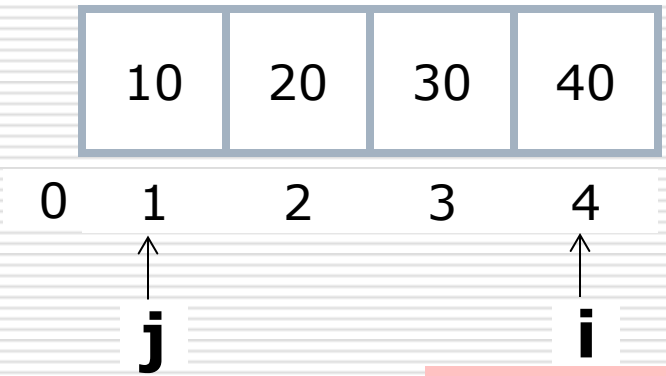


$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 20$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

An Example: Insertion Sort



$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 20$

Done!

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
    }
}
```

Statement	Cost	Times
`InsertionSort(A, n) {`		
`  for i = 2 to n {`	c_1	n
`  key = A[i]`	c_2	(n-1)
`  j = i - 1;`	c_3	(n-1)
`  while (j > 0) and (A[j] > key) {`	c_4	Tx
`  A[j+1] = A[j]`	c_5	Ty
`  j = j - 1`	c_6	Ty
`  }`	0	
`  A[j+1] = key`	c_7	(n-1)
`  }`	0	
`}`		

i = 2, 3, 4,…… ,n

Tx = 1+1, 2+1, 3+1,….(n-1)+1 = 2+3+4+..+n = [n(n+1)/2] – 1

Ty = 1, 2, 3,…. (n-1) = 1+2+3+..+(n-1)= n(n-1)/2

Total Time=c_1*n+(c_2 +c_3)*(n-1)+c_4*Tx+(c_5+c_6) *Ty+c_7* (n-1)

Best-case Analysis

Total Time=c_1*n+(c_2 +c_3)*(n-1)+c_4*Tx+(c_5+c_6) *Ty+c_7* (n-1)

$$T(n) = c_1 n + c_2 (n-1) + c_3 (n-1) + c_4 (n-1) + c_5 * 0 + c_6 * 0 + c_7 (n-1)$$
$$= (c_1 + c_2 + c_3 + c_4 + c_7)n + (c_1 + c_2 + c_3 + c_4 + c_7)$$
$$= an + b$$

We can express this running time as $an + b$ for *constants* a and b that depend on the statement costs c_i; it is thus a ***linear function*** of n.

- Least amount of (time) resource ever needed by algorithm
- Achieved when incoming list is ***already sorted*** in increasing order
- Inner loop is never iterated

Worse-case Analysis

i = 2, 3, 4,…… ,n
Tx = 1+1, 2+1, 3+1,….(n-1)+1 = 2+3+4+..+n = [n(n+1)/2] – 1
Ty = 1, 2, 3,…. (n-1) = 1+2+3+..+(n-1)= n(n-1)/2
Total Time=c_1*n+(c_2 +c_3)*(n-1)+c_4*Tx+(c_5+c_6) *Ty+c_7* (n-1)

$$T(n) = c_1 n + c_2 (n-1) + c_3 (n-1) + c_4 \left[\frac{n(n+1)}{2} - 1\right] + c_5 \left[\frac{n(n-1)}{2}\right] + c_6 \left[\frac{n(n-1)}{2}\right] + c_7 (n-1)$$

$$= \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right) n^2 + (c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7) n - (c_2 + c_3 + c_4 + c_7)$$

$$= an^2 + bn + c$$

We can express this worst-case running time as $an^2 + bn + c$ for constants a, b, and c that again depend on the statement costs c_i; it is thus a ***quadratic function*** of n.

- Greatest amount of (time) resource ever needed by algorithm
- Achieved when incoming list is in ***reverse order***
- Inner loop is iterated the maximum number of times