# Sorting

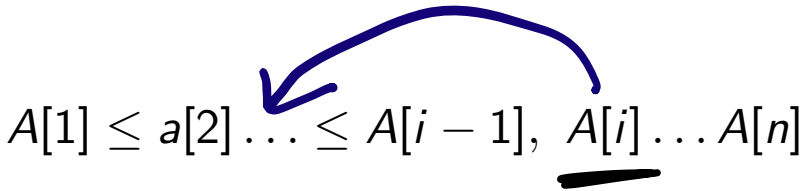
## Definition

**Sorting**: Given a partial order set $(X, \leq)$, problem is to arrange the elements in $X$ so that $x_1 \leq x_2 \leq .... \leq x_n$.

# Insertion Sort

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

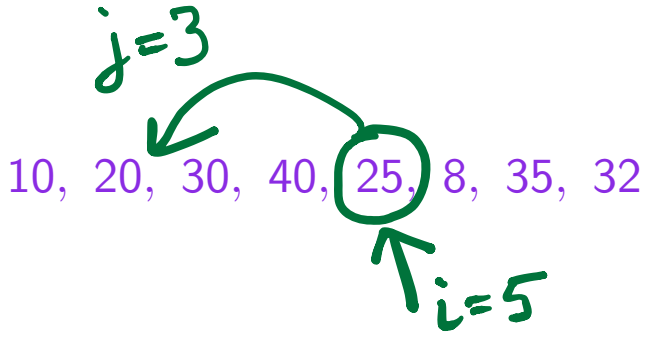
**for** *i: 2 to n* **do**
  Insert $A[i]$ in the partially sorted array $A[1 \ldots i-1]$
**end**



# Insertion Sort

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

**for** *i: 2 to n* **do**
    $j = Find - location(A, i)$ /* function returns the location where $A[i]$ must be inserted so that the array $A[1 \ldots i]$ is sorted */



# Insertion Sort

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

**for** *i: 2 to n* **do**
    $j = Find - location(A, i)$ /* function returns the location where $A[i]$ must be inserted so that the array $A[1 \ldots i]$ is sorted */
    $Insert(A, i, j)$ /* function inserts $A[i]$ in the $j^{th}$ location. This involves shifting elements to the right
**end**

10, 20, 25, 30, 40, 8, 35, 32

# Insertion Sort

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

**for** *i: 2 to n* **do**
    $j = Find - location(A, i)$ /* function returns the location where $A[i]$ must be inserted so that the array $A[1 \ldots i]$ is sorted */
    $Insert(A, i, j)$ /* function inserts $A[i]$ in the $j^{th}$ location. This involves shifting elements to the right
**end**

Let $T(n)$ be the number of primitive steps performed by Insertion Sort on $n$ numbers. Then,

# Insertion Sort

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

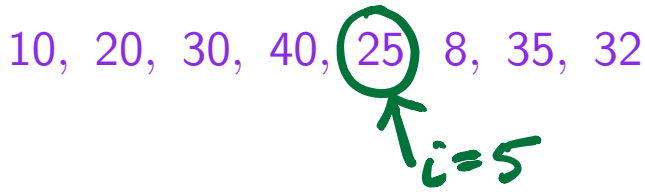
**for** *i: 2 to n* **do**
    $j = Find - location(A, i)$ /* function returns the location where $A[i]$ must be inserted so that the array $A[1 \ldots i]$ is sorted */
    $Insert(A, i, j)$ /* function inserts $A[i]$ in the $j^{th}$ location. This involves shifting elements to the right
**end**

Let $T(n)$ be the number of primitive steps performed by Insertion Sort on $n$ numbers. Then, $T(n) = \sum_i Tfl(i) + \sum_i TIns(i, j)$ where $Tfl()$ and $TIns(,)$ are the number of number of primitive steps performed by $Find - location()$ and $Insert()$ respectively.

# Find-location($A, i$)

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

$key = A[i]$
$j = i - 1$
**while** $j > 0$ && $A[j] > key$ **do**
  $j = j - 1$
**end**
return $j + 1$




A[2]<25
return 3

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

```
key = A[i]
j = i − 1
while j > 0 && A[j] > key do        ← t_i
    j = j − 1
end
return j + 1
```

What is the value of $t_i$ in the worst case?

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

```
key = A[i]
j = i − 1
while j > 0 && A[j] > key do        ← t_i
    j = j − 1
end
return j + 1
```

What is the value of $t_i$ in the worst case? $i$, right?

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

$key = A[i]$
$j = i - 1$
**while** $j > 0$ && $A[j] > key$ **do** $\leftarrow t_i$
  $j = j - 1$
**end**
return $j + 1$

What is the value of $t_i$ in the worst case? $i$, right? $(i-1)+1$.

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

$key = A[i]$
$j = i - 1$
**while** $j > 0$ && $A[j] > key$ **do** ← $t_i$
  $j = j - 1$
**end**
return $j + 1$

What is the value of $t_i$ in the worst case? $i$, right? $(i-1)+1$.
What is the value of $t_i$ in the best case?

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

$key = A[i]$
$j = i - 1$
**while** $j > 0$ && $A[j] > key$ **do** 
    $j = j - 1$
**end**
return $j + 1$

What is the value of $t_i$ in the worst case? $i$, right? $(i-1)+1$.
What is the value of $t_i$ in the best case? 1, right?

# Find-location($A, i$)

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

```
key = A[i]
j = i − 1
while j > 0 && A[j] > key do
|   j = j − 1
end
return j + 1
```

What is the value of $t_i$ in the worst case? $i$, right? $(i-1)+1$.
What is the value of $t_i$ in the best case? 1, right?
Number of times the while-statement is executed = the number of times the key comparison results in a success +1.

# Find-location$(A, i)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** index $j$: $A[j] \leq A[i] < A[j+1]$

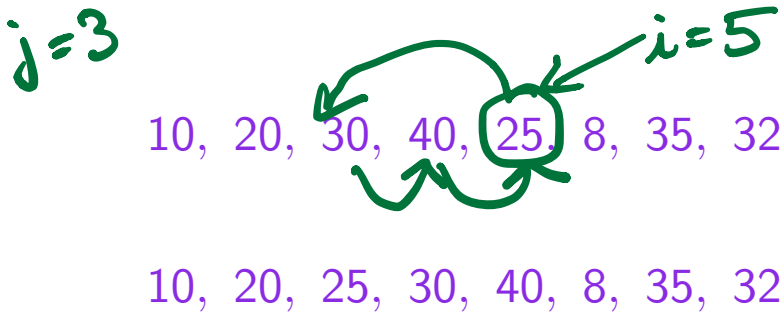
$key = A[i]$
$j = i - 1$
**while** $j > 0$ && $A[j] > key$ **do** $\leftarrow t_i$
  $j = j - 1$
**end**
return $j + 1$

What is the value of $t_i$ in the worst case? $i$, right? $(i-1)+1$.
What is the value of $t_i$ in the best case? 1, right?
Number of times the while-statement is executed = the number of times the key comparison results in a success +1. Thus,
$1 \leq Tfl(i) \leq i$.

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do**
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$



# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** $\leftarrow t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** $\longleftarrow t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1$ right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1)-(j-1)+1)$ right? i.e. $i - j + 1$.

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** 

    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1)-(j-1)+1)$ right? i.e. $i - j + 1$.
which in the worst case is?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** $\longleftarrow t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_j$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right? When?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** 
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1)-(j-1)+1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right? When? when $j = i$, right?

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right? When? when $j = i$, right?
Number of times the while-statement is executed = the number of assignments +1.

# Insert$(i, j)$

**input :** Sorted array $A[1] \leq A[2] \leq ..... \leq A[i-1]$ and $A[i]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[i]$ inserting $A[i]$ in the $j^{th}$ location.

$k = i - 1$
**while** $k > j - 1$ **do** ← $t_{ij}$
    $A[k+1] = A[k]$
    $k = k - 1$
**end**
$A[k+1] = key$

What is the value of $t_{ij}$? $(i-1) - (j-1) + 1)$ right? i.e. $i - j + 1$.
which in the worst case is? $i$, right? When? when $j = 1$, right?
and in the best case? 1, right? When? when $j = i$, right?
Number of times the while-statement is executed = the number of assignments +1. Thus, $1 \leq TIns(i, j) \leq i$.

# Insertion Sort : Analysis

**for** *i: 2 to n* **do**
  $j = Find - location(A, i)$ /* function returns the location where $A[i]$ must be inserted so that the array $A[1 \ldots i]$ is sorted */
  $Insert(A, i, j)$ /* function inserts $A[i]$ in the $j^{th}$ location. This involves shifting elements to the right
**end**

$T(n) = \sum_i Tfl(i) + \sum_i TIns(i, j)$ within constant factors.
Where $Tfl()$ is the number of comparisons done by $Find - location()$ and $TIns(,)$ is the number of shifts/moves/assignments performed by $Insert()$.
Thus the time complexity of Insertion sort is determined by the number of key-comparisons and the number of shift operations it performs.

# Insertion Sort : Analysis

Thus, in the worst case

$$T(n) = \sum_{i=1 \text{ to } n-1} i + \sum_{i=1 \text{ to } n-1} i = \theta(\frac{n(n-1)}{2})$$

# Insertion Sort : Analysis

Thus, in the worst case

$$T(n) = \sum_{i=1 \ to \ n-1} i + \sum_{i=1 \ to \ n-1} i = \theta(\frac{n(n-1)}{2})$$

And, in the best case

$$T(n) = \sum_{i=1 \ to \ n-1} 1 + \sum_{i=1 \ to \ n-1} 1 = \theta(n)$$

# Practically, a better implementation of Insertion Sort : Shift as you Compare

**input :** Array: $A[1], A[2]..., A[n]$
**output:** Sorted array; $A[1] \leq A[2] \leq ..... \leq A[n]$

```
for i: 2 to n do
    key = A[i]
    j = i − 1
    while j > 0 && A[j] > key do
        A[j + 1] = A[j]
        j = j − 1
    end
    A[j + 1] = key
end
```

No need to run the loop twice.

# Way 2

To analyse complexity of Insertion Sort

# A simpler way to look at running time of Insertion Sort

| INSERTION-SORT($A$) | *cost* | *times* |
|---|---|---|
| 1 **for** $j = 2$ **to** $A.length$ | $c_1$ | $n$ |
| 2 $key = A[j]$ | $c_2$ | $n-1$ |
| 3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$. | 0 | $n-1$ |
| 4 $i = j - 1$ | $c_4$ | $n-1$ |
| 5 **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ |
| 6 $A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 7 $i = i - 1$ | $c_7$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 8 $A[i+1] = key$ | $c_8$ | $n-1$ |

$$\text{total cost} = \sum_{\text{Statement } x} \text{cost}_x \times \text{times}_x$$

# BEST CASE

| INSERTION-SORT($A$) | *cost* | *times* |
|---|---|---|
| 1 **for** $j = 2$ **to** $A.length$ | $c_1$ | $n$ |
| 2 $key = A[j]$ | $c_2$ | $n-1$ |
| 3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$. | 0 | $n-1$ |
| 4 $i = j-1$ | $c_4$ | $n-1$ |
| 5 **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ |
| 6 $A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| 7 $i = i-1$ | $c_7$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| 8 $A[i+1] = key$ | $c_8$ | $n-1$ |

for best case $t_j = 1$

$- 0$

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

$$= c_1 n + c_2 n - c_2 + c_4 n - c_4 + c_5 n - c_5 + c_8 n - c_8$$

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

$$= an + b \quad \text{for some } a \text{ and } b$$

Running time of IS is a linear $f^n$ of n in best case.

# WORST CASE

| INSERTION-SORT($A$) | *cost* | *times* |
|---|---|---|
| 1 **for** $j = 2$ **to** $A.length$ | $c_1$ | $n$ |
| 2 $key = A[j]$ | $c_2$ | $n-1$ |
| 3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$. | 0 | $n-1$ |
| 4 $i = j-1$ | $c_4$ | $n-1$ |
| 5 **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ → for worst case $t_j = j$ |
| 6 $A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| 7 $i = i-1$ | $c_7$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| 8 $A[i+1] = key$ | $c_8$ | $n-1$ |

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1)$$

$$= c_1 n + c_2 n - c_2 + c_4 n - c_4 + c_5\left(\frac{n^2}{2}\right) + c_5\left(\frac{n}{2}\right) - c_5 + c_6\left(\frac{n^2}{2}\right) - c_6\left(\frac{n}{2}\right) + c_7\left(\frac{n^2}{2}\right) - c_7\left(\frac{n}{2}\right) + c_8 n - c_8$$

$$= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n + (c_2 - c_4 - c_5 - c_8)$$

$= an^2 + bn + c$

for some $a, b$ and $c$.

Running time of IS in worst case is a quadratic fn.

# Correctness of Insertion Sort

Refer Ch-2 (CLRS)

# Insertion Sort

**Consider the following algorithm for insertion sort**

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      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          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

# What is the loop invariant?

# Loop invariant for Insertion Sort

- At the start of each iteration of the for loop, the sub array A[1... j-1] consists of elements originally in A[1...j-1], but in sorted order.

# To show about loop invariant:

- **Initialisation**: It is true prior to the first iteration of the loop
- **Maintenance**: if it is true before an iteration of the loop, it remains true before the next iteration.
- **Termination**: when the loop terminates, the invariant gives us a property that helps to show that the algorithm is correct

# To show about loop invariant:

- **Initialisation**: It is true prior to the first iteration of the loop

  ↳ corresponds to the base case of MI

- **Maintenance**: if it is true before an iteration of the loop, it remains true before the next iteration.

  ↳ corresponds to the inductive step of MI

- **Termination**: when the loop terminates, the invariant gives us a property that helps to show that the algorithm is correct

  ↳ 'induction' stops when loop terminates

# Initialisation

**To show LI holds before the first iteration of the loop**

- J=2
- The sub array A[1 ... j-1] consists of a single element => A[1]
- This is the original element in A[1]
- Trivially sorted as an element in itself is already sorted
- LI holds prior to the first iteration of the loop.

# Maintenance

## To show that each iteration maintains this LI

- While considering A[j], we might move A[j-1], A[j-2], A[j-3] and so on one position to the right until you find the correct position of A[j]
- The sub array A[1...j] then contains all the elements originally in A[1...j] but in sorted order
- If you increment j again, the LI is true before the next iteration.

# Termination

## Examine conditions that terminate the loop

- When j> A.length() [A.length() = n]
- That means, at termination time, j=n+1. **Why not n+2?**
- Substituting n+1 for j in the LI, we can say that the subarray A[1...n] contains all the elements originally in A[1...n] but in sorted order.
- Because A[1...n] is the entire array, the algorithm is correct.

# Homework 2

Due: 23-01-2022, 11:59 PM

**1. Implementation of Insertion Sort, Selection Sort, Bubble sort**
**- Count the number of key comparisons and assignments for various inputs and plot the graph for both of them.**
**- For every input size n , run it with 10 different data points generated randomly.**
**- Compute the minimum, maximum and average of the number of key comparisons (and assignments) for each input size.**
**- Plot the graph for each case - best, worst and average number of comparisons (and assignments) .**
**- n varies from 10 to 100 in steps of 5.**

**2. Prove the correctness of selection sort and bubble sort. Very clearly state their loop invariants as per your implementation.**

n

IMPT