

# Sorting problem

## The sorting problem

- **Input:** a sequence of  $n$  numbers  $\langle a_1, a_2, \dots, a_n \rangle$
- **Output:** a permutation  $\langle a_1', a_2', \dots, a_n' \rangle$ , s.t.  $a_1' \leq a_2' \leq \dots \leq a_n'$
- An **instance** of a problem:
  - the input needed to compute a solution (e.g.:  $\langle 5, 3, 6, 2 \rangle$ )
- An algorithm is **correct** if it ends with the correct output in a finite amount of time, on any legitimate input



# Insertion Sort (C)

```
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  
    }  
}
```



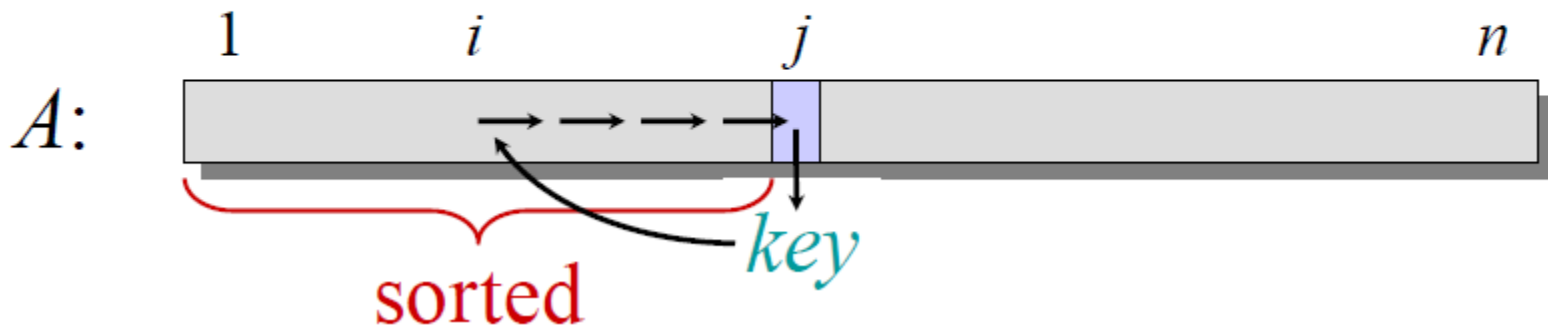
# Insertion sort(Pascal)

- For  $j = 2$  to  $\text{length}[A]$  do
- $\text{key} = A[j]$
- Insert  $A[j]$  into the sorted sequence  $A[1..j-1]$
- $i = j - 1$
- while  $i > 0$  and  $A[i] > \text{key}$  do
- $A[i+1] = A[i]$
- $i = i - 1$
- $A[i+1] = \text{key}$



“pseudocode”

```
INSERTION-SORT ( $A, n$ )    ▷  $A[1 \dots n]$   
  for  $j \leftarrow 2$  to  $n$   
    do  $key \leftarrow A[j]$   
       $i \leftarrow j - 1$   
      while  $i > 0$  and  $A[i] > key$   
        do  $A[i+1] \leftarrow A[i]$   
           $i \leftarrow i - 1$   
       $A[i+1] = key$ 
```



# An Example: Insertion Sort

30	10	40	20
1	2	3	4

$i = \emptyset$	$j = \emptyset$	$key = \emptyset$
$A[j] = \emptyset$	$A[j+1] = \emptyset$	



```
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

30	10	40	20
1	2	3	4

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

```
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  
    }  
}
```



# An Example: Insertion Sort

30	30	40	20
1	2	3	4

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

```
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  
    }  
}
```

➡




# An Example: Insertion Sort

30	30	40	20
1	2	3	4

$i = 2$	$j = 1$	$\text{key} = 10$
$A[j] = 30$	$A[j+1] = 30$	

```
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  
    }  
}
```





# An Example: Insertion Sort

30	30	40	20
1	2	3	4

$i = 2$	$j = 0$	$\text{key} = 10$
$A[j] = \emptyset$	$A[j+1] = 30$	

```
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

30	30	40	20
1	2	3	4

$i = 2$	$j = 0$	$\text{key} = 10$
$A[j] = \emptyset$	$A[j+1] = 30$	

```
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

10	30	40	20
1	2	3	4

$i = 2$	$j = 0$	$\text{key} = 10$
$A[j] = \emptyset$	$A[j+1] = 10$	

```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 0$	$\text{key} = 10$
$A[j] = \emptyset$	$A[j+1] = 10$	



```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 0$	$\text{key} = 40$
$A[j] = \emptyset$	$A[j+1] = 10$	



```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 0$	$\text{key} = 40$
$A[j] = \emptyset$	$A[j+1] = 10$	



```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 2$	$\text{key} = 40$
$A[j] = 30$	$A[j+1] = 40$	



```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 2$	$\text{key} = 40$
$A[j] = 30$	$A[j+1] = 40$	

```
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

10	30	40	20
1	2	3	4

$i = 3$	$j = 2$	$\text{key} = 40$
$A[j] = 30$	$A[j+1] = 40$	

```
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

10	30	40	20
1	2	3	4

$i = 4$	$j = 2$	$\text{key} = 40$
$A[j] = 30$	$A[j+1] = 40$	



```
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

10	30	40	20
1	2	3	4

$i = 4$	$j = 2$	$\text{key} = 20$
$A[j] = 30$	$A[j+1] = 40$	



```
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

10	30	40	20
1	2	3	4

$i = 4$	$j = 2$	$\text{key} = 20$
$A[j] = 30$	$A[j+1] = 40$	



```
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

10	30	40	20
1	2	3	4

$i = 4$	$j = 3$	$key = 20$
$A[j] = 40$	$A[j+1] = 20$	



```
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

10	30	40	20
1	2	3	4

$i = 4$	$j = 3$	$key = 20$
$A[j] = 40$	$A[j+1] = 20$	

```
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

10	30	40	40
1	2	3	4

$i = 4$	$j = 3$	$key = 20$
$A[j] = 40$		$A[j+1] = 40$

```
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

10	30	40	40
1	2	3	4

$i = 4$	$j = 3$	$key = 20$
$A[j] = 40$		$A[j+1] = 40$

```
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

10	30	40	40
1	2	3	4

$i = 4$	$j = 3$	$key = 20$
$A[j] = 40$		$A[j+1] = 40$

```
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

10	30	40	40
1	2	3	4

$i = 4$	$j = 2$	$key = 20$
$A[j] = 30$	$A[j+1] = 40$	

```
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

10	30	40	40
1	2	3	4

$i = 4$	$j = 2$	$key = 20$
$A[j] = 30$	$A[j+1] = 40$	

```
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

10	30	30	40
1	2	3	4

$i = 4$	$j = 2$	$key = 20$
$A[j] = 30$		$A[j+1] = 30$

```
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  
    }  
}
```




# An Example: Insertion Sort

10	30	30	40
1	2	3	4

$i = 4$	$j = 2$	$key = 20$
$A[j] = 30$	$A[j+1] = 30$	

```
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  
    }  
}
```



# An Example: Insertion Sort

10	30	30	40
1	2	3	4

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

```
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

10	30	30	40
1	2	3	4

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

```
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

10	20	30	40
1	2	3	4

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

```
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

10	20	30	40
1	2	3	4

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

```
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  
    }  
}
```

*Done!*



# Correctness Analysis

- Loop invariant:

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

- **Initialization:** 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 useful property that helps show that the algorithm is correct.



# 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
```

```
  }
```

```
}
```

What is the *precondition* for this loop?



# 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  
    }  
}
```

*How many times will this loop execute?*



# Insertion Sort

Statement	Effort
<b>InsertionSort(A, n) {</b>	
<b>for i = 2 to n {</b>	$c_1 n$
<b>key = A[i]</b>	$c_2(n-1)$
<b>j = i - 1;</b>	$c_3(n-1)$
<b>while (j &gt; 0) and (A[j] &gt; key) {</b>	$c_4 T$
<b>A[j+1] = A[j]</b>	$c_5(T-(n-1))$
<b>j = j - 1</b>	$c_6(T-(n-1))$
<b>}</b>	0
<b>A[j+1] = key</b>	$c_7(n-1)$
<b>}</b>	0
<b>}</b>	

$T = t_2 + t_3 + \dots + t_n$  where  $t_i$  is number of while expression evaluations for the  $i^{\text{th}}$  for loop iteration



# Analyzing Insertion Sort

- $T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4T + c_5(T - (n-1)) + c_6(T - (n-1)) + c_7(n-1)$   
 $= c_8T + c_9n + c_{10}$
- What can  $T$  be?
  - Best case -- inner loop body never executed  
✓  $t_i = 1 \rightarrow T(n)$  is a linear function
  - Worst case -- inner loop body executed for all previous elements  
✓  $t_i = i \rightarrow T(n)$  is a quadratic function
  - Average case  
✓ ???



# Analysis

- Simplifications
  - Ignore actual and abstract statement costs
  - *Order of growth* is the interesting measure:
    - Highest-order term is what counts

Remember, we are doing asymptotic analysis

As the input size grows larger it is the high order term that dominates



# Upper Bound Notation

- We say InsertionSort's run time is  $O(n^2)$ 
  - Properly we should say run time is *in*  $O(n^2)$
  - Read  $O$  as “Big- $O$ ” (you’ll also hear it as “order”)
- In general a function
  - $f(n)$  is  $O(g(n))$  if there exist positive constants  $c$  and  $n_0$  such that  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$
- Formally
  - $O(g(n)) = \{ f(n): \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } f(n) \leq c \cdot g(n) \forall n \geq n_0$





# Insertion Sort Is $O(n^2)$

- Proof

- Suppose runtime is  $an^2 + bn + c$ 
  - If any of  $a$ ,  $b$ , and  $c$  are less than 0 replace the constant with its absolute value
- $an^2 + bn + c \leq (a + b + c)n^2 + (a + b + c)n + (a + b + c)$
- $\leq 3(a + b + c)n^2$  for  $n \geq 1$
- Let  $c' = 3(a + b + c)$  and let  $n_0 = 1$

- Question

- Is InsertionSort  $O(n^3)$ ?
- Is InsertionSort  $O(n)$ ?



# Big O Fact

- A polynomial of degree  $k$  is  $O(n^k)$
- Proof:
  - Suppose  $f(n) = b_k n^k + b_{k-1} n^{k-1} + \dots + b_1 n + b_0$ 
    - Let  $a_i = |b_i|$
  - $f(n) \leq a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0$

$$\leq n^k \sum a_i \frac{n^i}{n^k} \leq n^k \sum a_i \leq cn^k$$

