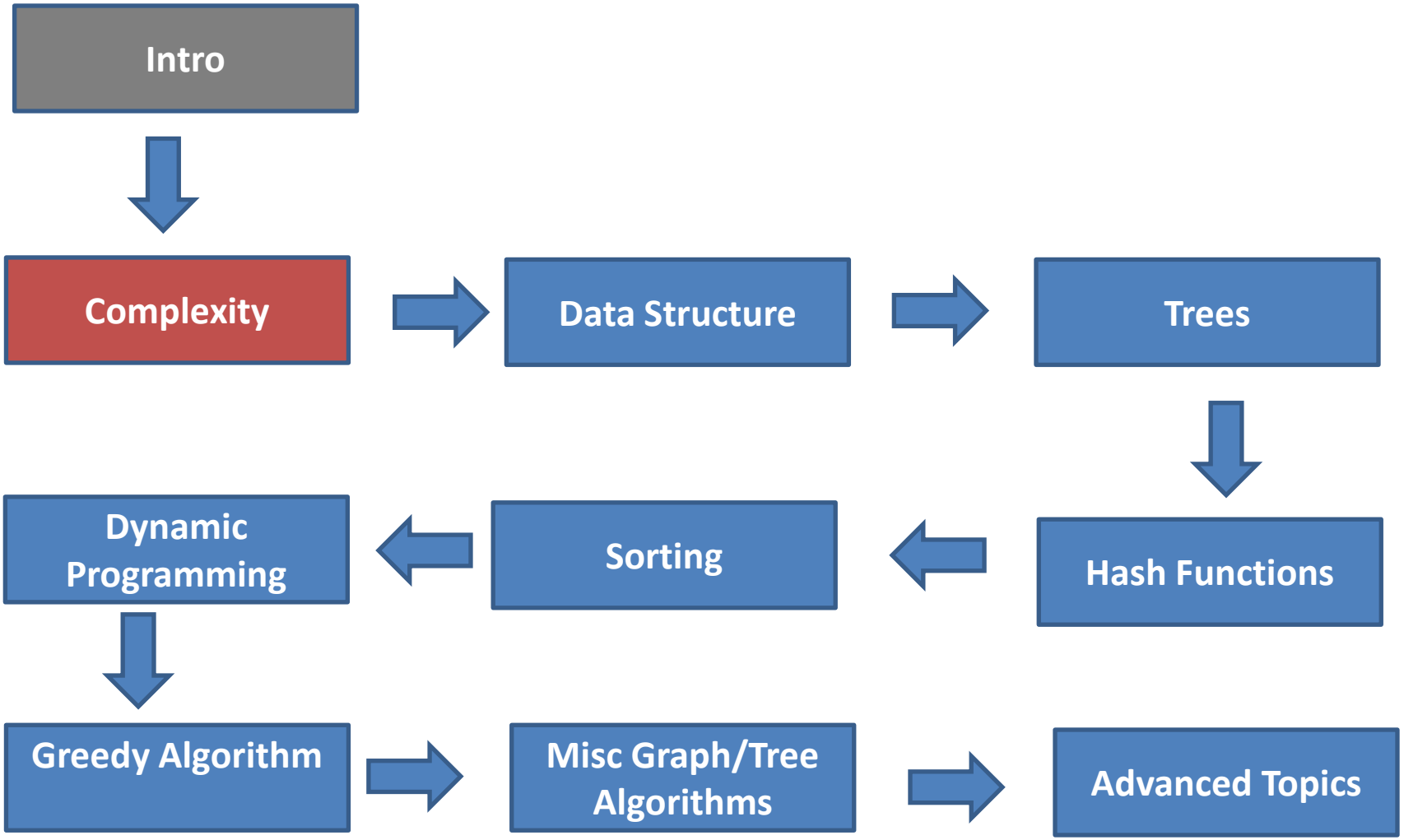
An Introduction to Algorithms

By

Hossein Rahmani

h_rahmani@iust.ac.ir

http://webpages.iust.ac.ir/h_rahmani/


Intro
Complexity
Data Structure
Trees
Hash Functions
Sorting
Dynamic Programming
Greedy Algorithm
Misc Graph/Tree Algorithms
Advanced Topics

The problem of sorting

Input: sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

Output: permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ Such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Example:

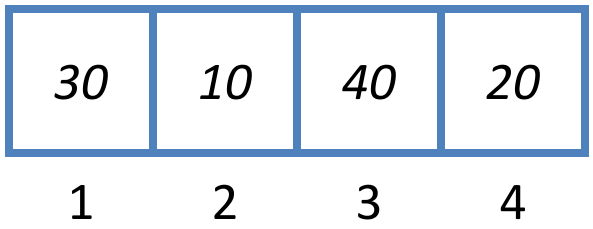
Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

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
  }
}
```

An Example: Insertion Sort

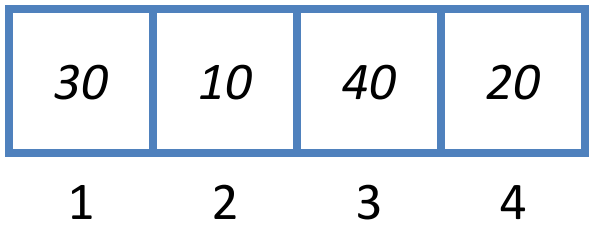


$i = \varnothing \quad j = \varnothing \quad key = \varnothing$

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

```
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
   }
}
```

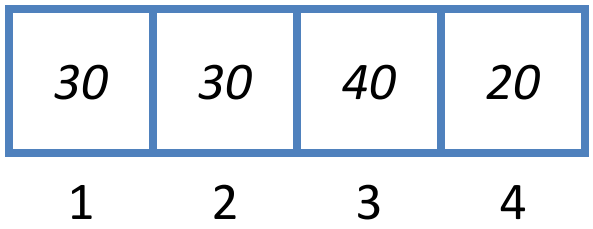
An Example: Insertion Sort



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
   }
}
```

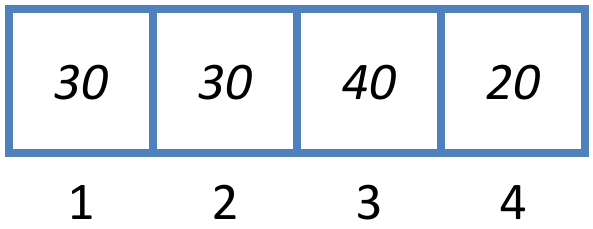
An Example: Insertion Sort



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
   }
}
```

An Example: Insertion Sort

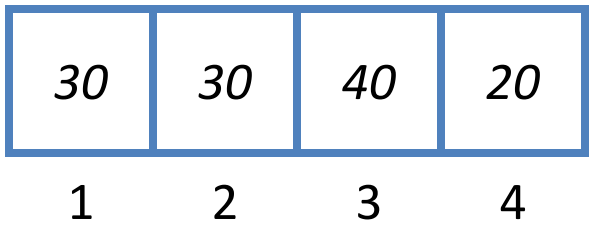


$i = 2 \quad j = 1 \quad key = 10$

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

```
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

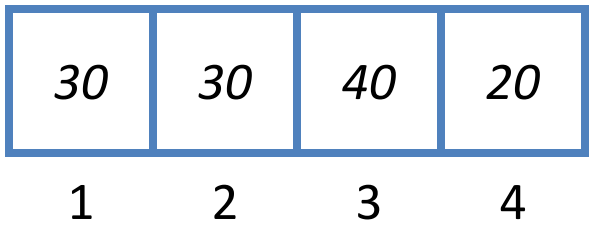


$i = 2 \quad j = 0 \quad key = 10$

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

```
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

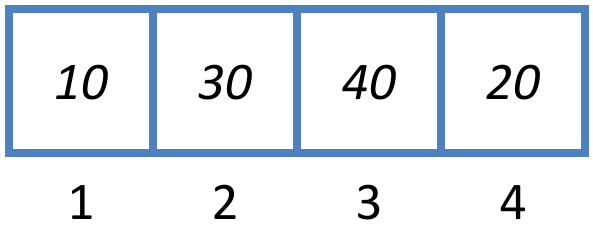


$i = 2 \quad j = 0 \quad key = 10$

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

```
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

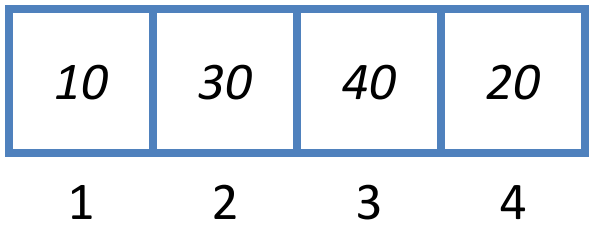


$i = 2 \quad j = 0 \quad key = 10$

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

```
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
   }
}
```

An Example: Insertion Sort

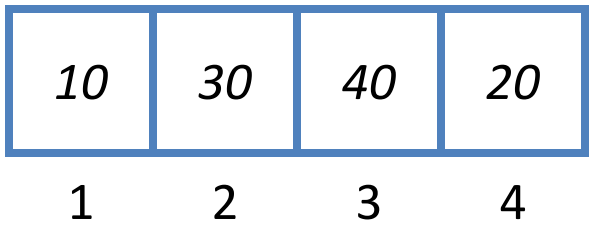


$i = 3 \quad j = 0 \quad key = 10$

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

```
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

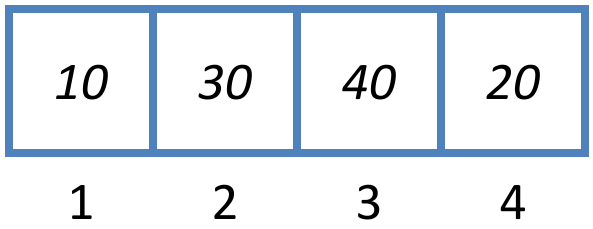


$i = 3 \quad j = 0 \quad key = 40$

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

```
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

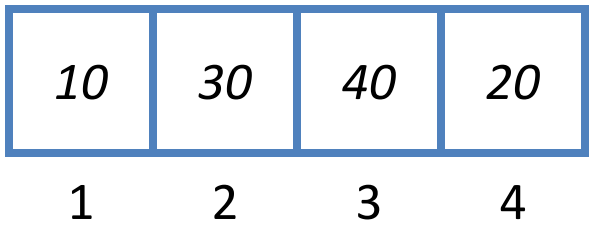


$i = 3 \quad j = 0 \quad key = 40$

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

```
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

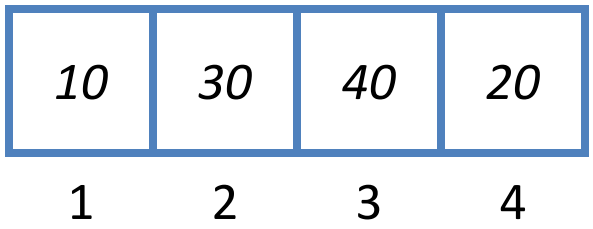


$i = 3 \quad j = 2 \quad key = 40$

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

```
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

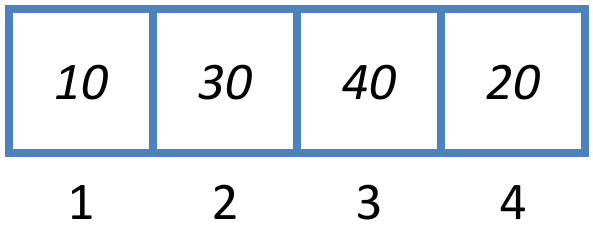


$i = 3 \quad j = 2 \quad key = 40$

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

```
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

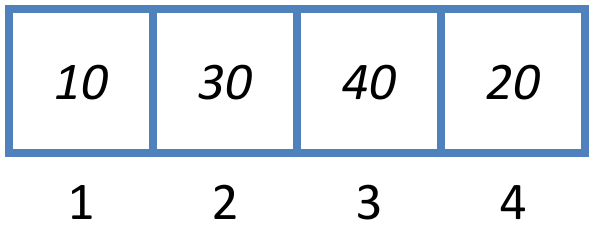


$i = 3 \quad j = 2 \quad key = 40$

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

```
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

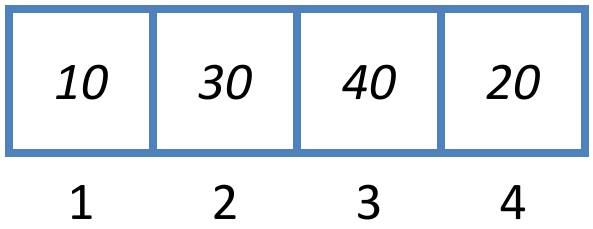


$i = 4 \quad j = 2 \quad key = 40$

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

```
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

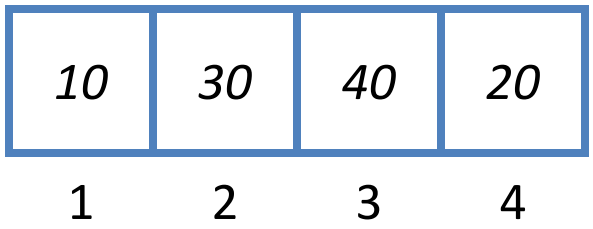


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

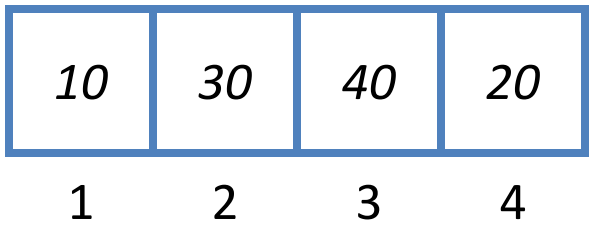


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

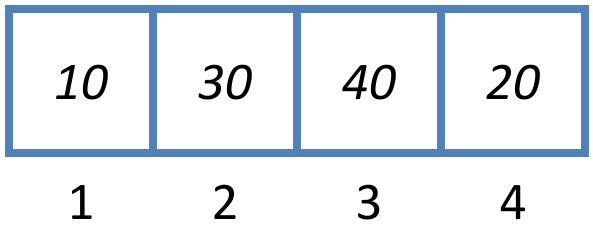


$i = 4 \quad j = 3 \quad key = 20$

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

```
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

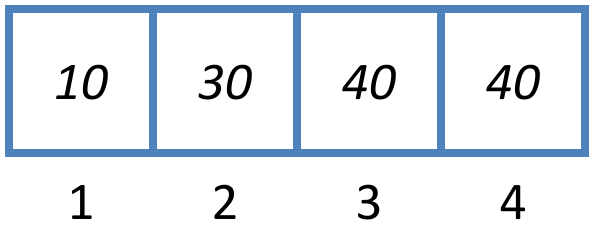


$i = 4 \quad j = 3 \quad key = 20$

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

```
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

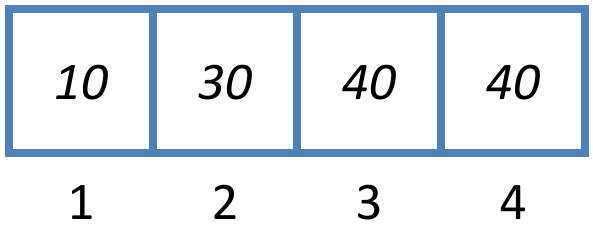


$i = 4 \quad j = 3 \quad key = 20$

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

```
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

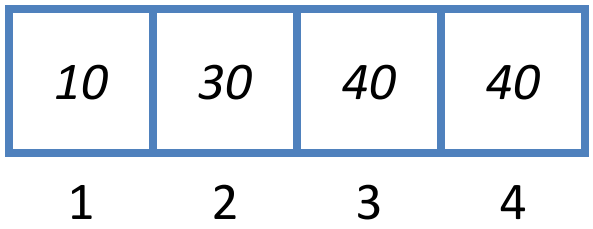


$i = 4 \quad j = 3 \quad key = 20$

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

```
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

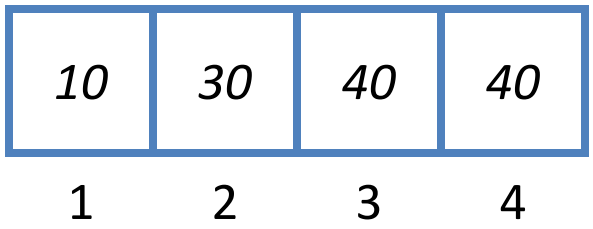


$i = 4 \quad j = 3 \quad key = 20$

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

```
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

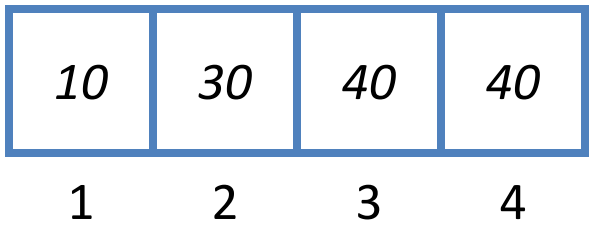


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

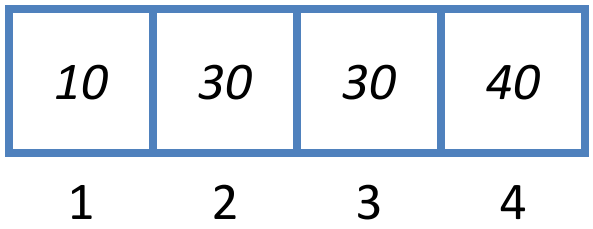


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

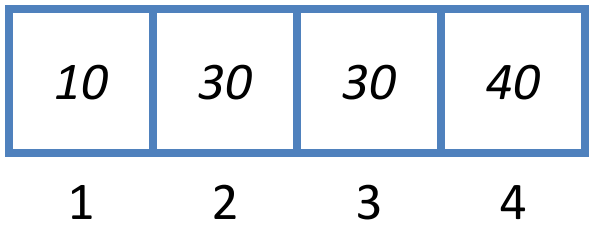


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

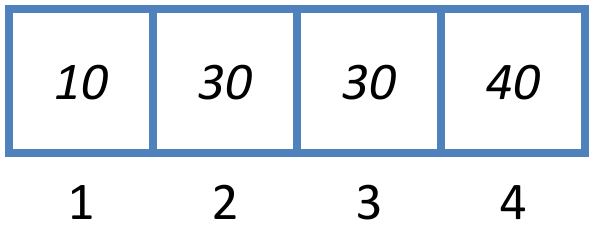


$i = 4 \quad j = 2 \quad key = 20$

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

```
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

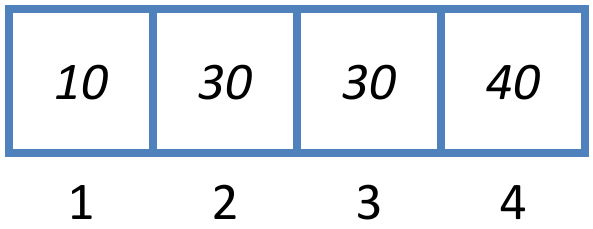


$i = 4 \quad j = 1 \quad key = 20$

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

```
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

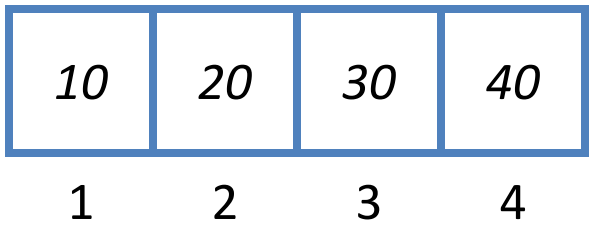


$i = 4 \quad j = 1 \quad key = 20$

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

```
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

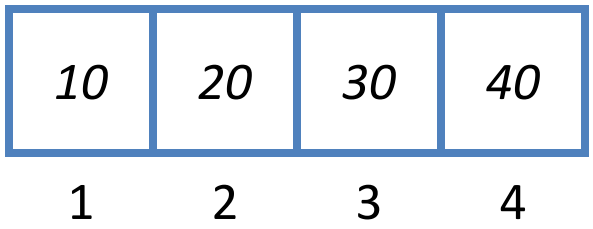


$i = 4 \quad j = 1 \quad key = 20$

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

```
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



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

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

```
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!

INSERTION-SORT

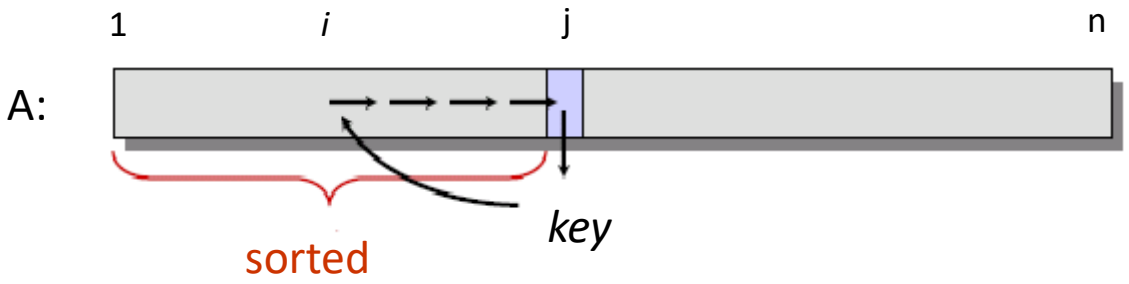
"pseudocode"

```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
    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
```

INSERTION-SORT

"pseudocode"

```
INSERTION-SORT (A, n)        ▷ A[1 . . n]
    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
```



Running time

- The running time depends on the input: an already sorted sequence is easier to sort.
- Parameterize the running time by the size of the input, since short sequences are easier to sort than long ones.
- Generally, we seek upper bounds on the running time, because everybody likes a guarantee.

Kinds of analyses

Worst-case: (usually)

- $T(n)$ =maximum time of algorithm on any input of size n.

Average-case: (sometimes)

- $T(n)$ =expected time of algorithm over all inputs of size n.
- Need assumption of statistical distribution of inputs.

Best-case:

- Cheat with a slow algorithm that works fast on *some* input.

Machine-independent time

What is insertion sort's worst-case time?

- It depends on the speed of our computer:
  - relative speed (on the same machine),
  - absolute speed (on different machines).

BIG IDEA:

- Ignore machine-dependent constants.
- Look at growth of $T(n)$ as $n \to \infty$.

"Asymptotic Analysis"

Θ-notation

Math:

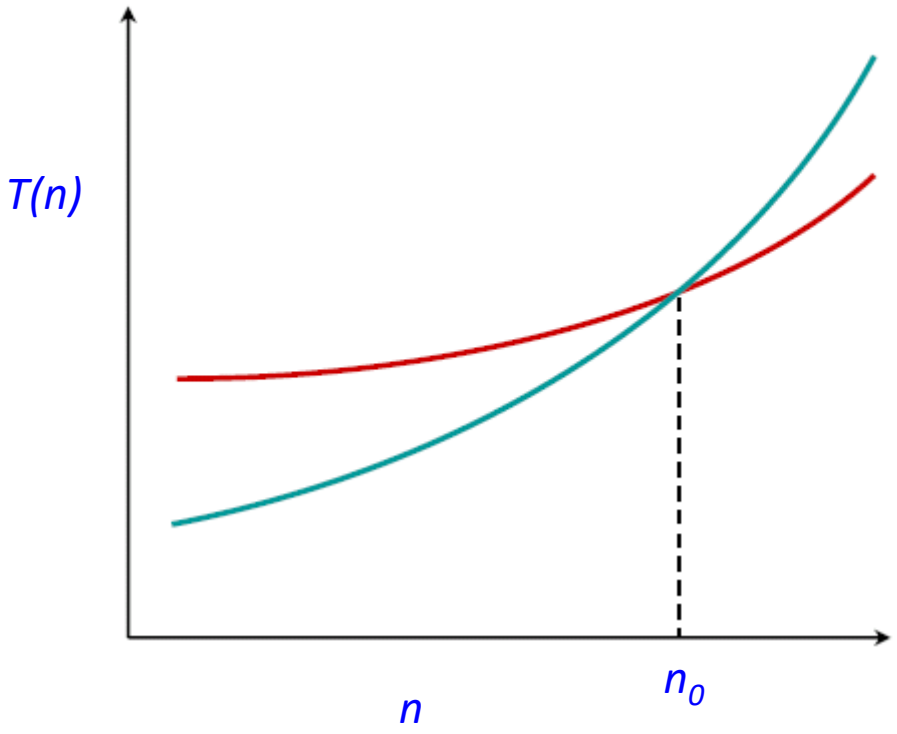
$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$

Engineering:

- Drop low-order terms; ignore leading constants.
- Example: $3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$

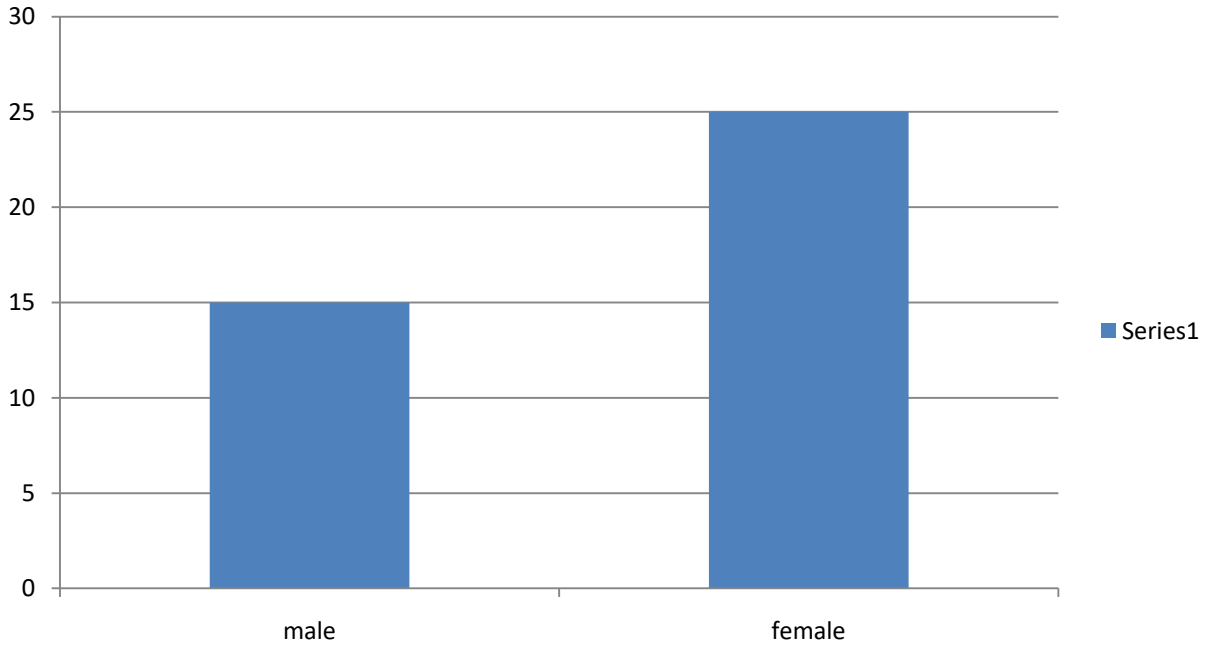
Asymptotic performance

When n gets large enough, a $\Theta(n^2)$algorithm always beats a $\Theta(n^3)$algorithm.

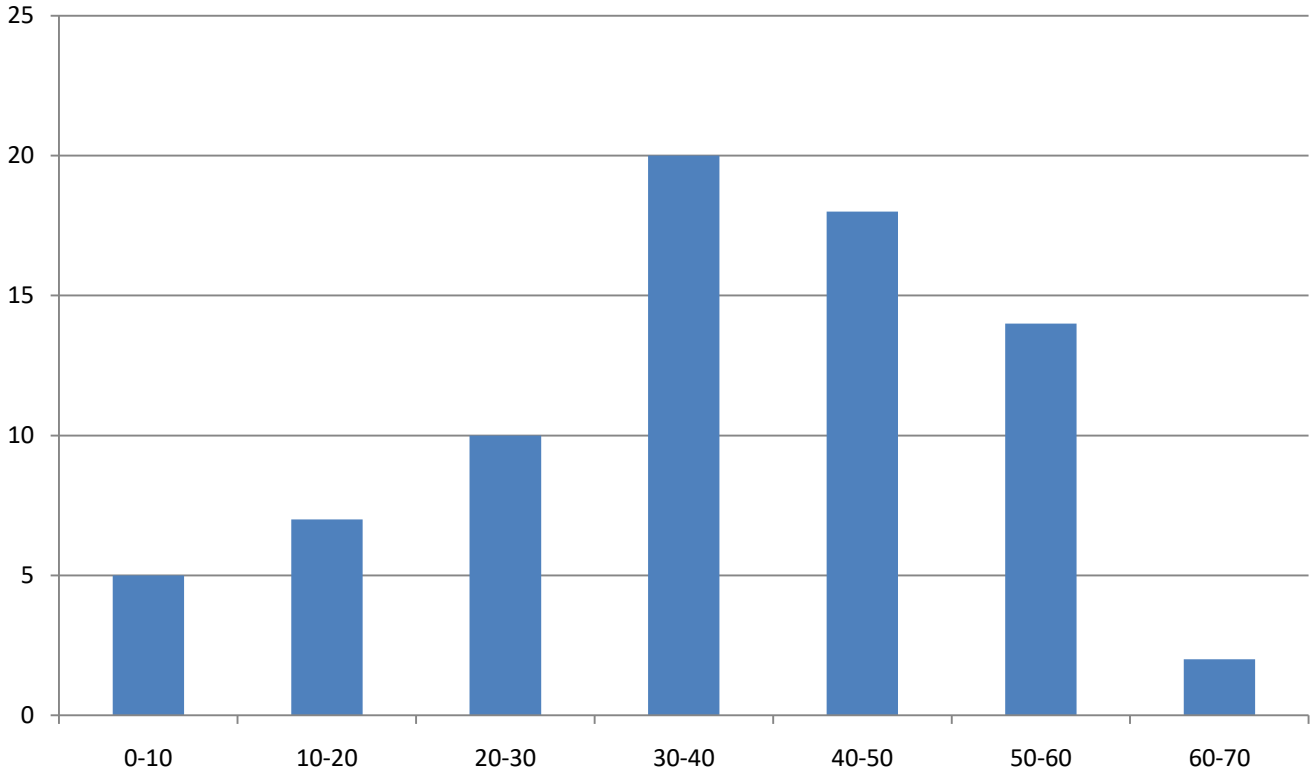


- We shouldn't ignore asymptotically slower algorithms, however.
- Real-world design situations often call for a careful balancing of engineering objectives.
- Asymptotic analysis is a useful tool to help to structure our thinking.

(Distribution)



(Distribution)



(Distribution)
Work At Home

- Work At home
  - Make the group of 2
  - Deadline: Next Session, Saturday Mehr 17th.

- What is the distribution in the nature?
- Give at least one example?

Quiz 1

- What are the correct intermediate steps of the following data set when it is being sorted with the Insertion sort? 15,20,10,18
- **A.** 15,20,10,18 -- 10,15,20,18 -- 10,15,18,20 -- 10,15,18,20
- **B.** 15,18,10,20 -- 10,18,15,20 -- 10,15,18,20 -- 10,15,18,20
- **C.** 15,10,20,18 -- 15,10,18,20 -- 10,15,18,20
- **D.** 10, 20,15,18 -- 10,15,20,18 -- 10,15,18,20

Quiz 2

- Consider the array A[]= {6,4,8,1,3} apply the insertion sort to sort the array . Consider the cost associated with each sort is 25 Rials, what is the total cost of the insertion sort when element 1 reaches the first position of the array ?

Quiz 3

- In the worst case what is the number of comparisons to order an array of 5 elements using the insertion sort?