Chapter 8

Algorithms

©Brooks/Cole, 2003

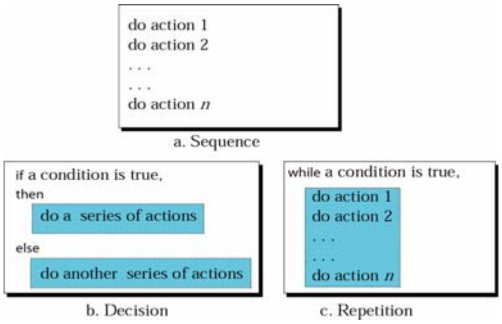
Informal Definition :: Review

- Algorithm: a step-by-step method for solving a problem or doing a task

©Brooks/Cole, 2003

Figure 8-6

Three constructs :: Review



```
do action 1
do action 2
. . .
. . .
do action n
```

a. Sequence

```
if a condition is true,
then
    do a series of actions
else
    do another series of actions
```

b. Decision

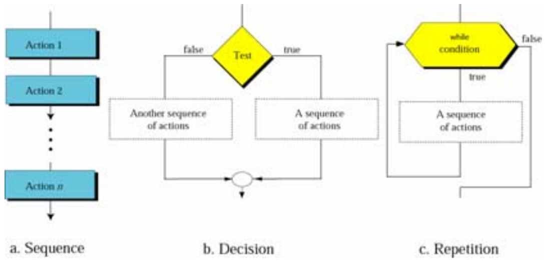
```
while a condition is true,
    do action 1
    do action 2
    . . .
    . . .
    do action n
```

c. Repetition

©Brooks/Cole, 2003

Figure 8-7

Flowcharts for three constructs :: Review



a. Sequence

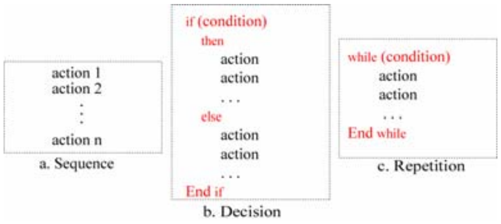
b. Decision

c. Repetition

©Brooks/Cole, 2003

Figure 8-8

Pseudocode for three constructs :: Review



```
action 1
action 2
   :
action n
```

a. Sequence

```
if (condition)
   then
      action
      action
      . . .
   else
      action
      action
      . . .
End if
```

b. Decision

```
while (condition)
      action
      action
      . . .
End while
```

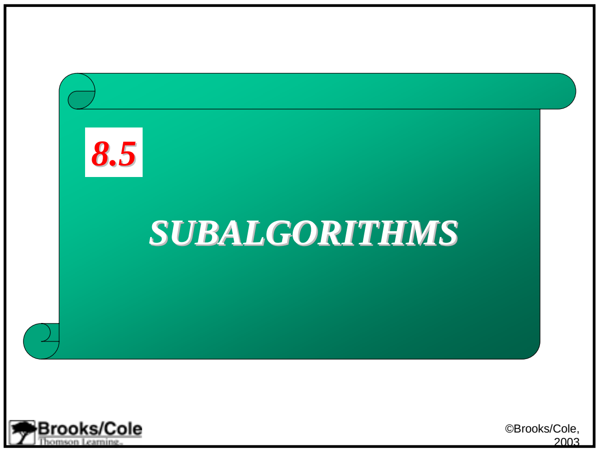
c. Repetition

©Brooks/Cole, 2003

Algorithm :: Formal Definition :: Review

An ordered set of unambiguous steps that produces a result and terminates in a finite time

©Brooks/Cole, 2003



8.5

SUBALGORITHMS

Brooks/Cole Thomson Learning ©Brooks/Cole, 2003

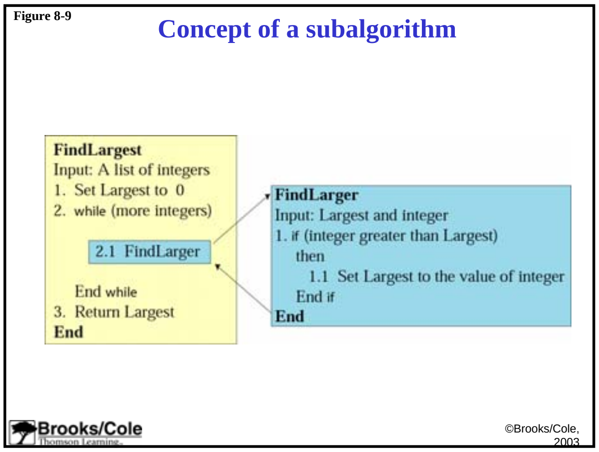


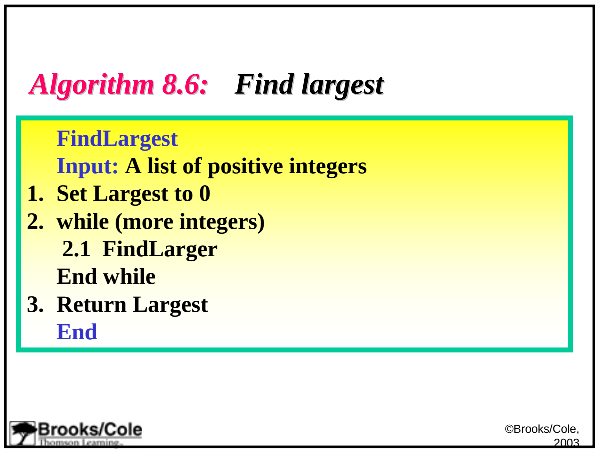
Figure 8-9

Concept of a subalgorithm

```
FindLargest
Input: A list of integers
1. Set Largest to 0
2. while (more integers)
      2.1 FindLarger
   End while
3. Return Largest
End
```

```
FindLarger
Input: Largest and integer
1. if (integer greater than Largest)
   then
      1.1 Set Largest to the value of integer
   End if
End
```

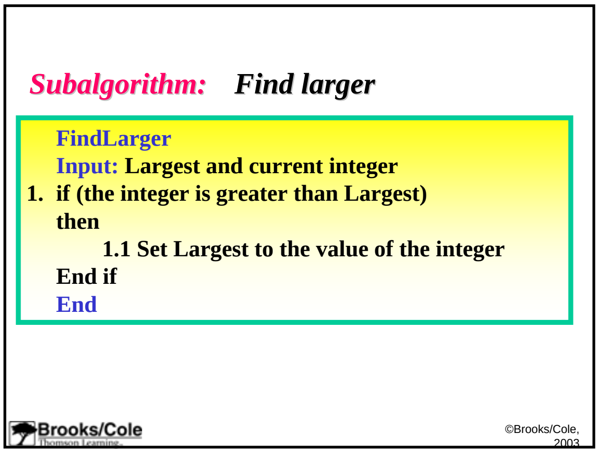
Brooks/Cole Thomson Learning ©Brooks/Cole, 2003



Algorithm 8.6: Find largest

```
   FindLargest
   Input: A list of positive integers
1. Set Largest to 0
2. while (more integers)
    2.1 FindLarger
   End while
3. Return Largest
   End
```

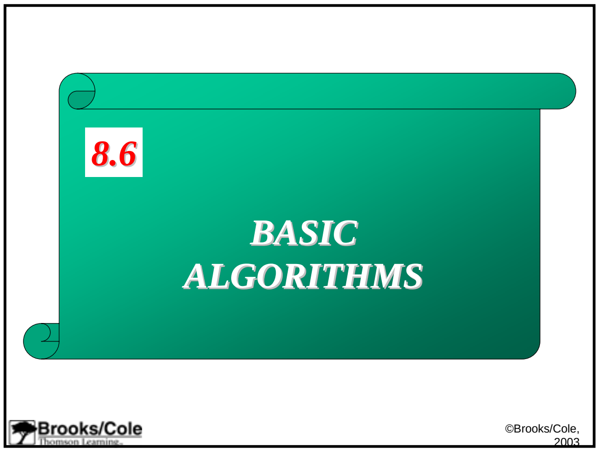
Brooks/Cole Thomson Learning ©Brooks/Cole, 2003



Subalgorithm: Find larger

```
   FindLarger
   Input: Largest and current integer
1. if (the integer is greater than Largest)
   then
         1.1 Set Largest to the value of the integer
   End if
   End
```

Brooks/Cole Thomson Learning ©Brooks/Cole, 2003



8.6

BASIC ALGORITHMS

Brooks/Cole Thomson Learning ©Brooks/Cole, 2003

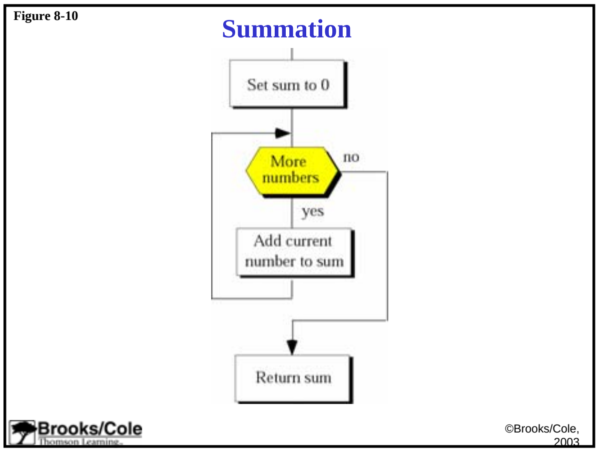


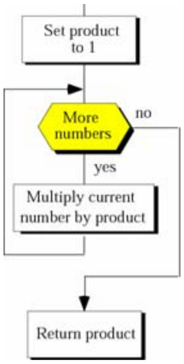
Figure 8-10

Summation

Brooks/Cole Thomson Learning ©Brooks/Cole, 2003

Figure 8-11

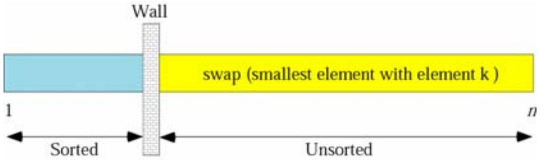
Product



©Brooks/Cole, 2003

Figure 8-12

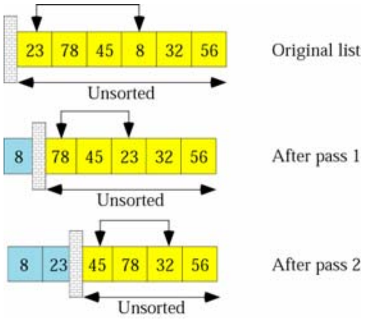
Selection sort



©Brooks/Cole, 2003

Figure 8-13: part I

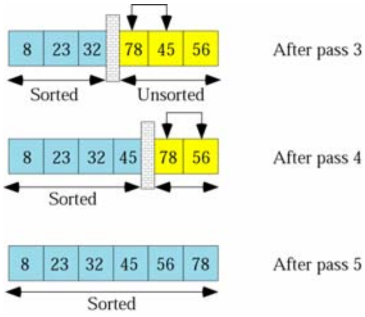
Example of selection sort



©Brooks/Cole, 2003

Figure 8-13: part II

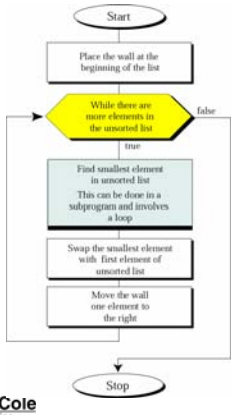
Example of selection sort



©Brooks/Cole, 2003

Figure 8-14

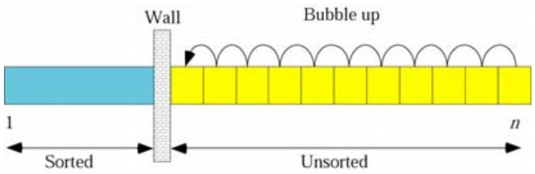
Selection sort algorithm


Start
Place the wall at the beginning of the list
While there are more elements in the unsorted list
false
true
Find smallest element in unsorted list
This can be done in a subprogram and involves a loop
Swap the smallest element with first element of unsorted list
Move the wall one element to the right
Stop

©Brooks/Cole, 2003

Figure 8-15

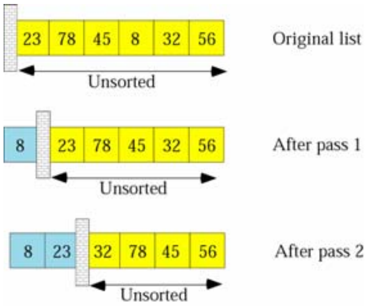
Bubble sort



©Brooks/Cole, 2003

Figure 8-16: part I

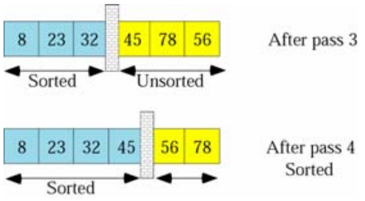
Example of bubble sort



©Brooks/Cole, 2003

Figure 8-16: part II

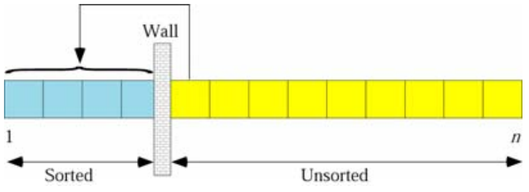
Example of bubble sort



©Brooks/Cole, 2003

Figure 8-17

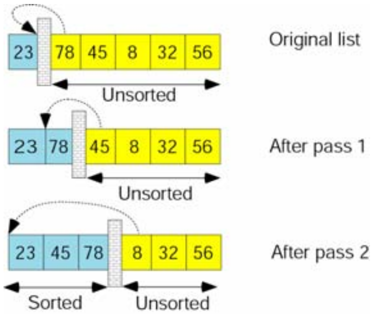
Insertion sort



©Brooks/Cole, 2003

Figure 8-18: part I

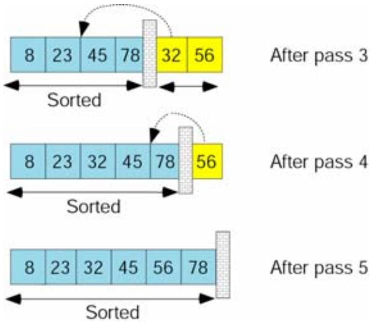
Example of insertion sort



©Brooks/Cole, 2003

Figure 8-18: part II

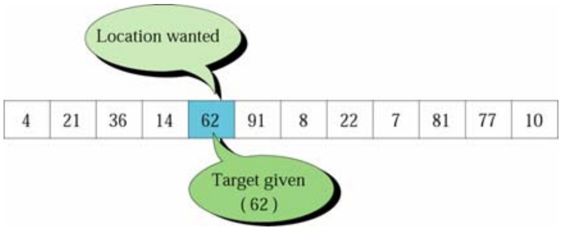
Example of insertion sort



©Brooks/Cole, 2003

Figure 8-19

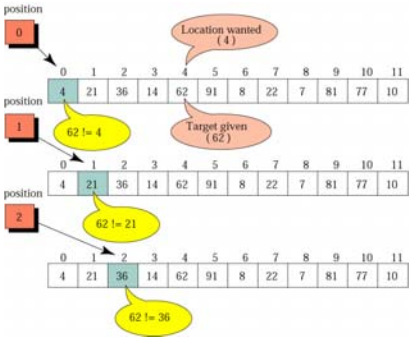
Search concept



©Brooks/Cole, 2003

Figure 8-20: Part I

Example of a sequential sort

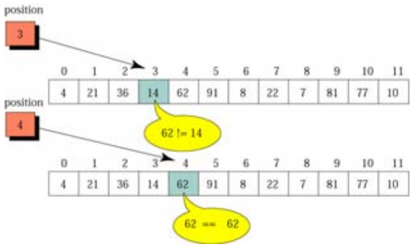


Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

Figure 8-20: Part II

Example of a sequential sort

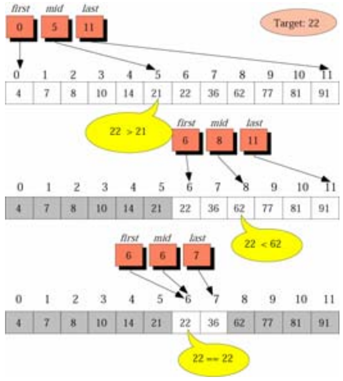


Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

Figure 8-21

Example of a binary sort



Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

8.1

RECURSION

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

Figure 8-22

Iterative definition of factorial

$$\text{Factorial }(n) = \left[\begin{array}{ll} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1 & \text{if } n > 0 \end{array}\right]$$

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

Figure 8-23

Recursive definition of factorial

$$\text{Factorial }(n) = \left[\begin{array}{ll} 1 & \text{if } n = 0 \\ n \times \text{Factorial }(n-1) & \text{if } n > 0 \end{array}\right]$$

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

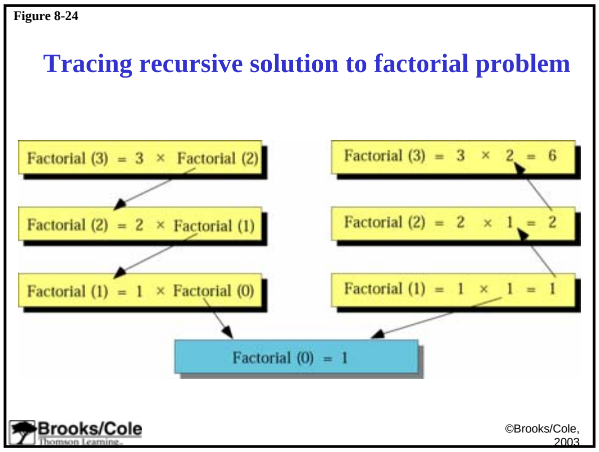


Figure 8-24

Tracing recursive solution to factorial problem

Factorial (3) = 3 × Factorial (2)

Factorial (2) = 2 × Factorial (1)

Factorial (1) = 1 × Factorial (0)

Factorial (0) = 1

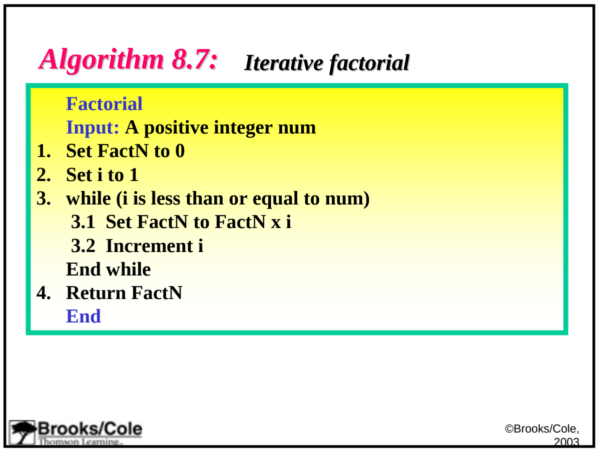
Factorial (1) = 1 × 1 = 1

Factorial (2) = 2 × 1 = 2

Factorial (3) = 3 × 2 = 6

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

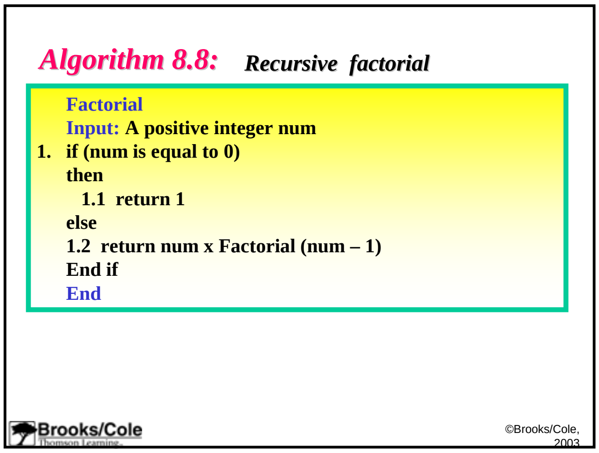


Algorithm 8.7: *Iterative factorial*

```
     Factorial
     Input: A positive integer num
1.   Set FactN to 0
2.   Set i to 1
3.   while (i is less than or equal to num)
       3.1  Set FactN to FactN x i
       3.2  Increment i
     End while
4.   Return FactN
     End
```

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

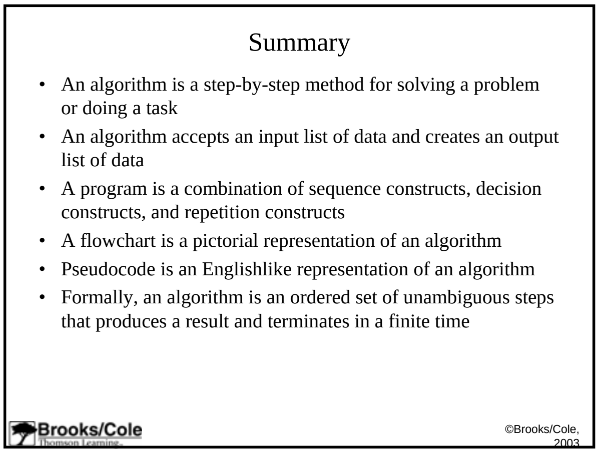


Algorithm 8.8: *Recursive factorial*

```
     Factorial
     Input: A positive integer num
1.   if (num is equal to 0)
     then
       1.1  return 1
     else
     1.2  return num x Factorial (num – 1)
     End if
     End
```

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

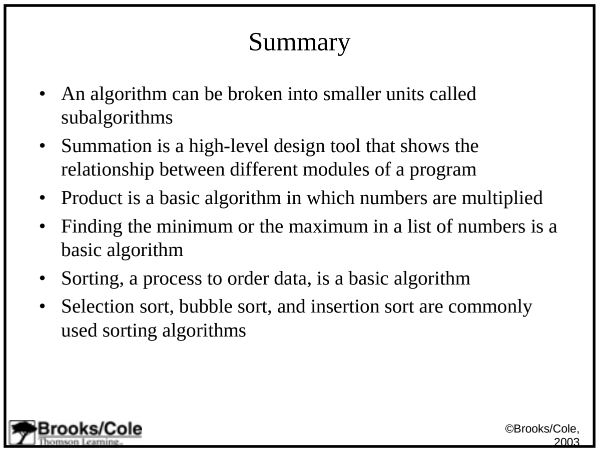


Summary

- An algorithm is a step-by-step method for solving a problem or doing a task
- An algorithm accepts an input list of data and creates an output list of data
- A program is a combination of sequence constructs, decision constructs, and repetition constructs
- A flowchart is a pictorial representation of an algorithm
- Pseudocode is an Englishlike representation of an algorithm
- Formally, an algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003

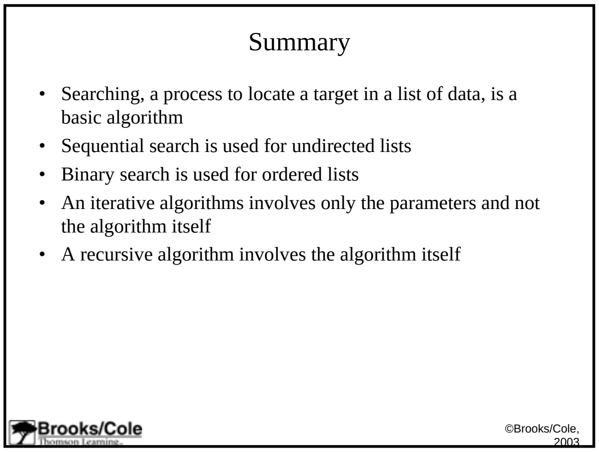


Summary

- An algorithm can be broken into smaller units called subalgorithms
- Summation is a high-level design tool that shows the relationship between different modules of a program
- Product is a basic algorithm in which numbers are multiplied
- Finding the minimum or the maximum in a list of numbers is a basic algorithm
- Sorting, a process to order data, is a basic algorithm
- Selection sort, bubble sort, and insertion sort are commonly used sorting algorithms

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003



Summary

- Searching, a process to locate a target in a list of data, is a basic algorithm
- Sequential search is used for undirected lists
- Binary search is used for ordered lists
- An iterative algorithms involves only the parameters and not the algorithm itself
- A recursive algorithm involves the algorithm itself

Brooks/Cole Thomson Learning

©Brooks/Cole, 2003