Chapter 8

Algorithms

OBJECTIVES

After reading this chapter, the reader should be able to:

- Understand the concept of an algorithm.
- Define and use the three constructs for developing algorithms: sequence, decision, and repetition.
- Understand and use three tools to represent algorithms: flowchart, pseudocode, and structure chart.
- Understand the concept of modularity and subalgorithms.
- List and comprehend common algorithms.

8.1

CONCEPT

Figure 8-1

Informal definition of an algorithm used in a computer

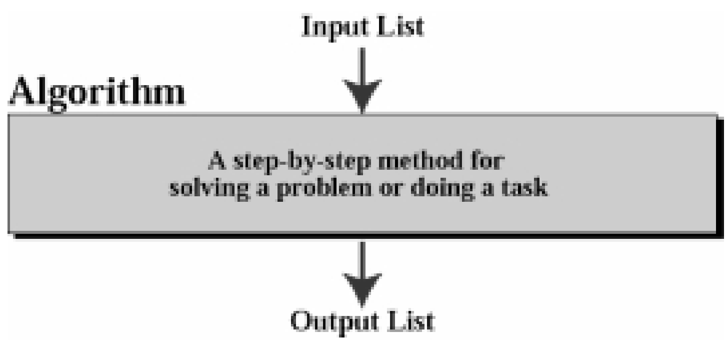


Figure 8-2

Finding the largest integer among five integers

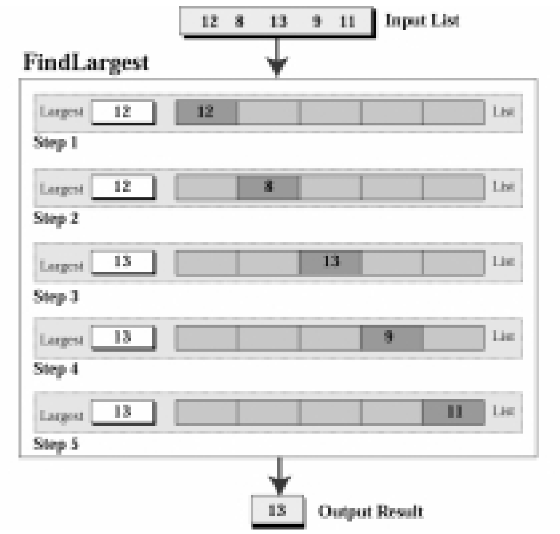


Figure 8-3

Defining actions in FindLargest algorithm

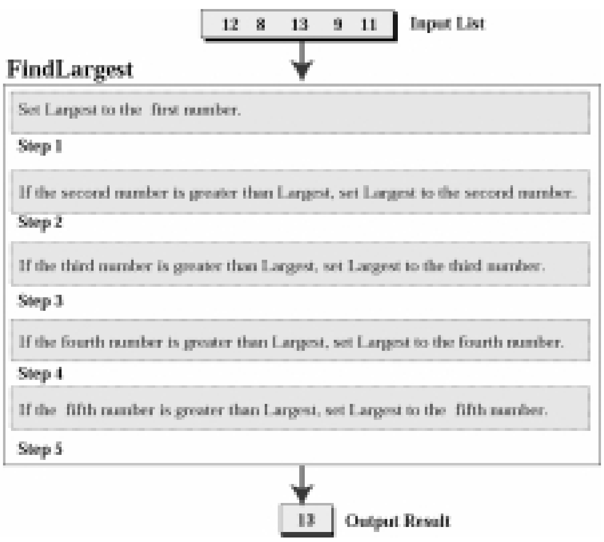


Figure 8-4

FindLargest refined

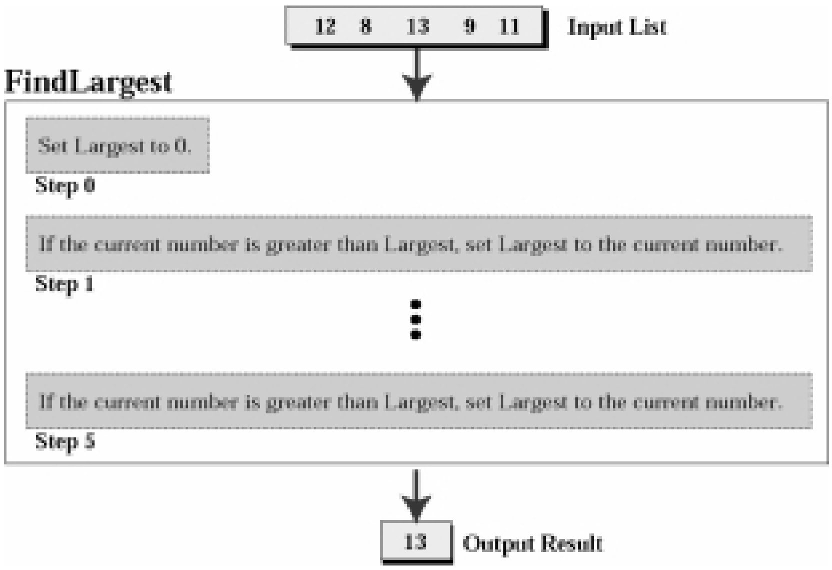
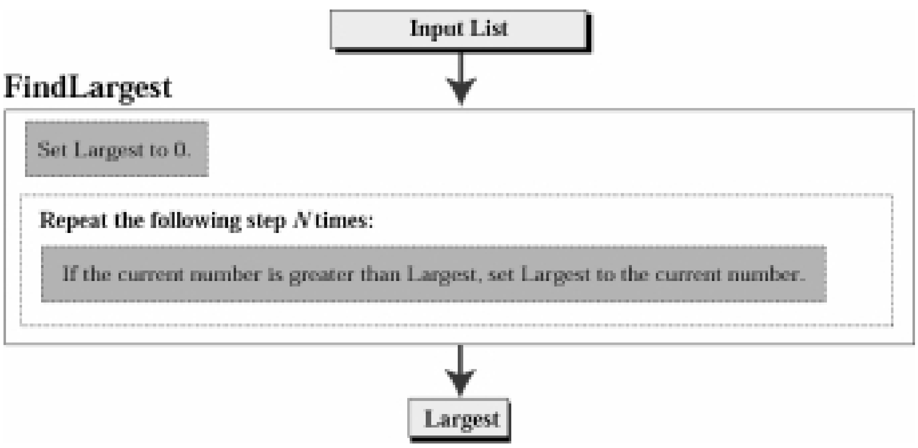


Figure 8-5

Generalization of FindLargest

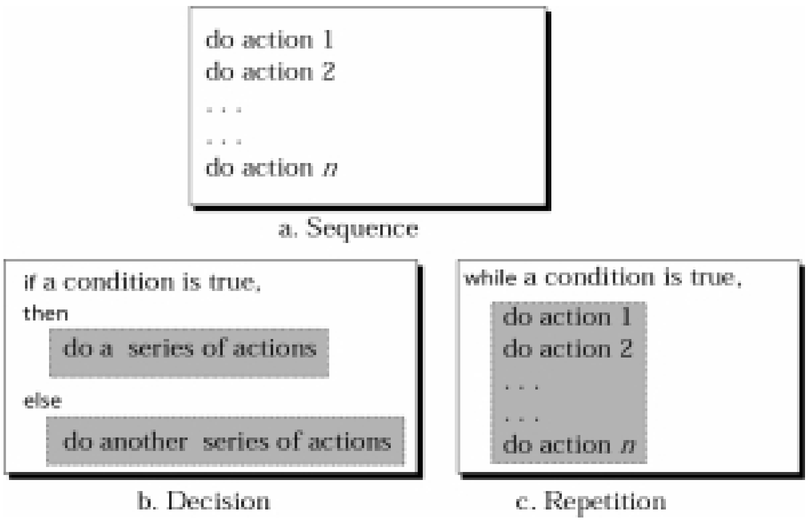


8.2

THREE CONSTRUCTS

Figure 8-6

Three constructs

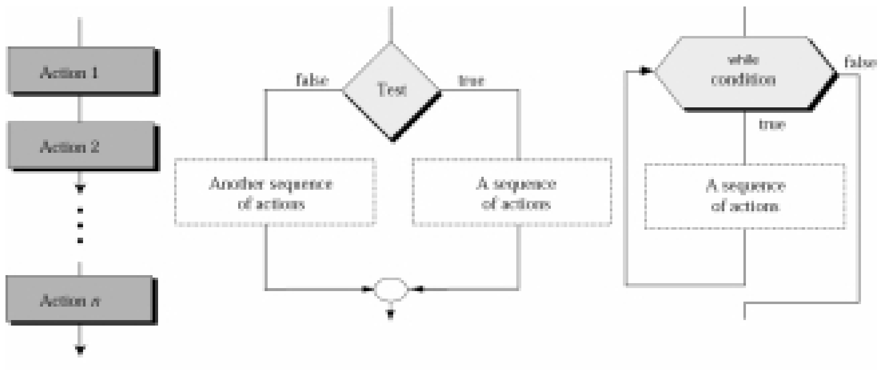


8.3

ALGORITHM REPRESENTATION

Figure 8-7

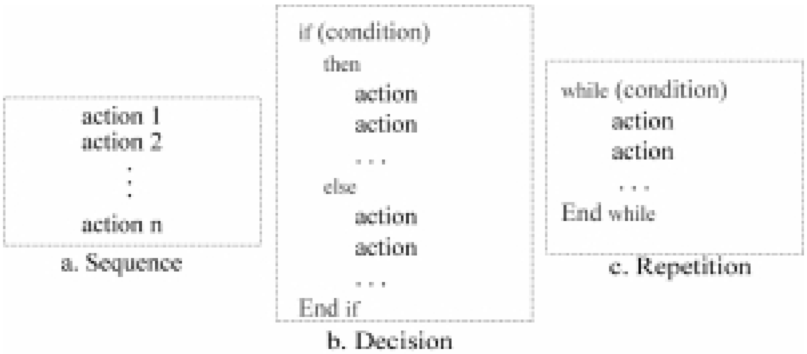
Flowcharts for three constructs



a. Sequence b. Decision c. Repetition

Figure 8-8

Pseudocode for three constructs



Example 1

Write an algorithm in pseudocode that finds the average of two numbers

Solution

See Algorithm 8.1 on the next slide.

Sequence

Algorithm 8.1: Average of two

AverageOfTwo
Input: Two numbers
1. **Add the two numbers**
2. **Divide the result by 2**
3. **Return the result by step 2**

End

Example 2

Write an algorithm to change a numeric grade to a pass/no pass grade.

Solution

See Algorithm 8.2 on the next slide.

Decision

Algorithm 8.2: Pass/no pass Grade

```
    Pass/NoPassGrade
    Input: One number
1.  if (the number is greater than or equal to 70)
    then
       1.1  Set the grade to "pass"
    else
       1.2   Set the grade to "nopass"
    End if
2.  Return the grade
    End
```

Example 3

Write an algorithm to change a numeric grade to a letter grade.

Solution

See Algorithm 8.3 on the next slide.

(Multiple decision)

Algorithm 8.3: Letter grade

LetterGrade
Input: One number

1. if (the number is between 90 and 100, inclusive) then
   1.1 Set the grade to “A”
   End if
2. if (the number is between 80 and 89, inclusive) then
   2.1 Set the grade to “B”
   End if

Continues on the next slide

Algorithm 8.3: Letter grade (continued)

3. if (the number is between 70 and 79, inclusive) then
   3.1 Set the grade to “C”
   End if
4. if (the number is between 60 and 69, inclusive) then
   4.1 Set the grade to “D”
   End if

Continues on the next slide

Algorithm 8.3: Letter grade (continued)

5. **If (the number is less than 60)**
   then
   5.1 Set the grade to "F"
   End if
6. **Return the grade**
   End

Example 4

Write an algorithm to find the largest of a set of numbers. You do not know the number of numbers.

Solution

See Algorithm 8.4 on the next slide.

Repetition + Decision

Algorithm 8.4: Find largest

```
     FindLargest
     Input: A list of positive integers
1.   Set Largest to 0
2.   while (more integers)
      2.1  if (the integer is greater than Largest)
             then
               2.1.1  Set largest to the value of the integer
            End if
     End while
3.   Return Largest
     End
```

Example 5

Write an algorithm to find the largest of 1000 numbers.

Solution

See Algorithm 8.5 on the next slide.

Add a counter to the repetition structure

Algorithm 8.5: *Find largest of 1000 numbers*

```
    FindLargest
    Input: 1000 positive integers
1.  Set Largest to 0
2.  Set Counter to 0
3.  while (Counter less than 1000)
     3.1  if (the integer is greater than Largest)
             then
               3.1.1  Set Largest to the value of the integer
            End if
      3.2  Increment Counter
    End while
4.  Return Largest
    End
```

In Class Exercise

- Write the pseudo code to get the second largest number from the input.

8.4

FORMAL DEFINITION

- ***Ordered set***
- ***Unambiguous steps***
- ***Effectiveness***
- ***Termination***

Definition of Algorithm

- An algorithm is an ordered set of unambiguous steps that produce results and halts in finite time.
- Ordered set: Each instruction is well-defined.
- Unambiguous steps: Each step is certain
- Produce a result: Effectiveness
- Termination in finite time
  - Infinite loop
  - Unsolvable program : Halting problem

8.5

SUBALGORITHMS

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
```

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
```

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
```

8.6

BASIC ALGORITHMS

Figure 8-10

Summation

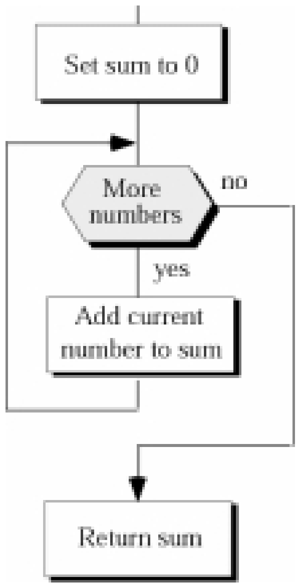


Figure 8-11

Product

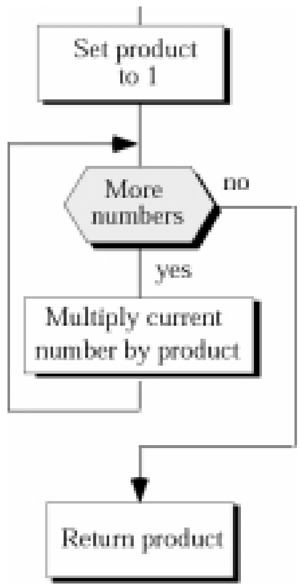


Figure 8-12

Selection sort

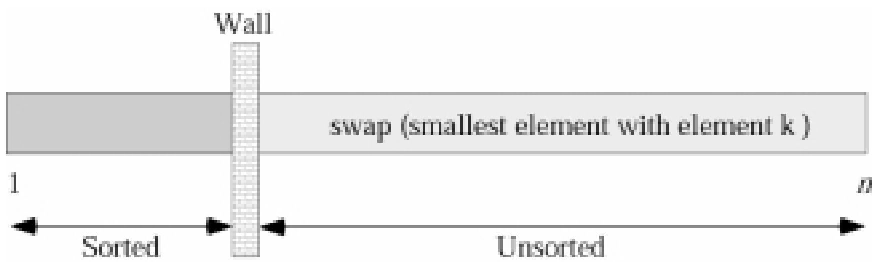


Figure 8-13: part I

Example of selection sort

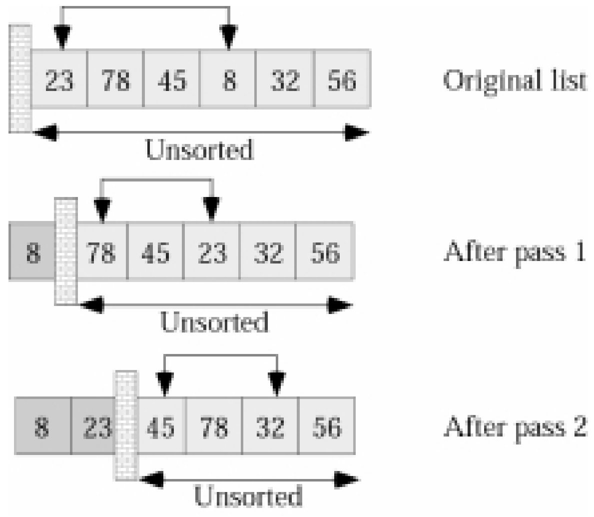


Figure 8-13: part II

Example of selection sort

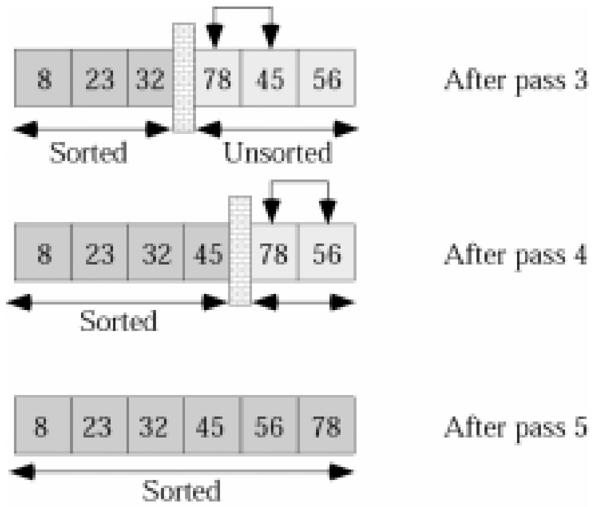
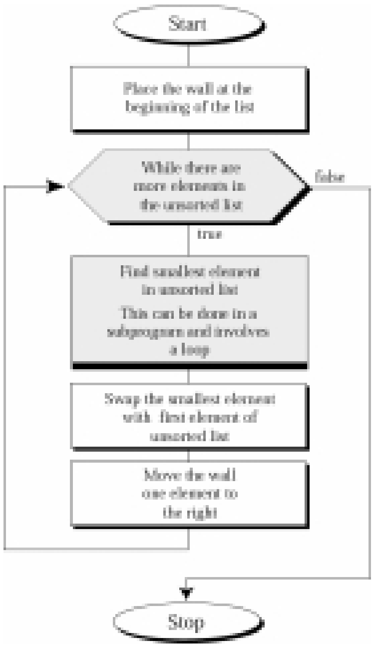


Figure 8-14



Selection sort algorithm

In Class Exercise: Sort

- Sort 4,2,7,6,3,1,9,8

Figure 8-15

Bubble sort

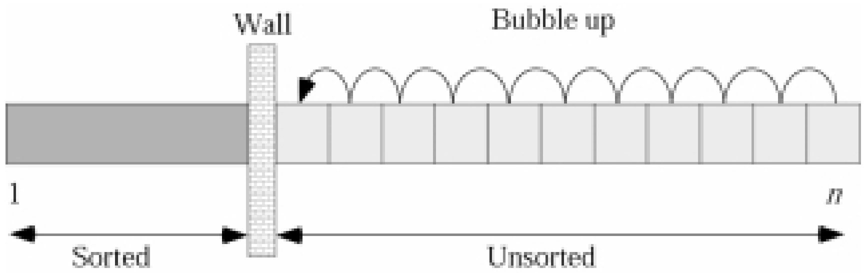


Figure 8-16: part I

Example of bubble sort

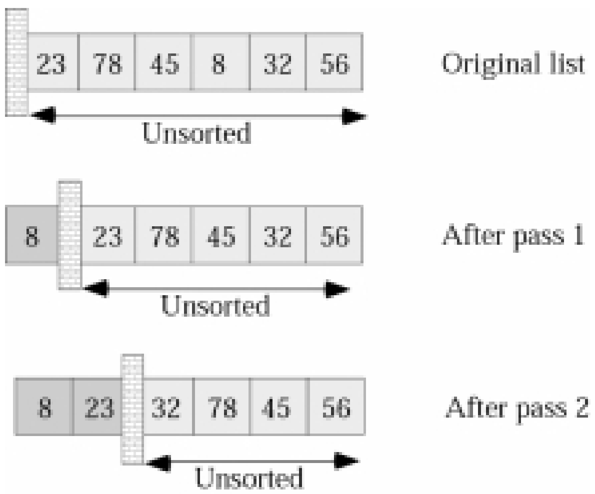


Figure 8-16: part II

Example of bubble sort

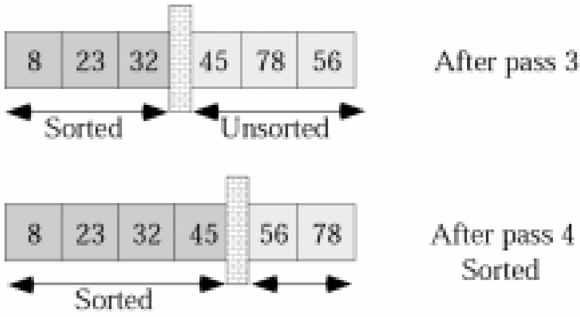


Figure 8-17

Insertion sort

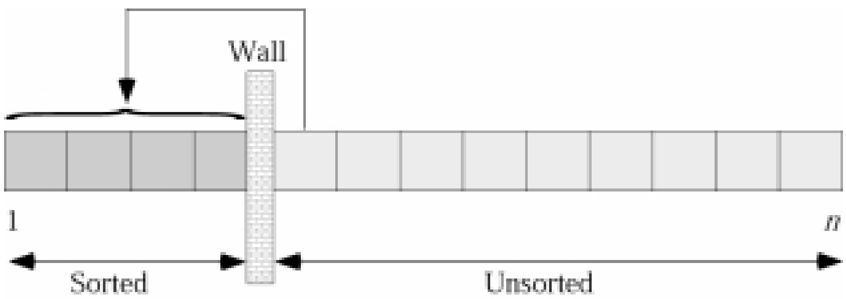


Figure 8-18: part I

Example of insertion sort

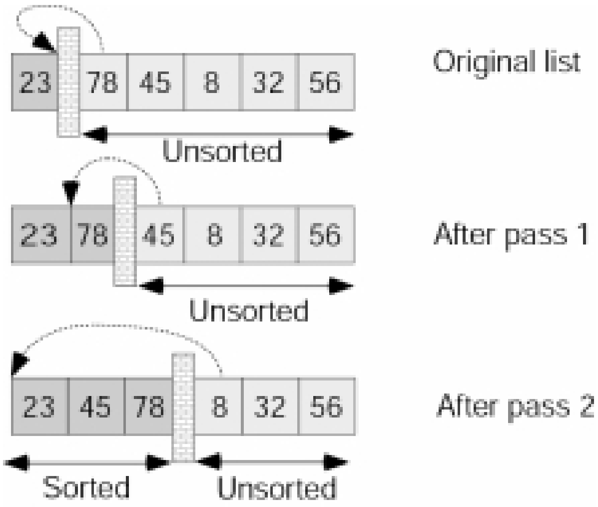


Figure 8-18: part II

Example of insertion sort

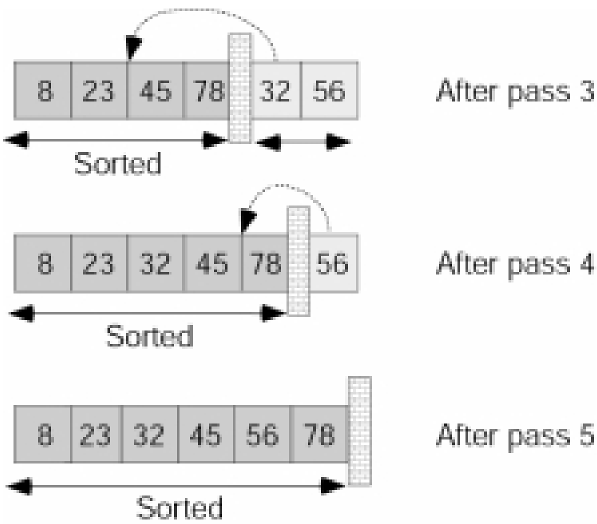


Figure 8-19

Search concept

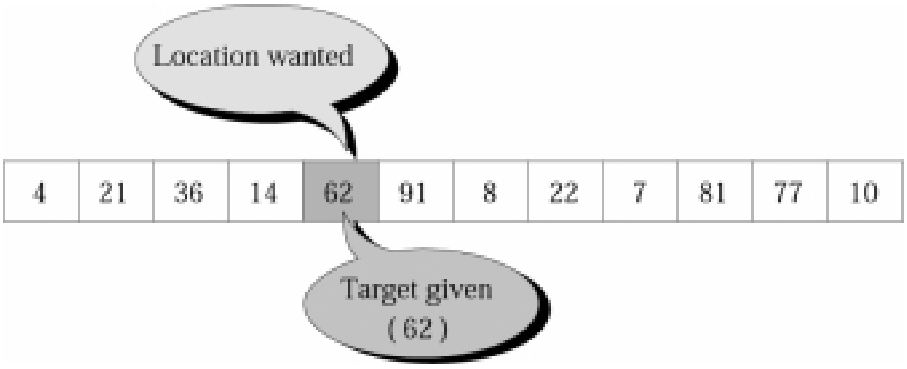


Figure 8-20: Part I

Example of a sequential search

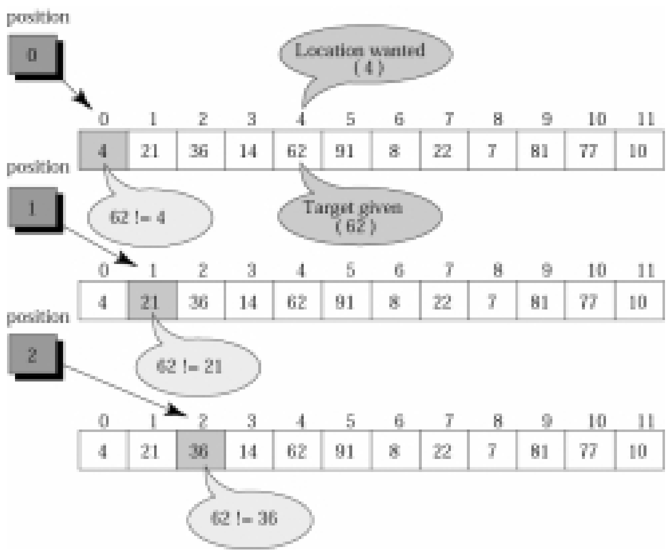


Figure 8-20: Part II

Example of a sequential search

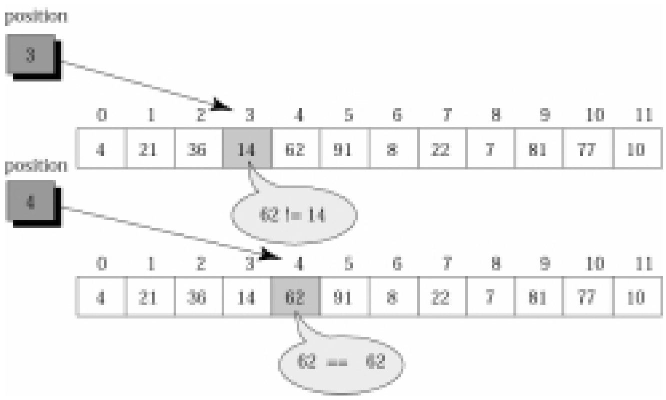
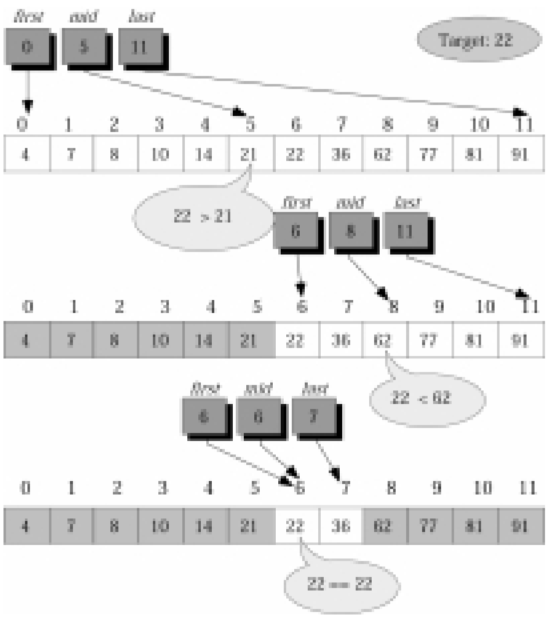


Figure 8-21

Example of a binary search



8.1

RECURSION

Figure 8-22

Iterative definition of factorial

$$\text{Factorial}(n) = \begin{bmatrix} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1 & \text{if } n > 0 \end{bmatrix}$$

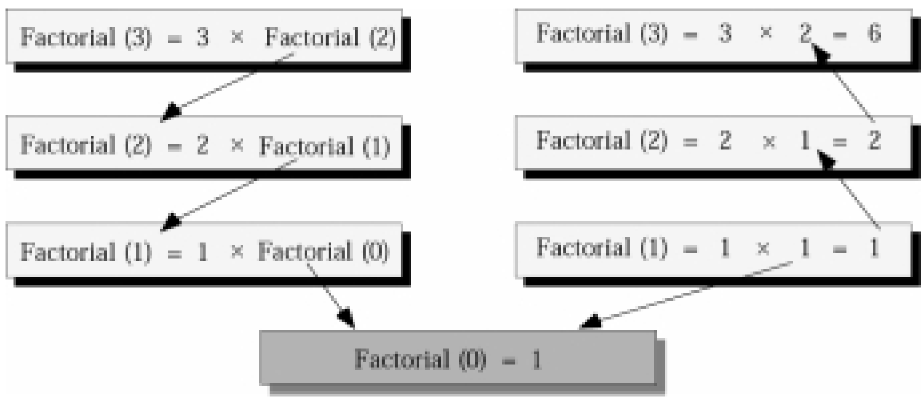
Figure 8-23

Recursive definition of factorial

$$\text{Factorial}(n) = \begin{bmatrix} 1 & \text{if } n = 0 \\ n \times \text{Factorial}(n-1) & \text{if } n > 0 \end{bmatrix}$$

Figure 8-24

Tracing recursive solution to factorial problem



Algorithm 8.7: Iterative factorial

```
    Factorial
    Input: A positive integer num
1.  Set FactN to 1
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
```

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
```