Chapter 5: Algorithms

5.1 The Concept of an Algorithm
5.2 Algorithm Representation
5.3 Algorithm Discovery
5.4 Iterative Structures
5.5 Recursive Structures
5.6 Efficiency and Correctness
Definition of Algorithm

An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process.

Algorithm Representation

- Requires well-defined primitives
- A collection of primitives constitutes a programming language.
Figure 5.2  Folding a bird from a square piece of paper

Figure 5.3  Origami primitives
Pseudocode Primitives

• Assignment

   \[ name \leftarrow expression \]

• Conditional selection

   \[ \text{if } \text{condition} \text{ then } \text{action} \]

Pseudocode Primitives (continued)

• Repeated execution

   \[ \text{while } \text{condition} \text{ do } \text{activity} \]

• Procedure

   \[ \text{procedure } name (\text{generic names}) \]
Figure 5.4  The procedure Greetings in pseudocode

**procedure** Greetings
Count ← 3;
**while** (Count > 0) **do**
  (print the message “Hello” and Count ← Count – 1)

Polya’s Problem Solving Steps

• 1. Understand the problem.
• 2. Devise a plan for solving the problem.
• 3. Carry out the plan.
• 4. Evaluate the solution for accuracy and its potential as a tool for solving other problems.
Getting a Foot in the Door

• Try working the problem backwards
• Solve an easier related problem
  – Relax some of the problem constraints
  – Solve pieces of the problem first (bottom up methodology)
• Stepwise refinement: Divide the problem into smaller problems (top-down methodology)

Ages of Children Problem

• Person A is charged with the task of determining the ages of B’s three children.
  – B tells A that the product of the children’s ages is 36.
  – A replies that another clue is required.
  – B tells A the sum of the children’s ages.
  – A replies that another clue is needed.
  – B tells A that the oldest child plays the piano.
  – A tells B the ages of the three children.
• How old are the three children?
Iterative Structures

- Pretest loop:
  \[ \text{while (condition) do} \]
  \[ \text{(loop body)} \]
- Posttest loop:
  \[ \text{repeat (loop body)} \]
  \[ \text{until (condition)} \]
Figure 5.6  **The sequential search algorithm in pseudocode**

```pseudocode
procedure Search (List, TargetValue)
  if (List empty)
    then (Declare search a failure)
  else
    (Select the first entry in List to be TestEntry:
      while (TargetValue > TestEntry and there remain entries to be considered)
        do (Select the next entry in List as TestEntry);
      if (TargetValue = TestEntry)
        then (Declare search a success.)
      else (Declare search a failure.)
    ) end if
```

Copyright © 2012 Pearson Education, Inc.

Figure 5.7  **Components of repetitive control**

**Initialize:** Establish an initial state that will be modified toward the termination condition

**Test:** Compare the current state to the termination condition and terminate the repetition if equal

**Modify:** Change the state in such a way that it moves toward the termination condition

Copyright © 2012 Pearson Education, Inc.
Figure 5.8  The while loop structure

Test condition

Condition true

Activity

Condition false

Figure 5.9  The repeat loop structure

Activity

Condition false

Test condition

Condition true
Figure 5.10 Sorting the list Fred, Alex, Diana, Byron, and Carol alphabetically

Figure 5.11 The insertion sort algorithm expressed in pseudocode

```plaintext
procedure Sort (List)
    N 2
    while (the value of N does not exceed the length of List) do
        (Select the Nth entry in List as the pivot entry:
            Move the pivot entry to a temporary location leaving a hole in List;
            while (there is a name above the hole and that name is greater than the pivot) do
                (move the name above the hole down into the hole leaving a hole above the name)
                Move the pivot entry into the hole in List;
                N  N + 1
        )
```
Recursion

- The execution of a procedure leads to another execution of the procedure.
- Multiple activations of the procedure are formed, all but one of which are waiting for other activations to complete.

Figure 5.12 Applying our strategy to search a list for the entry John

<table>
<thead>
<tr>
<th>Original list</th>
<th>First sublist</th>
<th>Second sublist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice, Bob, Carol, David, Elaine, Fred, George, Harry, Irene, John, Kelly, Larry, Mary, Nancy, Oliver</td>
<td>Irene, John, Kelly</td>
<td>Irene, John, Kelly</td>
</tr>
</tbody>
</table>

Copyright © 2012 Pearson Education, Inc.
Figure 5.13  A first draft of the binary search technique

if (List empty)
    then
        (Report that the search failed.)
else
    {Select the "middle" entry in the List to be the TestEntry;
     Execute the block of instructions below that is
     associated with the appropriate case.
     case 1: TargetValue = TestEntry
             (Report that the search succeeded.)
     case 2: TargetValue < TestEntry
             (Search the portion of List preceding TestEntry for
              TargetValue, and report the result of that search.)
     case 3: TargetValue > TestEntry
             (Search the portion of List following TestEntry for
              TargetValue, and report the result of that search.)
    } end if

Figure 5.14  The binary search algorithm in pseudocode

procedure Search (List, TargetValue)
if (List empty)
    then
        (Report that the search failed.)
else
    {Select the "middle" entry in List to be the TestEntry;
     Execute the block of instructions below that is
     associated with the appropriate case.
     case 1: TargetValue = TestEntry
             (Report that the search succeeded.)
     case 2: TargetValue < TestEntry
             (Apply the procedure Search to see if TargetValue
              is in the portion of the List preceding TestEntry,
              and report the result of that search.)
     case 3: TargetValue > TestEntry
             (Apply the procedure Search to see if TargetValue
              is in the portion of List following TestEntry,
              and report the result of that search.)
    } end if
Algorithm Efficiency

- Measured as number of instructions executed
- Big theta notation: Used to represent efficiency classes
  - Example: Insertion sort is in Θ(n²)
- Best, worst, and average case analysis
Figure 5.18  Applying the insertion sort in a worst-case situation

<table>
<thead>
<tr>
<th>Initial list</th>
<th>1st pivot</th>
<th>2nd pivot</th>
<th>3rd pivot</th>
<th>4th pivot</th>
<th>Sorted list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaine</td>
<td>Elaine</td>
<td>David</td>
<td>Carol</td>
<td>Barbara</td>
<td>Alfred</td>
</tr>
<tr>
<td>David</td>
<td>David</td>
<td>Elaine</td>
<td>David</td>
<td>Elaine</td>
<td>Alfred</td>
</tr>
<tr>
<td>Carol</td>
<td>Carol</td>
<td>Barbara</td>
<td>David</td>
<td>Elaine</td>
<td>Alfred</td>
</tr>
<tr>
<td>Barbara</td>
<td>Barbara</td>
<td>Alfred</td>
<td>David</td>
<td>Elaine</td>
<td>Alfred</td>
</tr>
<tr>
<td>Alfred</td>
<td>Alfred</td>
<td>Barbara</td>
<td>David</td>
<td>Elaine</td>
<td>Alfred</td>
</tr>
</tbody>
</table>

Figure 5.19  Graph of the worst-case analysis of the insertion sort algorithm
Software Verification

- Proof of correctness
  - Assertions
    - Preconditions
    - Loop invariants
- Testing
Chain Separating Problem

- A traveler has a gold chain of seven links.
- He must stay at an isolated hotel for seven nights.
- The rent each night consists of one link from the chain.
- What is the fewest number of links that must be cut so that the traveler can pay the hotel one link of the chain each morning without paying for lodging in advance?

Figure 5.21 Separating the chain using only three cuts

Copyright © 2012 Pearson Education, Inc.
Figure 5.22  Solving the problem with only one cut

Figure 5.23  The assertions associated with a typical while structure