CMPS1134
Fundamentals of Computing

Algorithms 2

Computer Science: An Overview
Eleventh Edition
J. Glenn Brookshear
Chapter 5

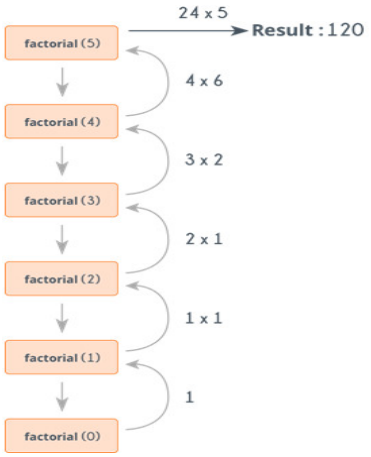
Chapter 5: Algorithms

- Recursive Structures
- Efficiency and Correctness

Recursion

- An alternative to the loop paradigm for implementing the repetition of activities.
- It involves repeating a set of instructions as a subset of itself:
  - The execution of a procedure leads to another execution of the same procedure.
  - Multiple activations of the procedure are formed, all but one of which are waiting for other activations to complete.

```
factorial(n)
   if(n > 1)
       then return n * factorial(n - 1)
       else return 1
```

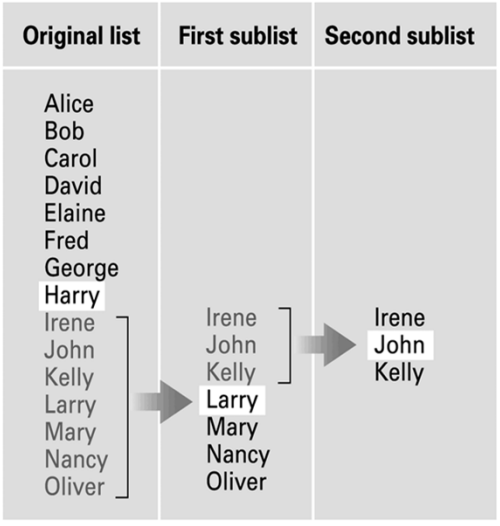


 3

Binary Search: search a list for John

Searching a directory for a person:

1. Find the "middle value" of the list being searched
2. If the "middle" entry" is equal to the target value declare success
3. Otherwise we continue the search (go to step 1) in first or last half, depending on whether the target value is less than or greater than the "middle entry"



 4

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
```

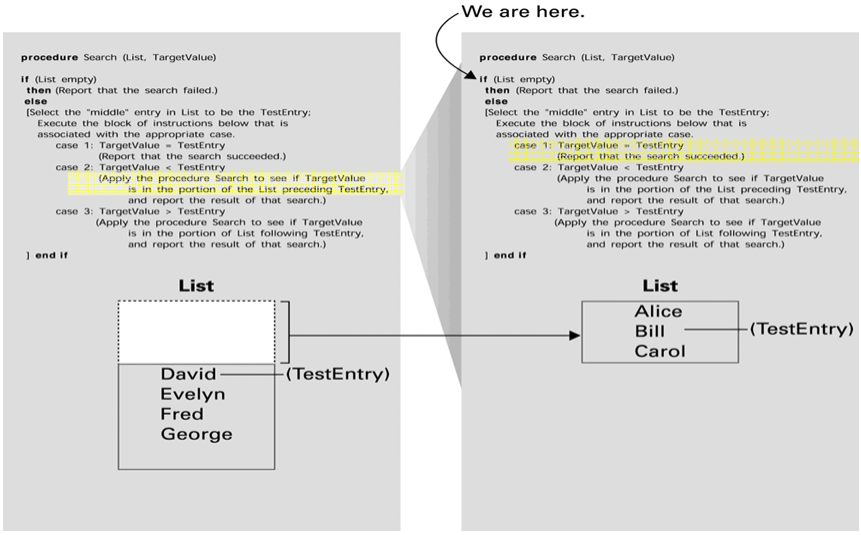
 5

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
```

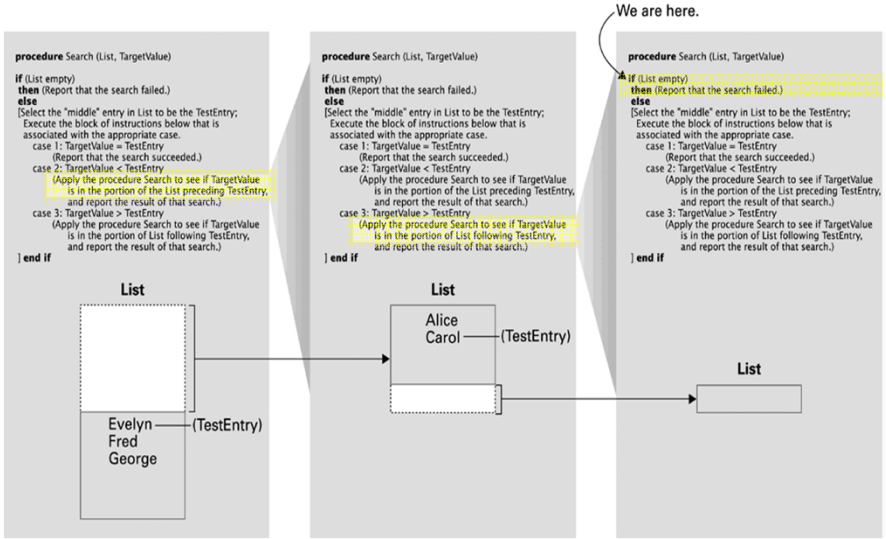
 6

Figure 5.15 (search for Bill)



 7

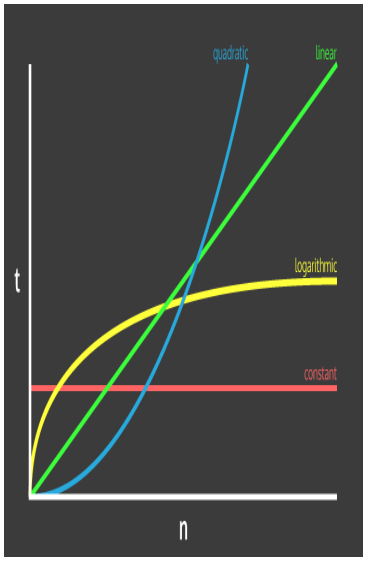
Figure 5.17 (search for David)



 8

Algorithm Efficiency

- Measured as number of instructions executed
- **Big theta** notation: Used to represent efficiency classes
  - Examples:
    - Insertion sort: $\Theta(n^2)$
    - Binary search: $\Theta(\lg n)$
- Best, worst, and average case analysis



 9

Figure 5.18 **Applying the insertion sort in a worst-case situation**

Initial list	Comparisons made for each pivot				Sorted list
	1st pivot	2nd pivot	3rd pivot	4th pivot	
Elaine	1 Elaine	3 David	6 Carol	10 Barbara	Alfred
David	David	2 Elaine	5 David	9 Carol	Barbara
Carol	Carol	Carol	4 Elaine	8 David	Carol
Barbara	Barbara	Barbara	Barbara	7 Elaine	David
Alfred	Alfred	Alfred	Alfred	Alfred	Elaine

 10

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

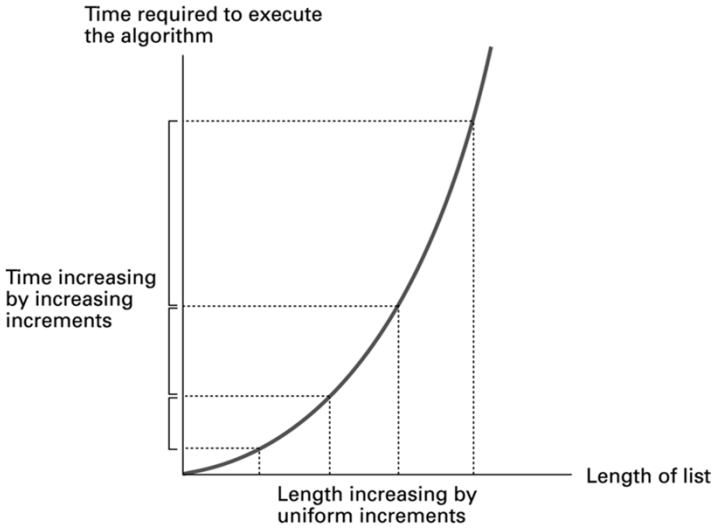
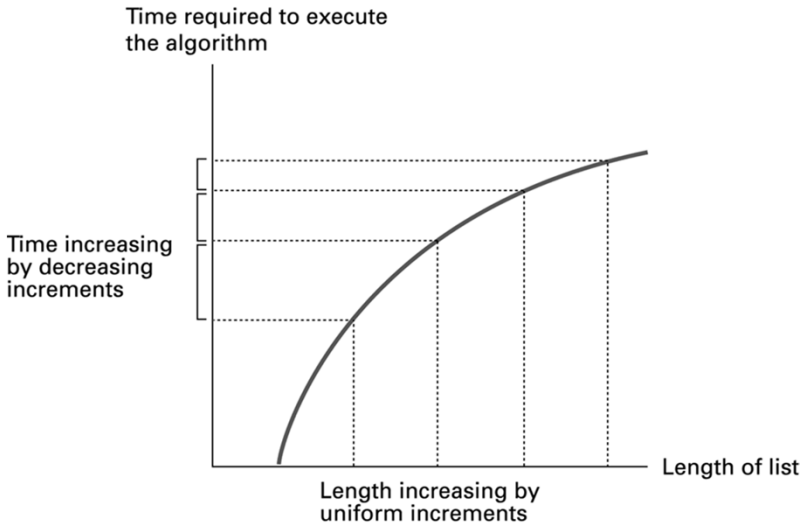


Figure 5.20 **Graph of the worst-case analysis of the binary search algorithm**



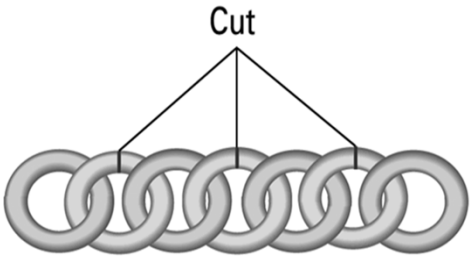
Software Verification

- **Proof of correctness**
  - Not widely used
  - Formal: Formal Logic
  - Assertions
    - Preconditions
    - Loop invariants
- **Testing**
  - Most software use this method for "verification"
  - Informal: Intuition
  - Proves nothing more than that the program performs correctly for the cases under which it was tested.
  - Additional conclusions merely projections.
  - Errors are often consequences of oversight.

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**

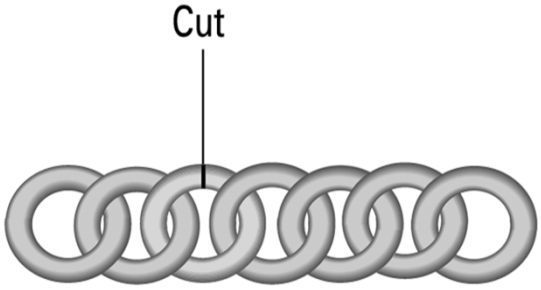


Solution: Give 1 link every morning

CORRECT / OPTIMAL SOLUTION?

 15

Figure 5.22 **Solving the problem with only one cut**



OPTIMAL

Why?
At minimum a single link must be given to the hotel on the first morning, which requires 1 cut.

Solution:
1: Give 1 **2**: Get 1/ Give 2 **3**: Give 1 **4**: Get 3 / Give 4
5: Give 1 **6**: Get 1/ Give 2 **7**: Give 1

 16

Believed Correct vs. Correct

- Testing
  - Intuitive
  - Many instance where programs were thought to be correct
    - AT&T: Software for 114 switching stations error after about a month of operation caused approx. 5 million calls to be blocked over a nine-hour period
- Verification using Assertions
  - Formal Logic
  - Protected from inaccurate conclusions
  - Proof of correctness based on assertions
    - **Preconditions**: Conditions satisfied at start of program
    - **Postconditions**: At end of program, output conditions are met
    - **Loop Invariants**: Every time a point in loop reached, is true

Figure 5.23 **The assertions associated with a typical while structure**