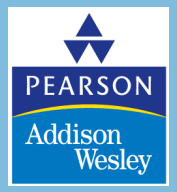
# Chapter 5: Algorithms

## Computer Science: An Overview
## Tenth Edition

by
J. Glenn Brookshear



# What is an algorithm?

- **Let’s play a game!**

- Step1: Please stand up
- Step2: Each student initiate with number X = 1
- Step 3: Add the X of your closest neighbor to your X $X_{new} = X_{old} + X$
- Step 4: Please let your neighbor **sit down**
- The remaining standing students repeat from Step 3 to Step 4

# Algorithm and program

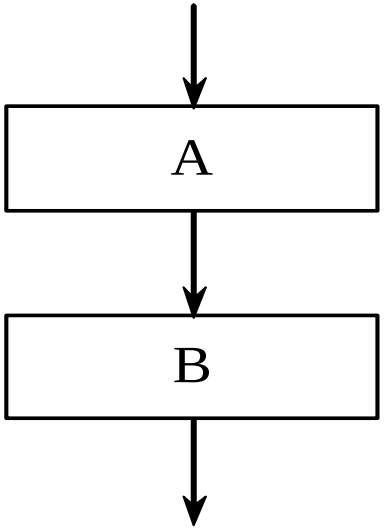
- Algorithm
  - An **ordered** set of **unambiguous**, **executable** steps that defines a **terminating** process to solve the problem
- Program
  - A set of instructions, which describe how computers process data and solve the problem

# Two elements of algorithm

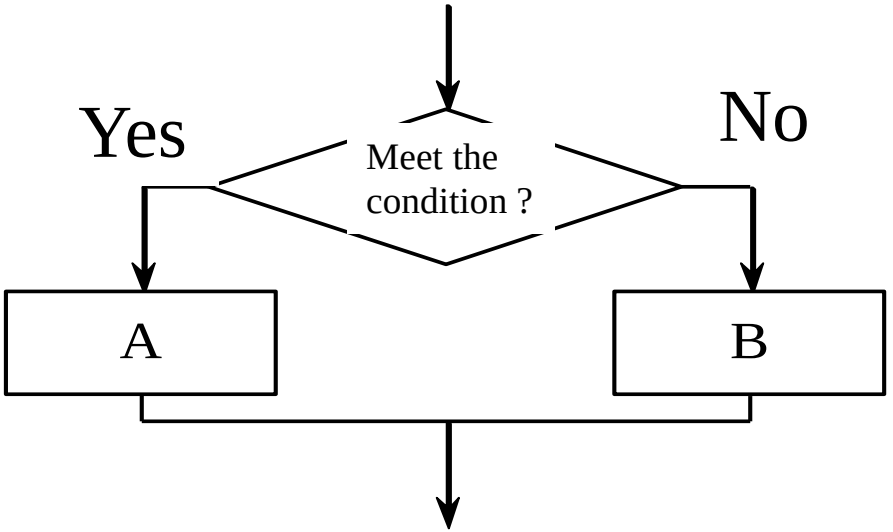
- **Algorithm: operation + control structure**
- **Operation:**
  - Arithmetic: +, -, *, / , etc.
  - Relation: >=, <=, etc.
  - Logic: and, or, not, etc.
  - Data transfer: load, store

# Control structure

- Sequential
- Conditional
- loop

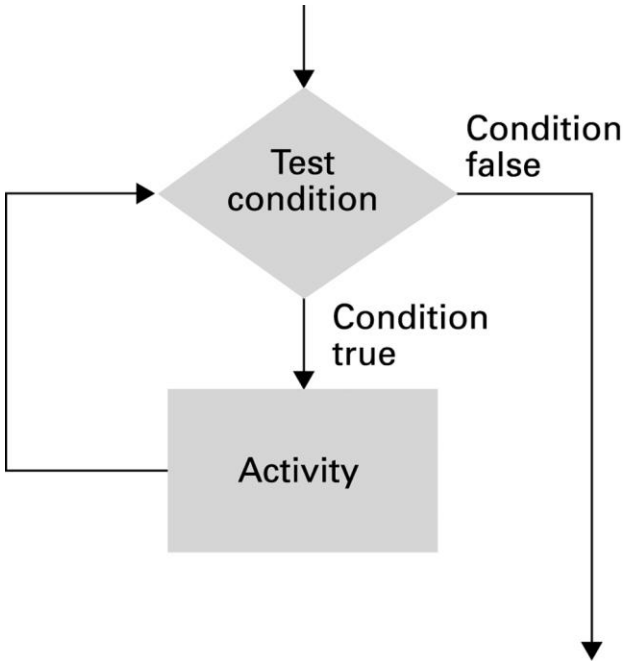



Sequential

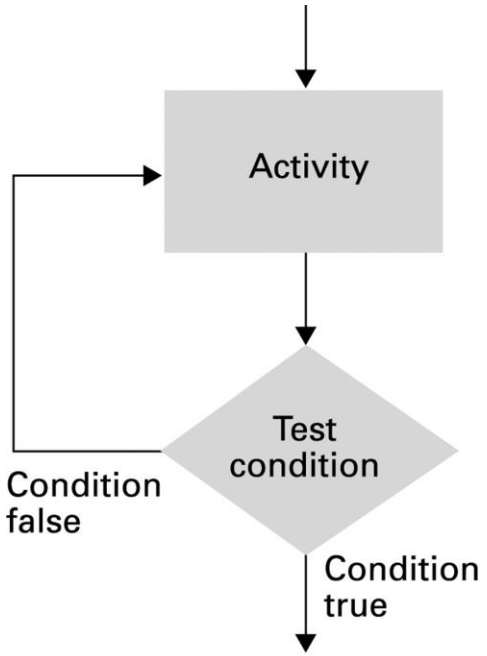



Conditional

# Loop structure




While loop structure




Repeat loop structure

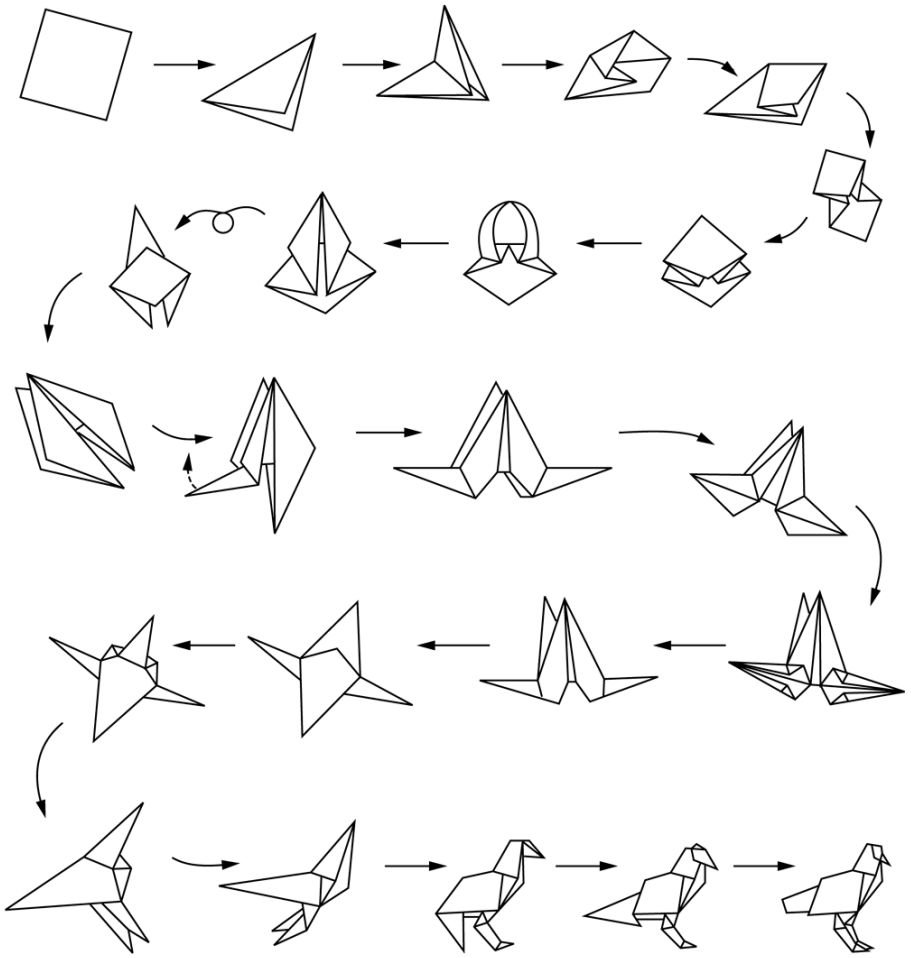
# Algorithm Representation

- Requires well-defined primitives
- A collection of primitives constitutes a programming language.

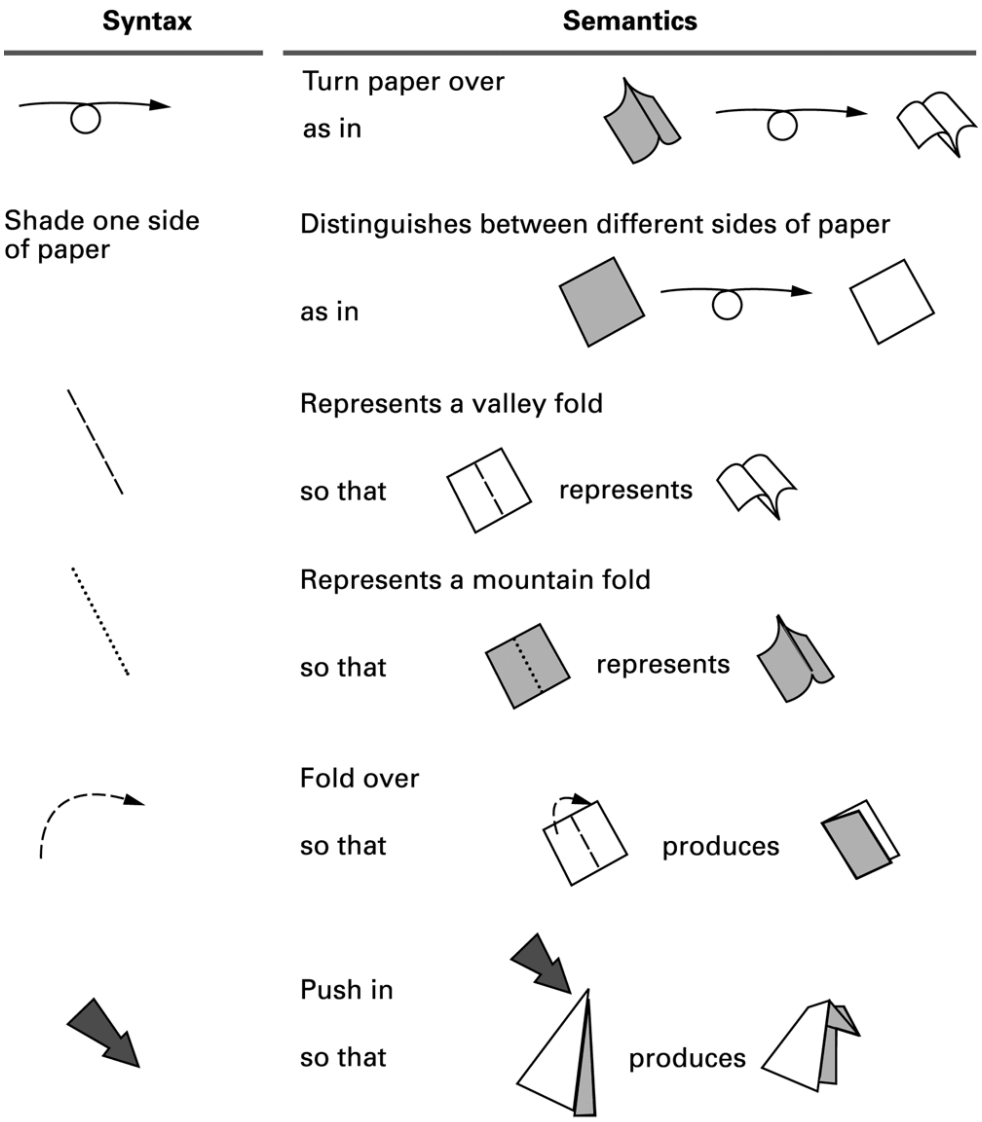
# Algorithm representation

- Natural language
- Traditional flow chart
- N-S flow chart
- Pseudo code

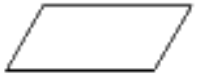
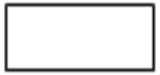
# Figure 5.2 **Folding a bird from a square piece of paper**



# Natural language: Origami primitives



# Traditional flow chart

| 符号名称 | 图形 | 功能 |
| --- | --- | --- |
| 起止框 |  | 表示算法的开始和结束 |
| 输入/输出框 |  | 表示算法的输入/输出操作 |
| 处理框 |  | 表示算法中的各种处理操作 |
| 判断框 |  | 表示算法中的条件判断操作 |
| 流程线 |  | 表示算法的执行方向 |
| 连接点 |  | 表示流程图的延续 |

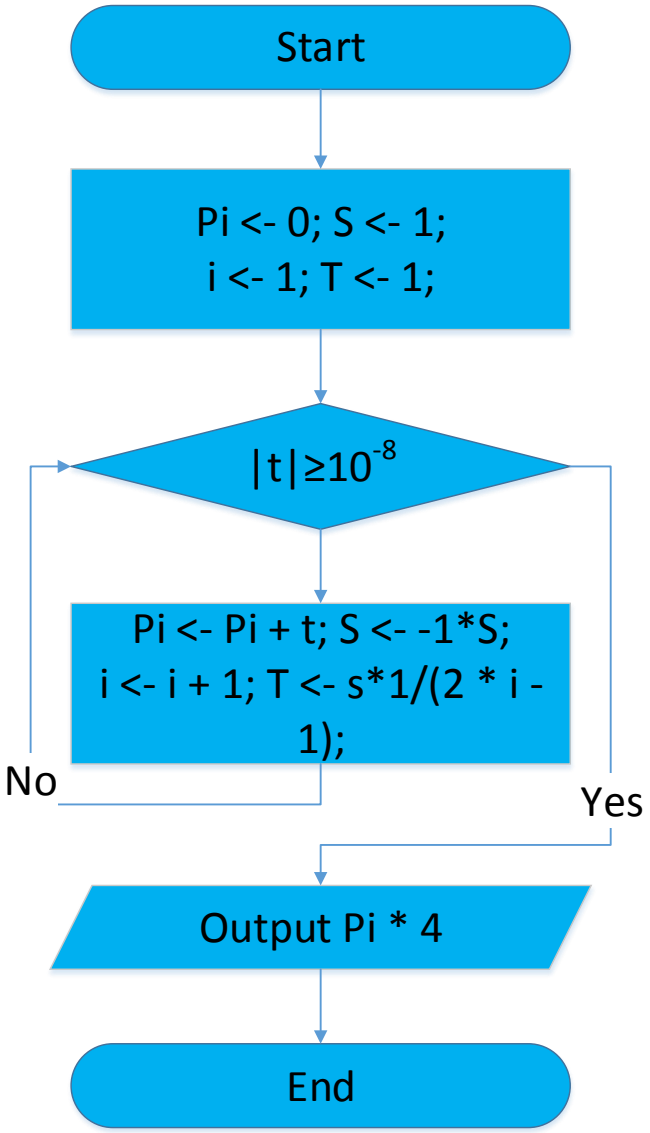
# Example: estimation of π

公式 1: $\frac{\pi}{2}=\frac{2^2}{1\times3}\times\frac{4^2}{3\times5}\times\frac{6^2}{5\times7}\times\frac{8^2}{7\times9}\times\cdots$

公式 2: $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots$

公式 3: $\frac{\pi}{6}=\frac{1}{\sqrt{3}}\times(1-\frac{1}{3\times3}+\frac{1}{3^2\times5}-\frac{1}{3^3\times7}+\cdots)$

# Computation of Pi

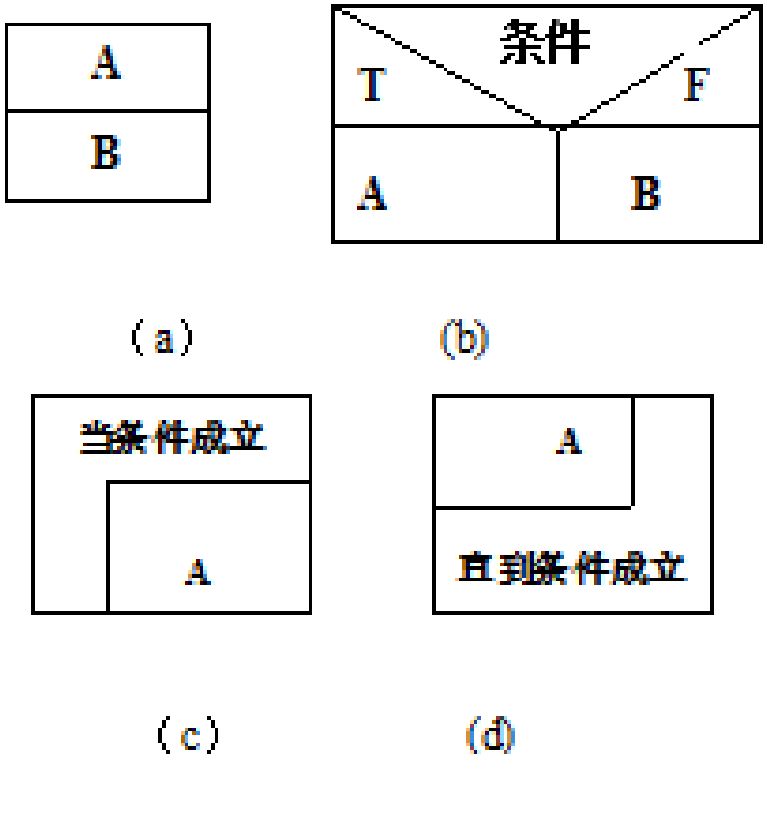


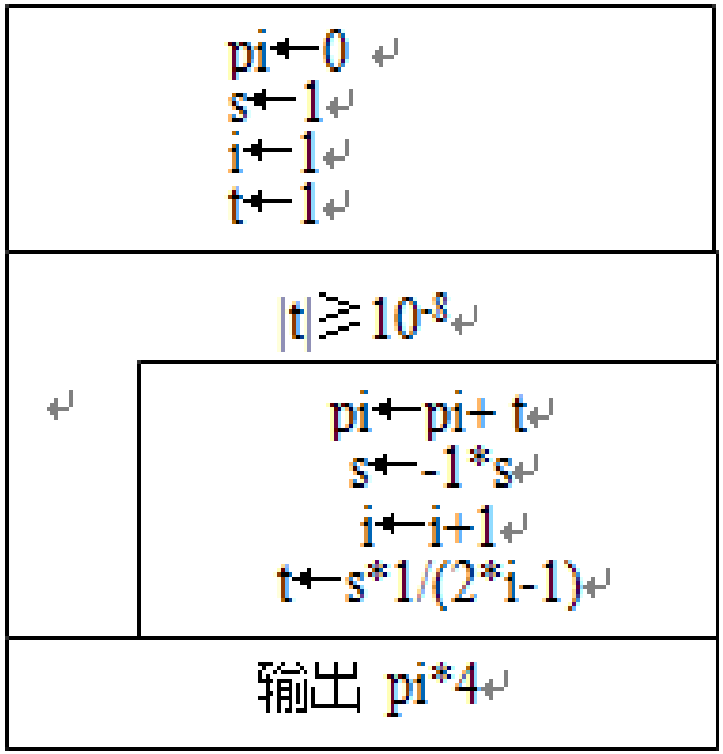

公式 2: $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots+$

# N-S flow chart

- **Proposed by I.Nassi and B.Shneideman**
- **For** structured programming

结构化程序设计（structured programming）的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。




# 3 control structures for N-S flow chart

# Pseudocode Primitives

- Assignment

  *name* ← *expression*

- Conditional selection

  **if** *condition* **then** *action*

# Pseudocode Primitives (continued)

- Repeated execution

  **while** *condition* **do** *activity*

- Procedure

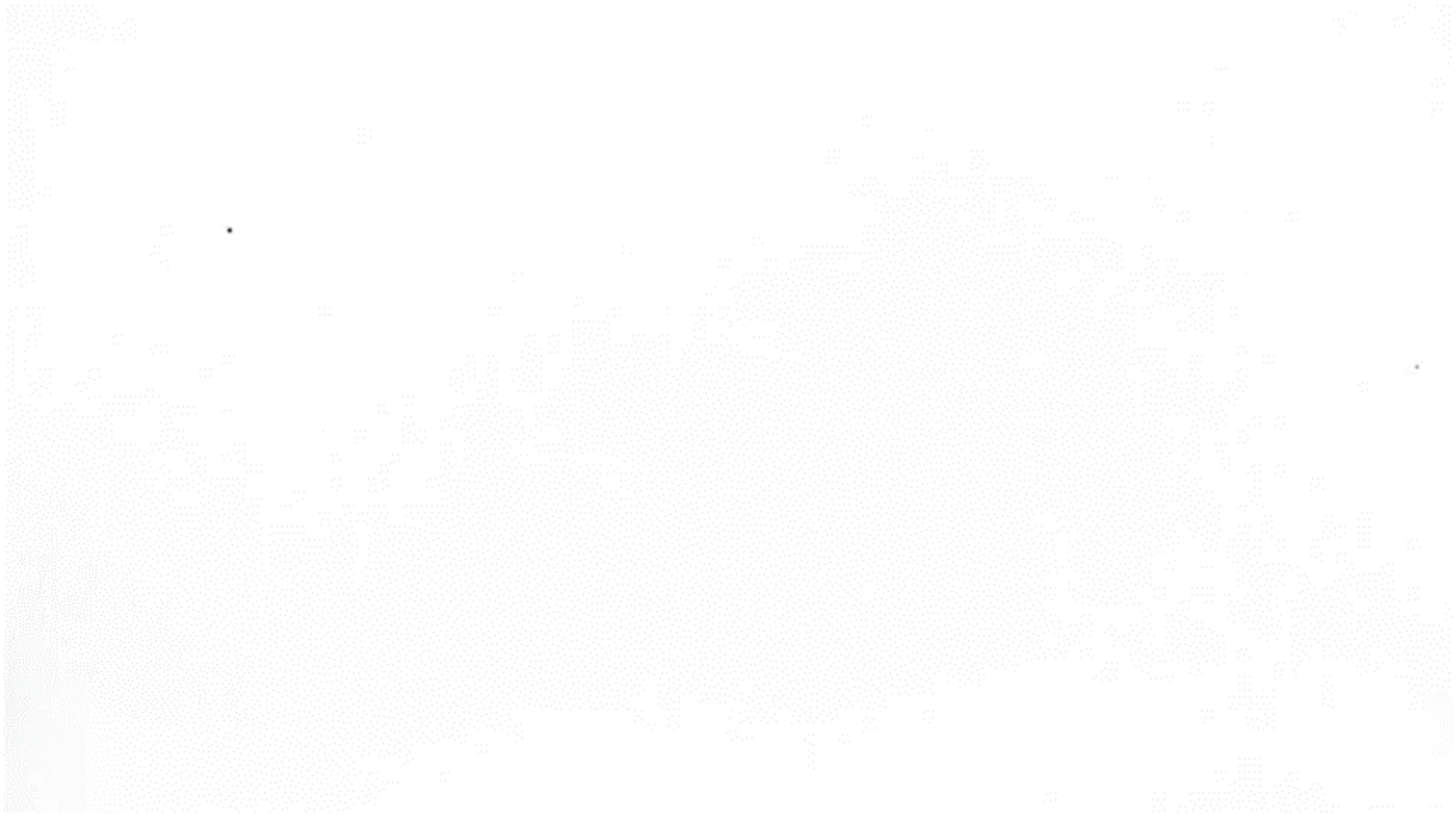
  **procedure** *name* (*generic names*)

# Figure 5.4 The procedure Greetings in pseudocode

```
procedure Greetings
Count ← 3;
while (Count > 0) do
    (print the message "Hello" and
        Count ← Count –1)
```

# Real code?

# What algorithms can do?




RUN
00.00 m/h

# Repair using shrink-wrapping

Junqiao Zhao
Hugo Ledoux
Jantien Stoter

TU Delft 2013

# Algorithm discovery

- Art

  = analysis + knowledge + experiment + inspiration (potential)

# 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?

# Figure 5.5

**a.** Triples whose product is 36

| | |
|---|---|
| (1,1,36) | (1,6,6) |
| (1,2,18) | (2,2,9) |
| (1,3,12) | (2,3,6) |
| (1,4,9) | (3,3,4) |

**b.** Sums of triples from part (a)

| | |
|---|---|
| 1 + 1 + 36 = 38 | 1 + 6 + 6 = 13 |
| 1 + 2 + 18 = 21 | 2 + 2 + 9 = 13 |
| 1 + 3 + 12 = 16 | 2 + 3 + 6 = 11 |
| 1 + 4 + 9 = 14 | 3 + 3 + 4 = 10 |

# ACM ICPC 2014 (International Collegiate Programming Contest)

# Iterative Structures

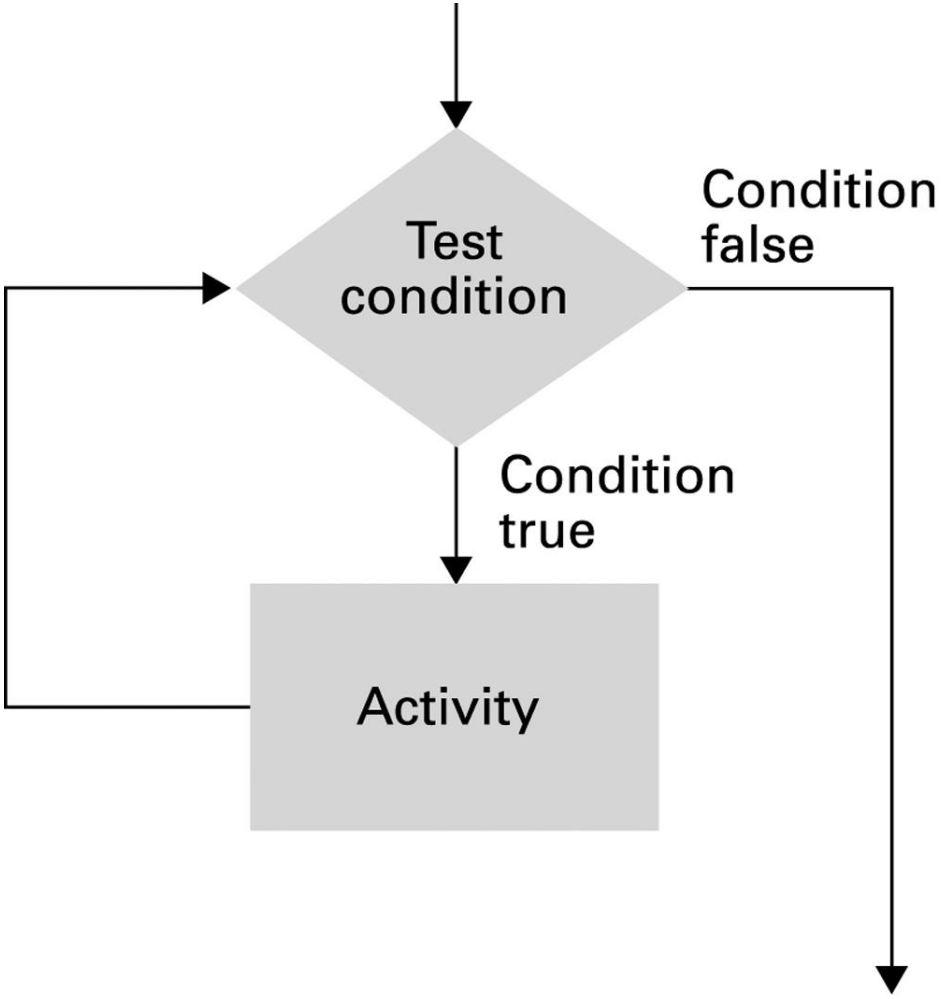
- Pretest loop:

```
while (condition) do
    (loop body)
```

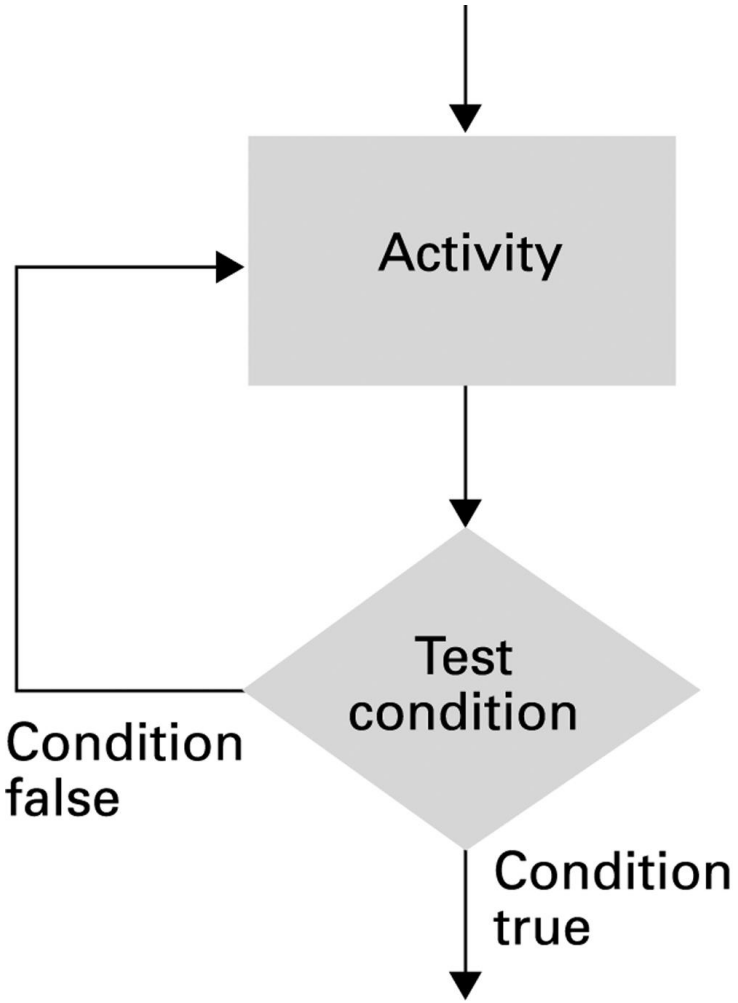
- Posttest loop:

```
repeat (loop body)
    until(condition)
```

# Figure 5.8 The while loop structure



# Figure 5.9 The repeat loop structure



# Search algorithms

- Sequential search (顺序查找)
- Binary search (二分查找)

# Sequential search algorithm in 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
```

# Figure 5.7 Components of repetitive control

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

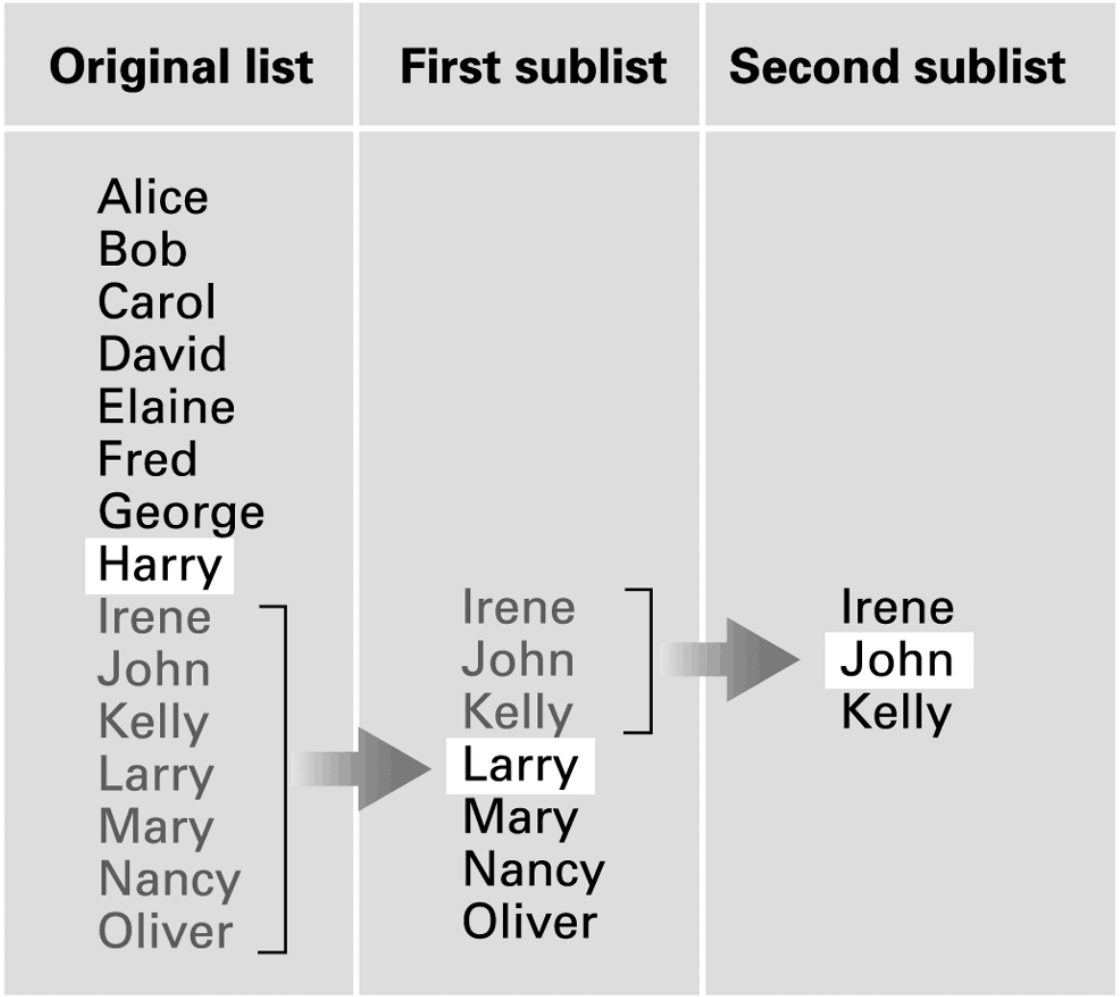
**Test:** Compare the current state to the termination condition and terminate the repetition if equal

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

# 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.

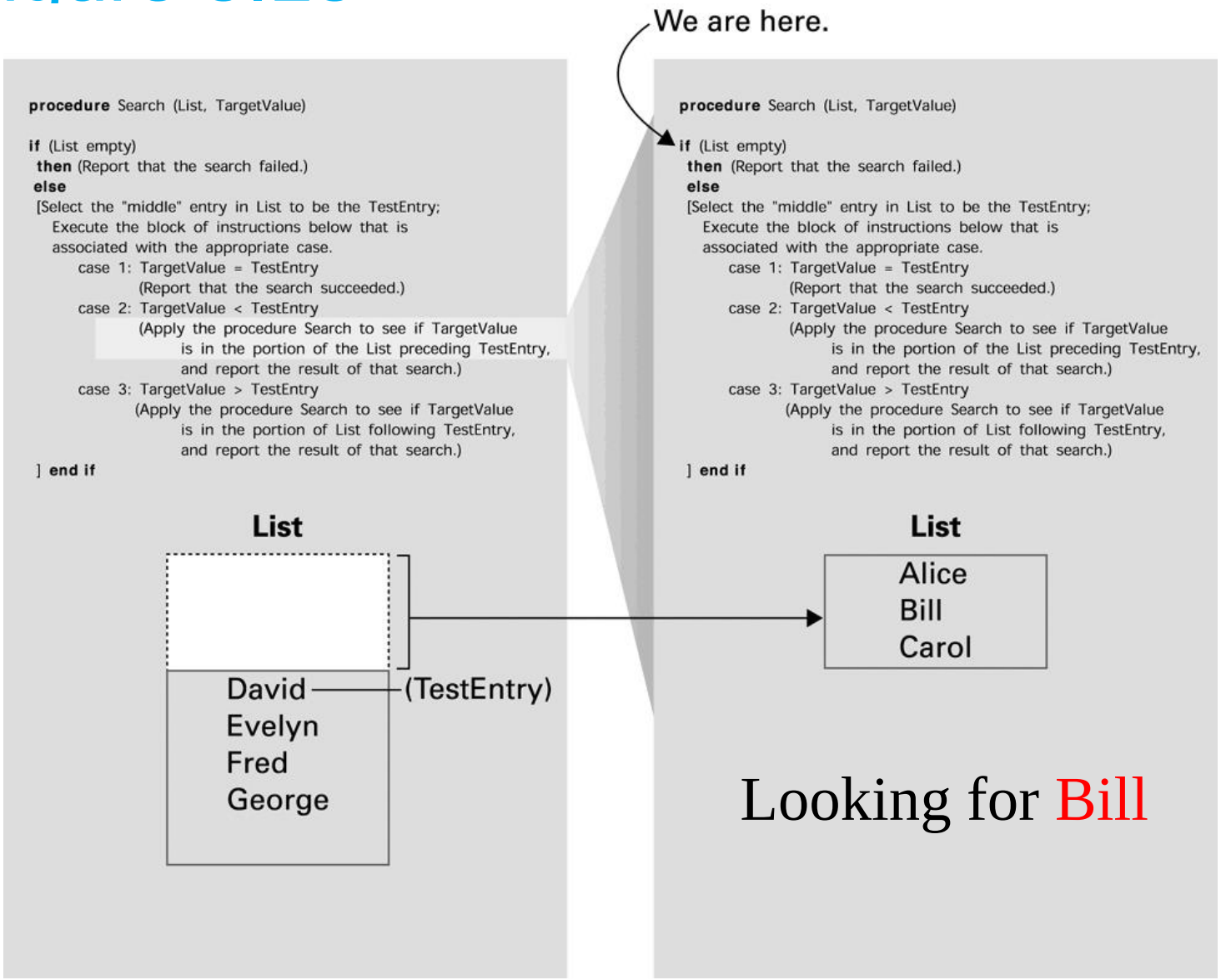
# Figure 5.12 **Applying our strategy to search a list for the entry John**



# 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
```

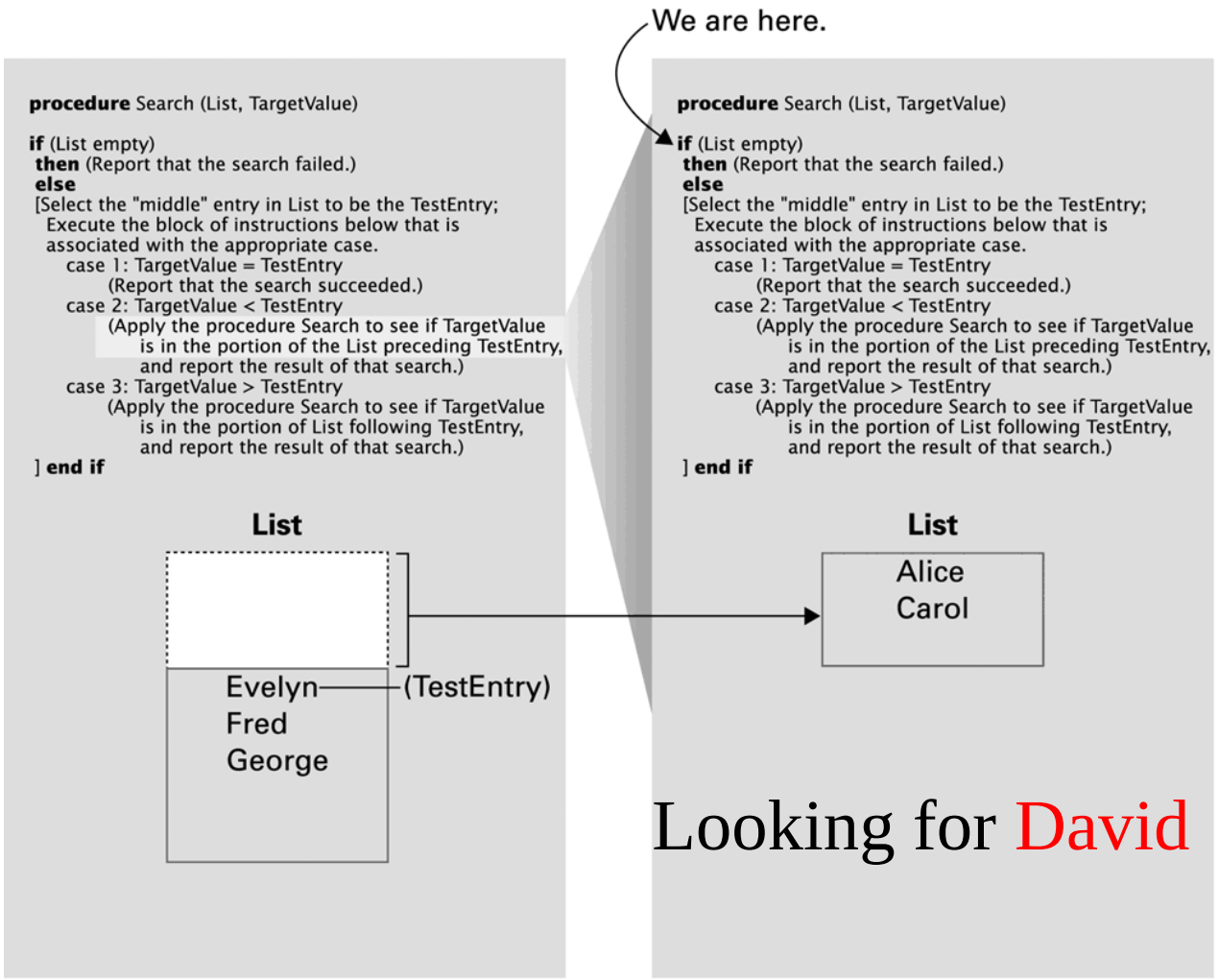
# Figure 5.15



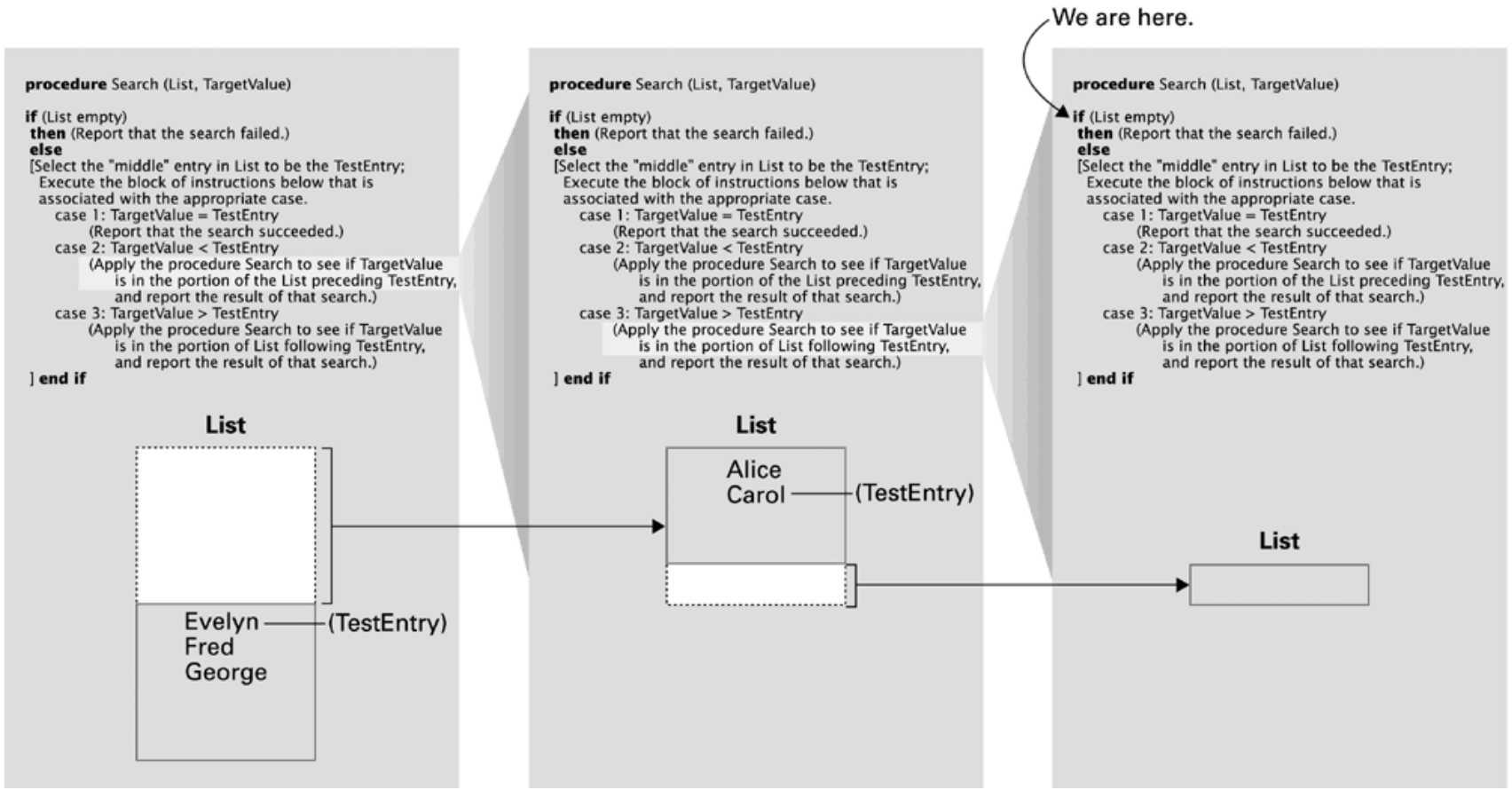

Alice
Bill
Carol
David
Evelyn
Fred
George

# Figure 5.16

Alice
Carol
Evelyn
Fred
George



# Figure 5.17

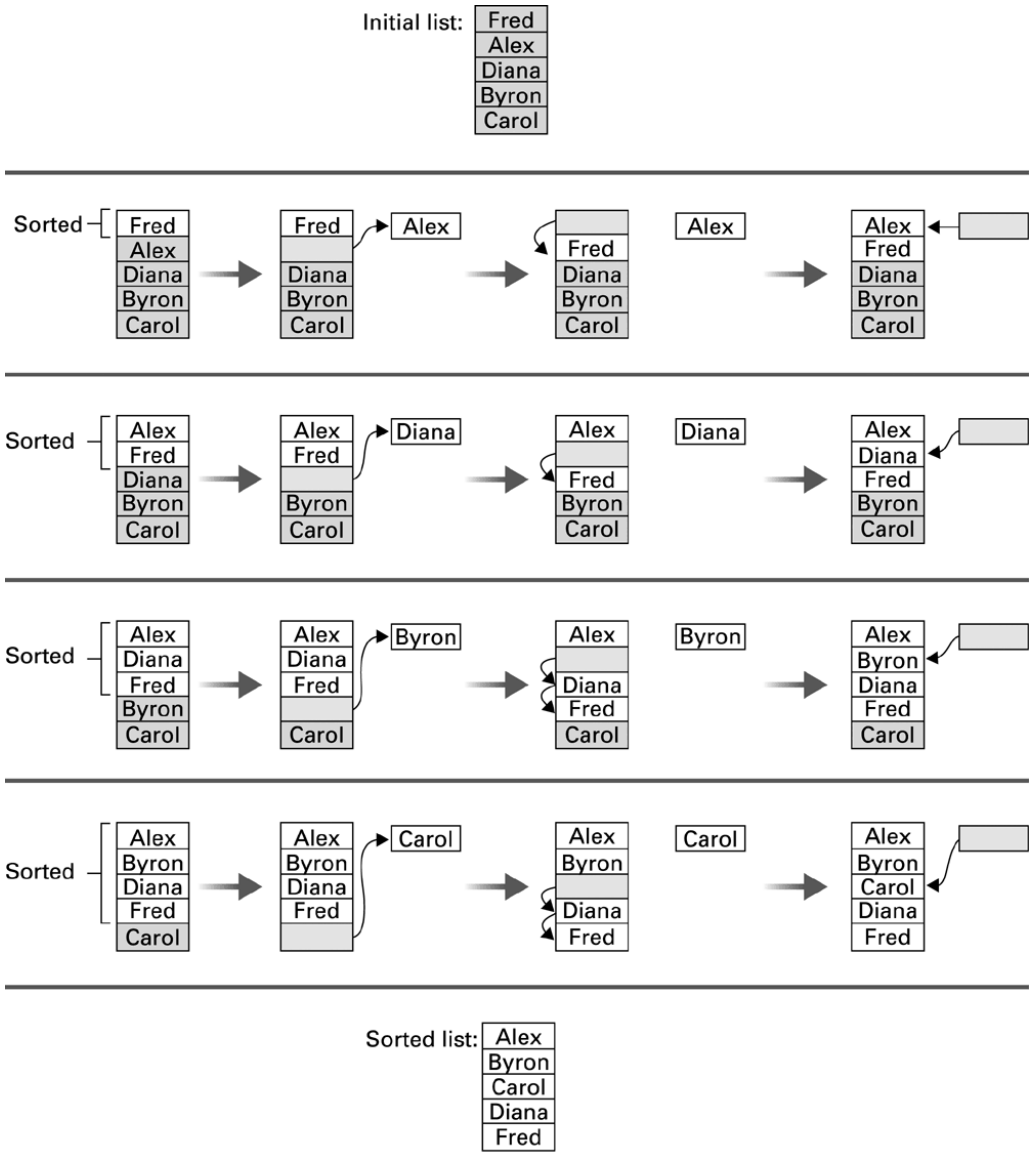



Looking for David

# Sort algorithm

- Insertion sort (插入排序)
- Selection sort (选择排序)
- Bubble sort (冒泡排序)

# Insertion sort



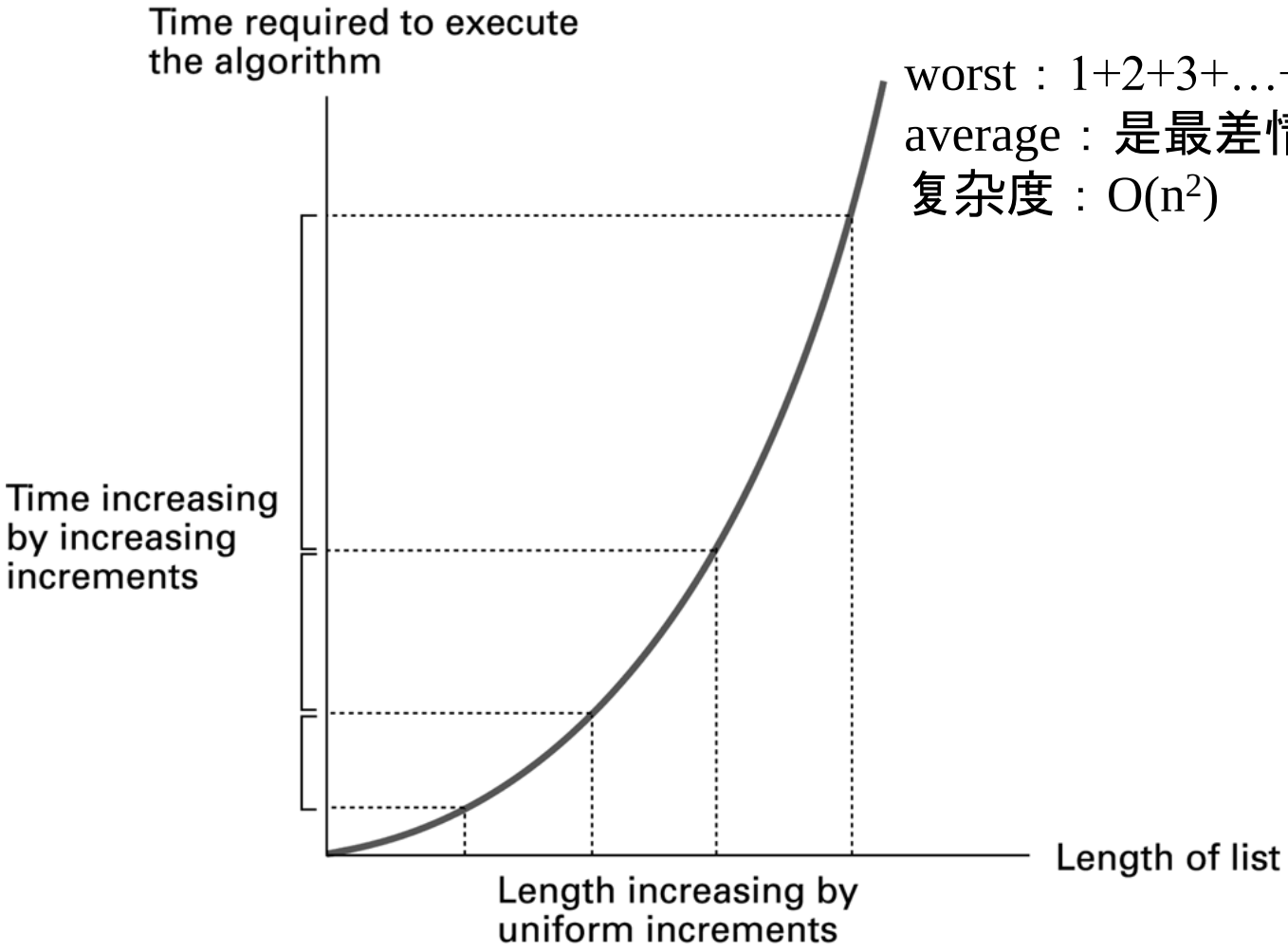
# Figure 5.11 The insertion sort algorithm expressed in pseudocode

```
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
      )
```

# Algorithm Efficiency

- Measured as number of instructions executed
- Big O notation: Used to represent efficiency classes
  - Example: Insertion sort is in $O(n^2)$
- Best, worst, and average case analysis

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

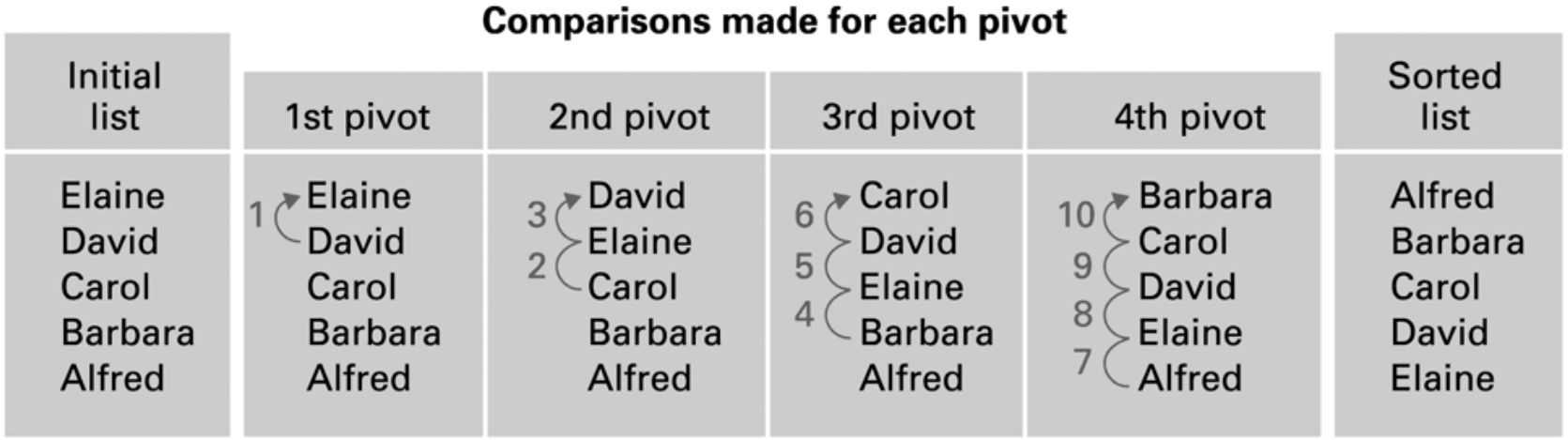



worst：$1+2+3+\ldots+(n-1)=1/2(n^2-n)$

average：是最差情况的一半$1/4(n^2-n)$

复杂度：$O(n^2)$

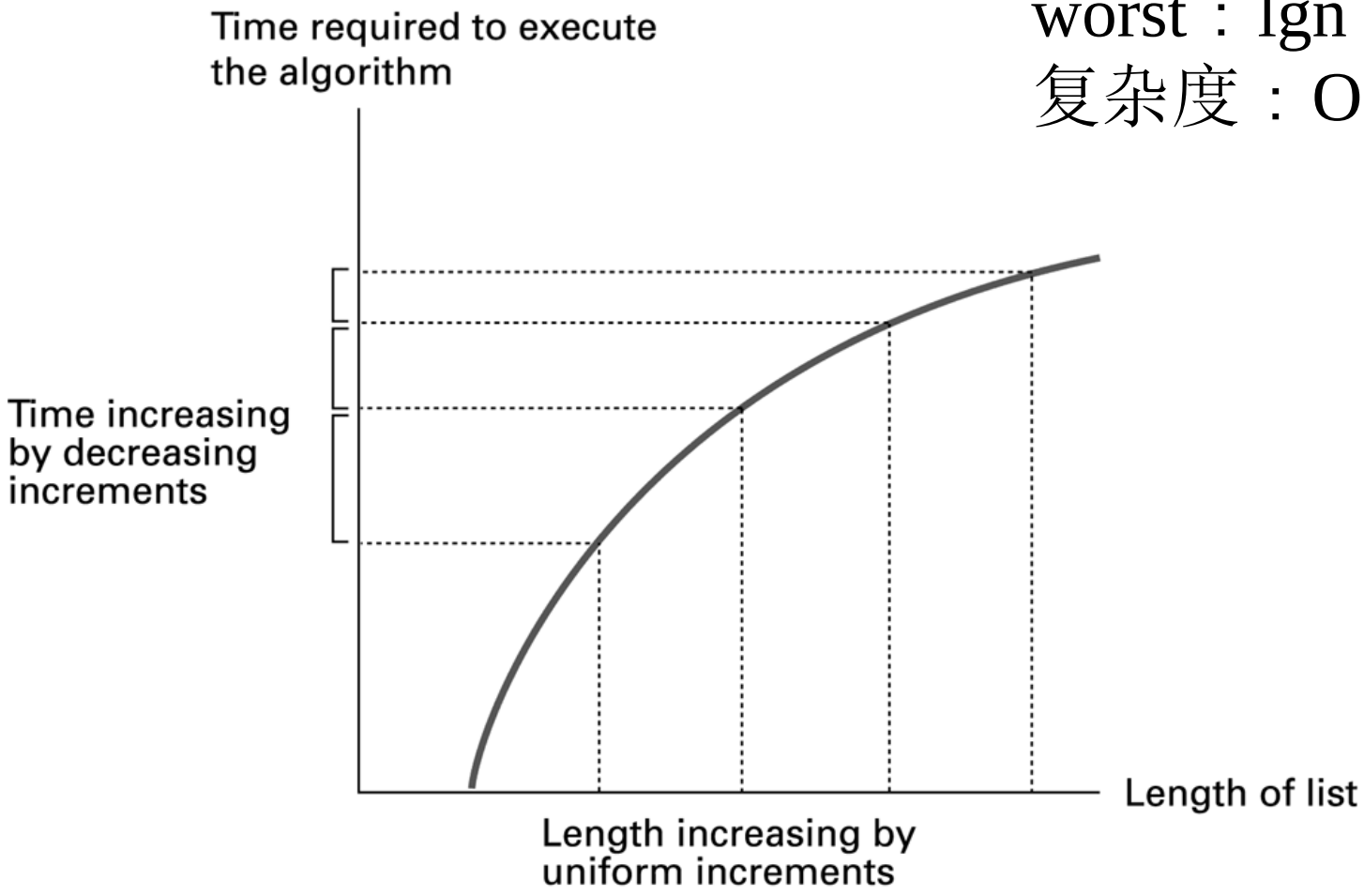
# Figure 5.18 Applying the insertion sort in a worst-case situation



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

worst：lgn

复杂度：O(lgn)



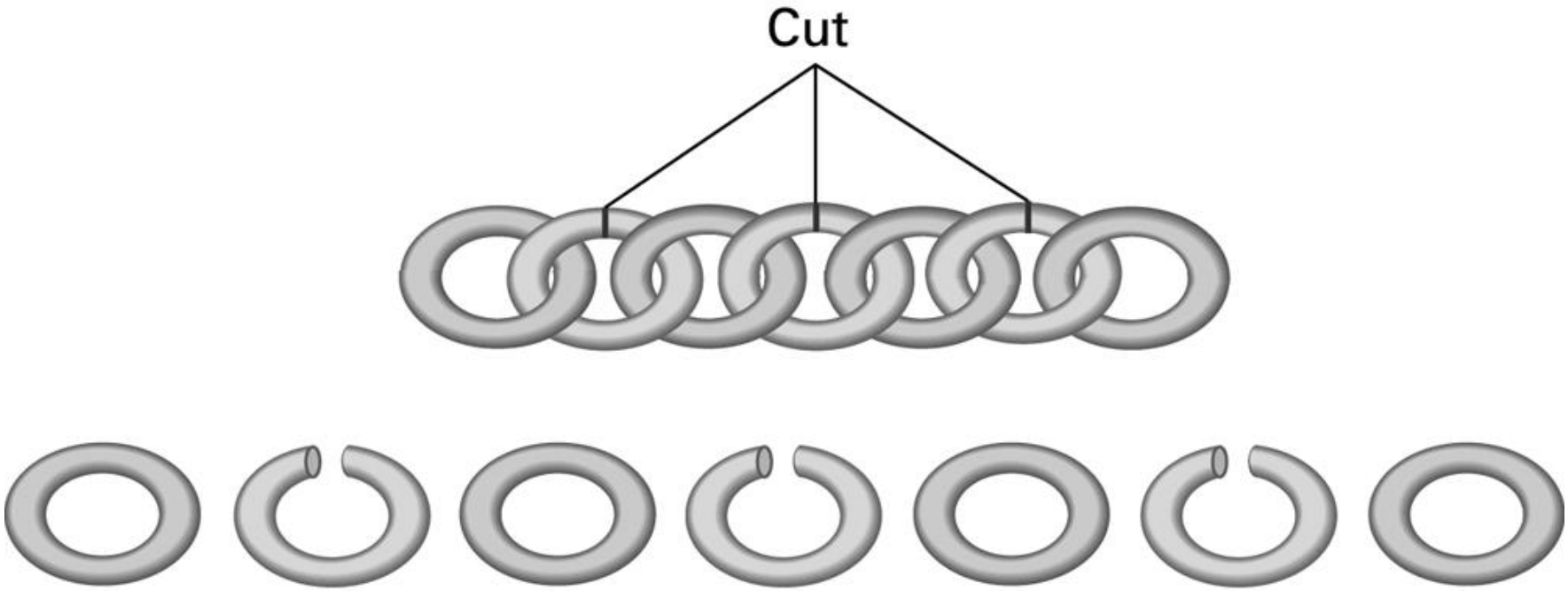
# Software Verification

- Proof of correctness
  - Assertions
    - Preconditions
    - Loop invariants
- Testing

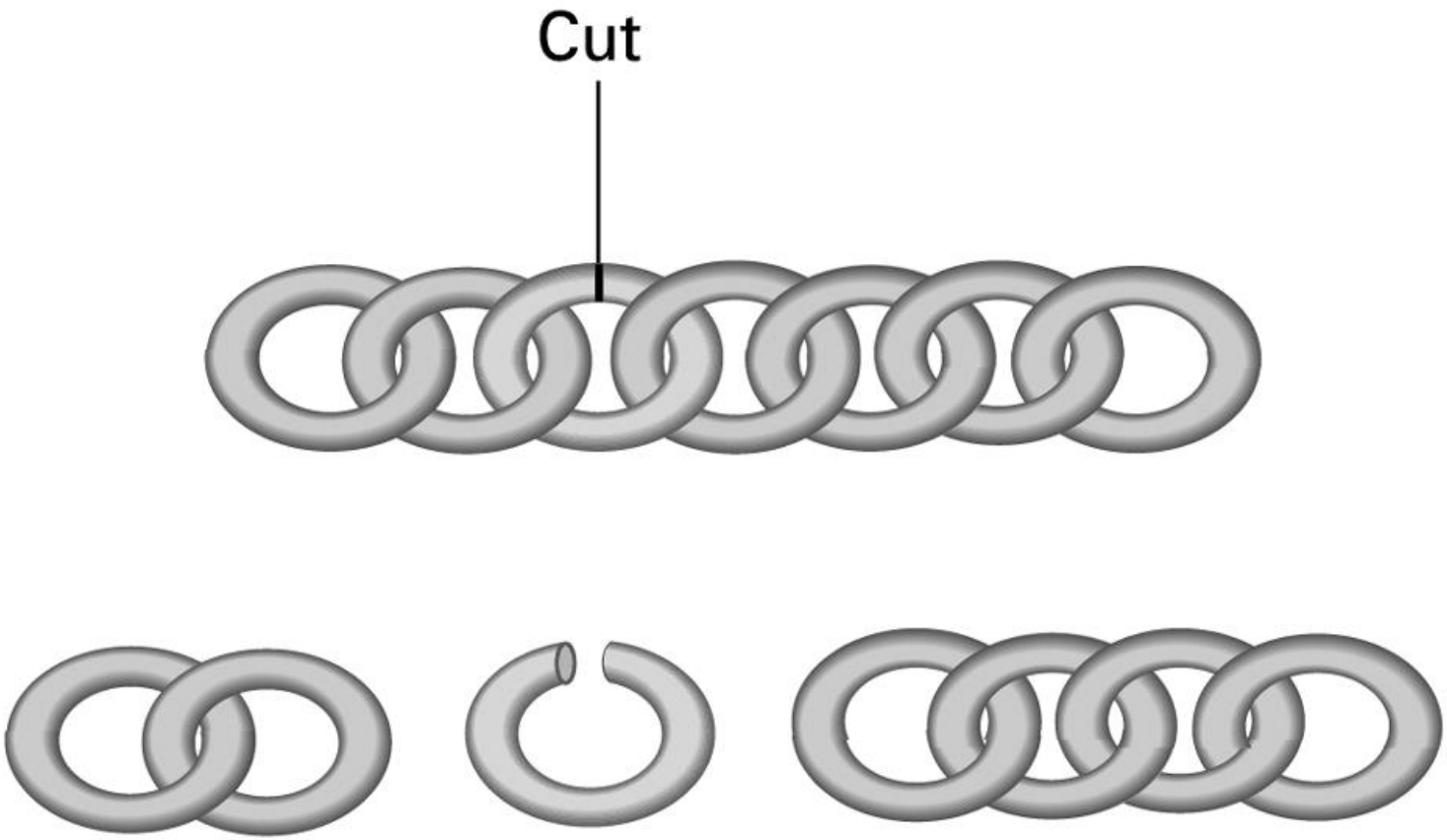
# Chain Separating Problem

- 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?

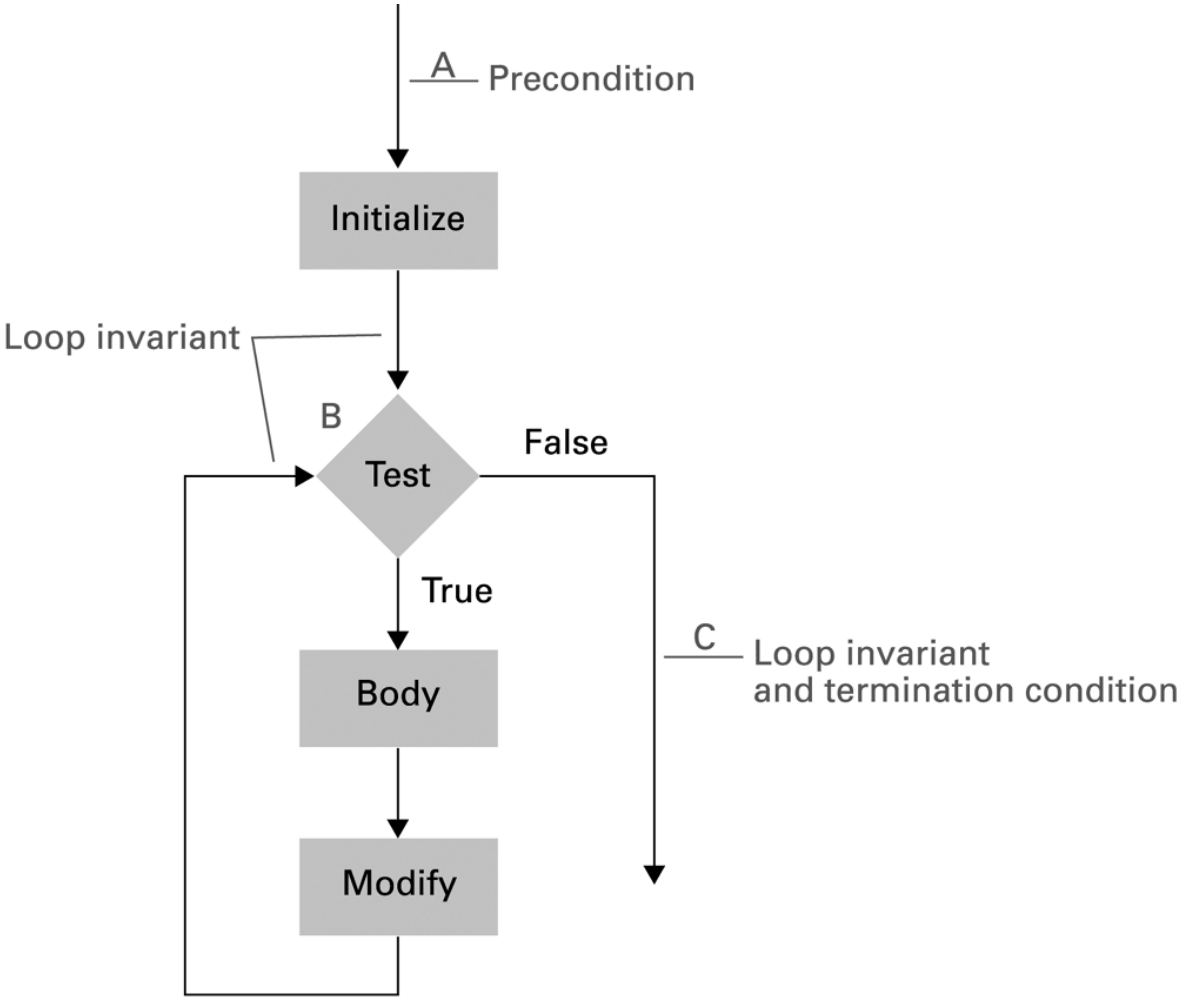
# Figure 5.21 **Separating the chain using only three cuts**



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



# Figure 5.23 The assertions associated with a typical while structure



- Questions?

# Movie: the social network

# Learning Algorithms

- Every class in CS has some relations with algorithms
  - But some classes are not obviously related to algorithms, such as math classes.
    - **微积分, 离散数学, 线性代数, 概率统计, …**
- Why should we study them?
  - They are difficult, and boring, and difficult…

# Why Study Math?

- Train the logical thinking and reasoning
  - Algorithm is a result of logical thinking: correctness, efficiency, …
  - Good programming needs logical thinking and reasoning.  For example, debugging.
- Learn some basic tricks
  - Many beautiful properties of problems can be revealed through the study of math
  - Standing on the shoulders of giants

# Try to Solve These Problems

1. Given a network of thousands nodes, find the shortest path from node A to node B
   - The shortest path problem (**离散数学**)
2. Given a circuit of million transistors, find out the current on each wire
   - Kirchhoff's current law (**线性代数**)
3. In CSMA/CD, what is the probability of collisions?
   - Network analysis (**概率统计**)

# 微积分

- Limits, derivatives, and integrals of continuous functions.
- Used inalmost every field
  - 网络分析, 效能分析
  - 信号处理, 图像处理, 模拟电路设计
  - 科学计算, 人工智能, 计算机视觉, 计算机图学
  - …
- Also the foundation of many other math
  - 概率统计, 工程数学

# 离散数学

- Discrete structure, graph, integer, logic, abstract algebra, combinatorics
- The foundation of computer science
  - Every field in computer science needs it
  - Particularly, 算法, 数字逻辑设计, 密码学, 编码理论, 计算机网络, CAD, 计算理论
- Extended courses
  - Special topics on discrete structure, graph theory, concrete math

# 线性代数

- Vectors, matrices, tensors, vector spaces, linear transformation, system of equations
- Used in almost everywhere when dealing with more than one number
  - 网络分析, 效能分析
  - 信号处理, 图像处理
  - 科学计算，人工智能, 计算机视觉, 计算机图学
  - CAD,电路设计

# 概率统计

- Probability models, random variables, probability functions, stochastic processes
- Every field uses it
  - Particularly,网络分析, 效能分析, 信号处理, 图像处理, 科学计算，人工智能, 计算机视觉…
  - Other examples: randomized algorithm, queue theory, computational finance, performance analysis

# 工程数学

- A condensed course containing essential math tools for most engineering disciplines
  - Partial differential equations, Fourier analysis, Vector calculus and analysis
- Used in the fields that need to handle continuous functions
  - 网络分析, 效能分析, 信号处理, 图像处理, 科学计算, 人工智能, 计算机视觉, CAD,电路设计