บทที่ 8

ระเบียบขั้นตอนวิธี

Algorithms

วัตถุประสงค์

หลังจากเรียนจบบทที่ 8 แล้ว นักศึกษาต้องสามารถ:

- เข้าใจมโนทัศน์ของระเบียบขั้นตอนวิธี หรือ อัลกอริธึม
- อธิบายและใช้ 3 โครงสร้างเพื่อการพัฒนาระเบียบขั้นตอนวิธี: sequence, decision, และ repetition
- เข้าใจและสามารถใช้เครื่องมือ 3 ประเภทเพื่อแทนระเบียบขั้นตอนวิธี: flowchart, pseudocode, และ structure chart
- เข้าใจหลักการของ modularity และ subalgorithms
- ระบุและเข้าใจระเบียบวิธีที่ใช้กันทั่วไป

8.1

แนวคิดเบื้องต้น
CONCEPT

8.1 แนวความคิดเบื้องต้น

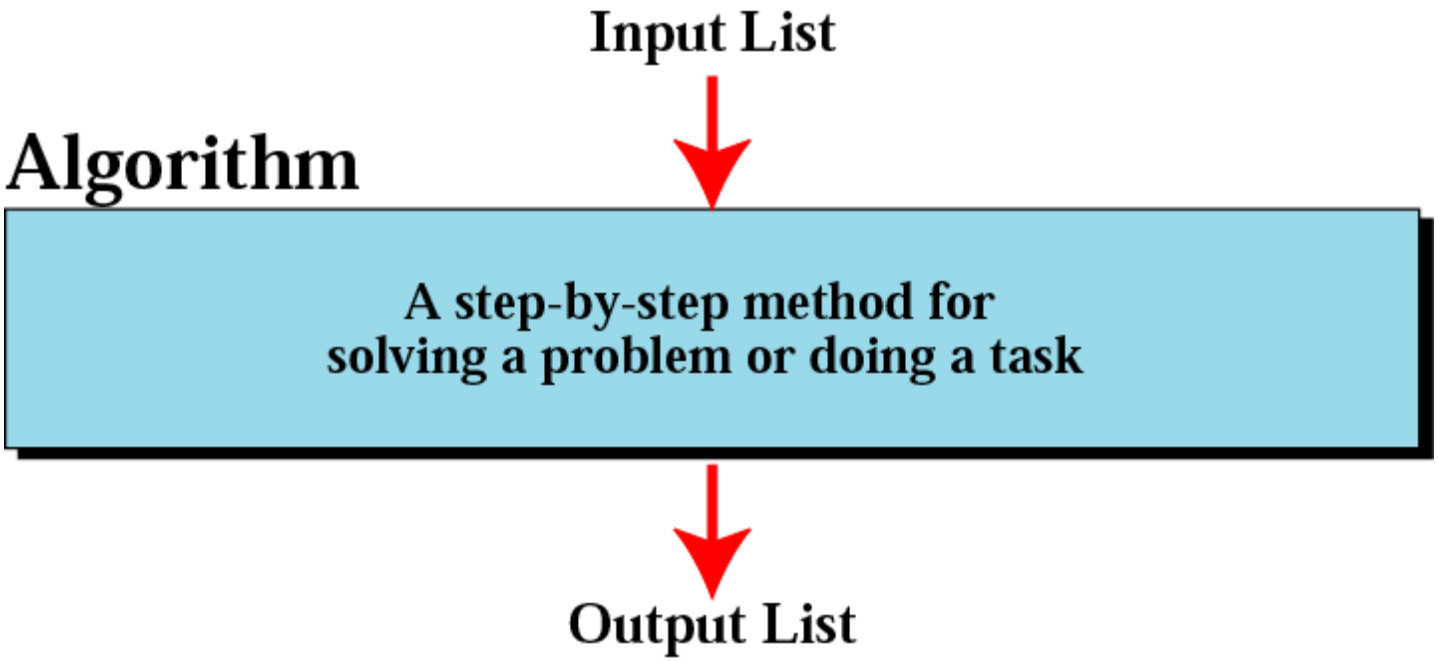
8.1.1 นิยามอย่างไม่เป็นทางการ

"อัลกอริธึม" คือระเบียบวิธีหรือขั้นตอนที่มีจุดเริ่มต้นและจุดสิ้นสุดในการแก้ปัญหาหรือทำงานอย่างใดอย่างหนึ่ง

ตามนิยามนี้ อัลกอริธึมเป็นขั้นตอนการแก้ปัญหาที่เป็นอิสระต่อระบบและภาษาคอมพิวเตอร์ ถ้าจะให้นิยามที่กระชับมากขึ้นอาจเราอาจระบุได้ว่า อัลกอริธึมจะรับรายการข้อมูลนำเข้าและสร้างรายการผลลัพธ์ ดังรูปที่ 8.1

รูปที่ 8-1

นิยามของอัลกอริธึมที่ใช้ในคอมพิวเตอร์



8.1.2 ตัวอย่าง: การค้นหาเลขจำนวนเต็มที่มากที่สุด

โจทย์: สมมติว่าเราต้องการออกแบบอัลกอริธึมเพื่อหาเลขจำนวนเต็มที่มากที่สุดในกลุ่มของเลขจำนวนเต็มบวกที่กำหนดให้ อัลกอริธึมจะต้องไม่ขึ้นอยู่กับเลขชุด ใดชุดหนึ่งโดยเฉพาะ ต้องสามารถทำงานได้กับทุกกรณีกับเซตของเลขจำนวนเต็มบวก

แนวคิด: เราเห็นชัดเจนว่าการหาเลขที่มากที่สุดจากชุดของตัวเลขจำนวนมากๆนั้น คงไม่สามารถทำได้เพียงขั้นตอนเดียวแน่นอน ไม่ว่าจะทำโดยคนหรือโดยคอมพิวเตอร์ ดังนั้นอัลกอริธึมจำเป็นจะต้องตรวจสอบเลขแต่ละจำนวนอย่างละเอียด

วิธีแก้ปัญหา: เราจะใช้วิธีการหยั่งรู้เพื่อแก้ปัญหานี้ เริ่มต้นจากเซตของเลขจำนวนน้อยๆก่อนเช่น 5 จำนวน แล้วจึงค่อยๆขยายวิธีการไปยังเลขจำนวนมากๆ วิธีการแก้ปัญหาสำหรับเลข 5 ตัวที่มีหลักการและข้อจำกัดที่ชัดเจนก็

สามารถใช้แก้ปัญหาที่มีเลข 1000 หรือ 1000000 จำนวนได้เช่นกัน สมมติว่าเรามีเลข 5 จำนวน อัลกอริธึมจะตรวจสอบตัวเลขทีละตัว ดูตัวแรกโดยไม่ทราบว่าตัวที่สองคืออะไร? จากนั้นดูตัวที่สอง สาม สี่ ห้า ตามลำดับ (ดูรูปที่ 8.2 ประกอบ)

สมมติเราเรียกอัลกอริธึมนี้ว่า FindLargest อัลกอริธึมจะรับอินพุทเป็นจำนวนเต็ม 5 จำนวน แล้วส่งผลลัพธ์เป็นจำนวนเต็มที่มีค่ามากที่สุดออกมา เราเขียนเป็นรูปแบบดังนี้

อินพุท: อัลกอริธึมรับจำนวนเต็ม 5 จำนวนคือ 12 8 13 9 11

การประมวลผล: อัลกอริธึมทำงาน 5 ขั้นตอนดังนี้

ขั้นที่ 1: อัลกอริธึมจะตรวจสอบตัวแรก (คือ 12) ก่อน เนื่องจากอัลกอริธึมไม่สามารถเห็นเลขที่เหลืออยู่ได้ มันจึงกำหนดให้เลขตัวแรกเป็นค่าตัวเลขที่มีค่ามากที่สุด สมมติเราเก็บตัวเลขไว้ที่ตัวแปร Largest ดังนั้น ณ จุดนี้ ตัว

แปร Largest จะมีค่าเท่ากับ 12 แล้วจึงไปทำในขั้นตอนต่อไป

ขั้นที่ 2: อัลกอริธึมตรวจสอบตัวเลขตัวที่ 2 (คือ 8) แล้วเปรียบเทียบกับค่าใน Largest พบว่าค่าใน Largest มีค่ามากกว่าตัวเลขตัวที่ 2 ตัวแปร Largest จึงยังต้องเก็บค่าเดิม (คือ 12) ไว้ตามเดิมไม่มีการเปลี่ยนแปลง

ขั้นที่ 3: ตรวจสอบอินพุทตัวที่ 3 (คือ 13) ซึ่งมีค่ามากกว่าค่าที่เก็บอยู่ใน Largest (คือ 12) แสดงว่าค่าที่เก็บอยู่ใน Largest ไม่ใช่ตัวเลขที่มีค่ามากที่สุดอีกต่อไปแล้ว จึงต้องเปลี่ยนค่าของ Largest จาก 12 เป็น 13 จากนั้นจึงไปทำขั้นตอนต่อไป

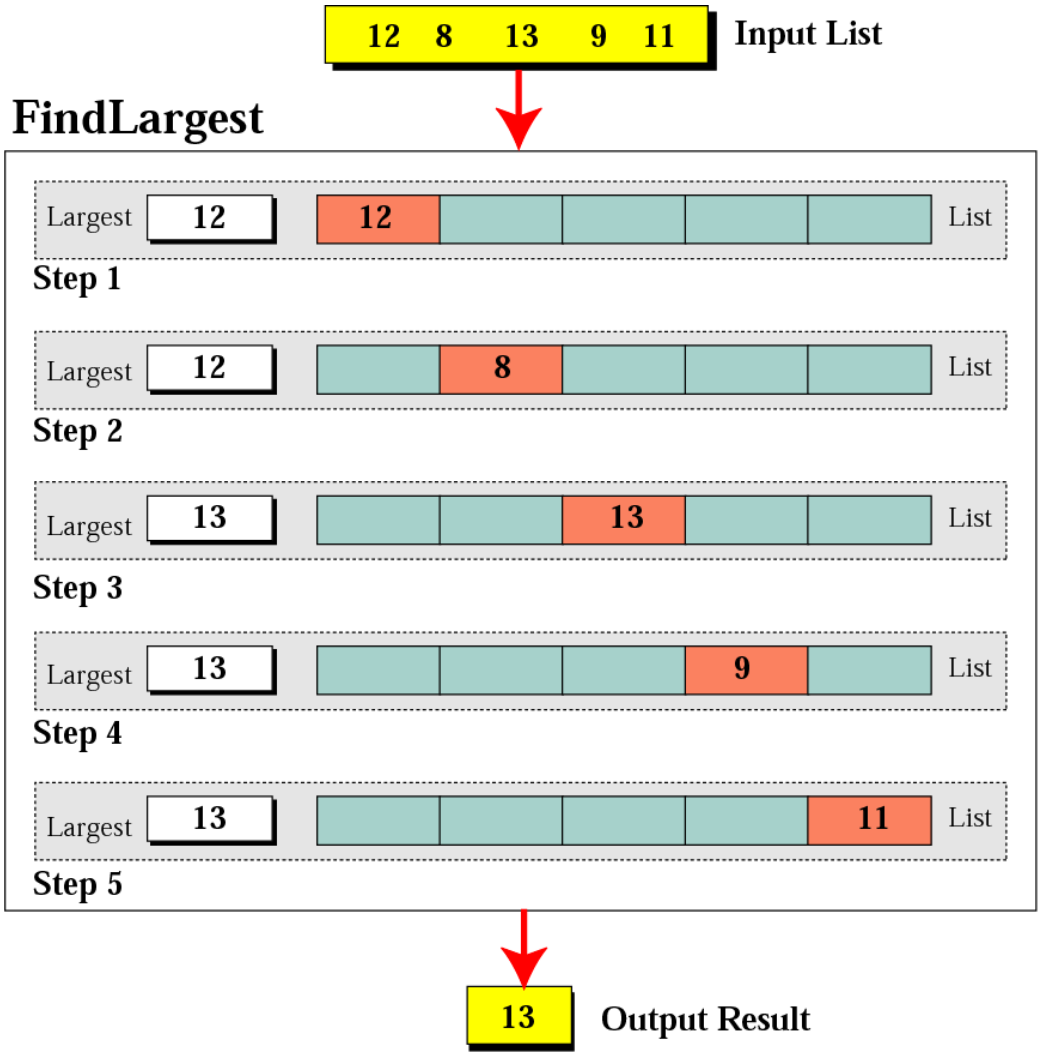
ขั้นที่ 4: ตรวจสอบอินพุทตัวที่ 4 (คือ 9) ซึ่งมีค่าน้อยกว่าค่าใน Largest จึงไม่ต้องเปลี่ยนแปลงอะไร ไปทำขั้นตอนที่ 5 ได้เลย

ขั้นที่ 5: ตรวจสอบอินพุทตัวที่ 5 ก็ไม่มีอะไรเปลี่ยนแปลง

ผลลัพธ์: อัลกอริธึมส่งผลลัพธ์เป็นค่าของตัวแปร Largest คือ 13

รูปที่ 8-2

การหาเลขที่มีค่ามากที่สุด ระหว่างเลขจำนวนเต็ม 5 จำนวน



8.1.3 กำหนดการกระทำ (actions) ให้ชัดเจนมากขึ้น

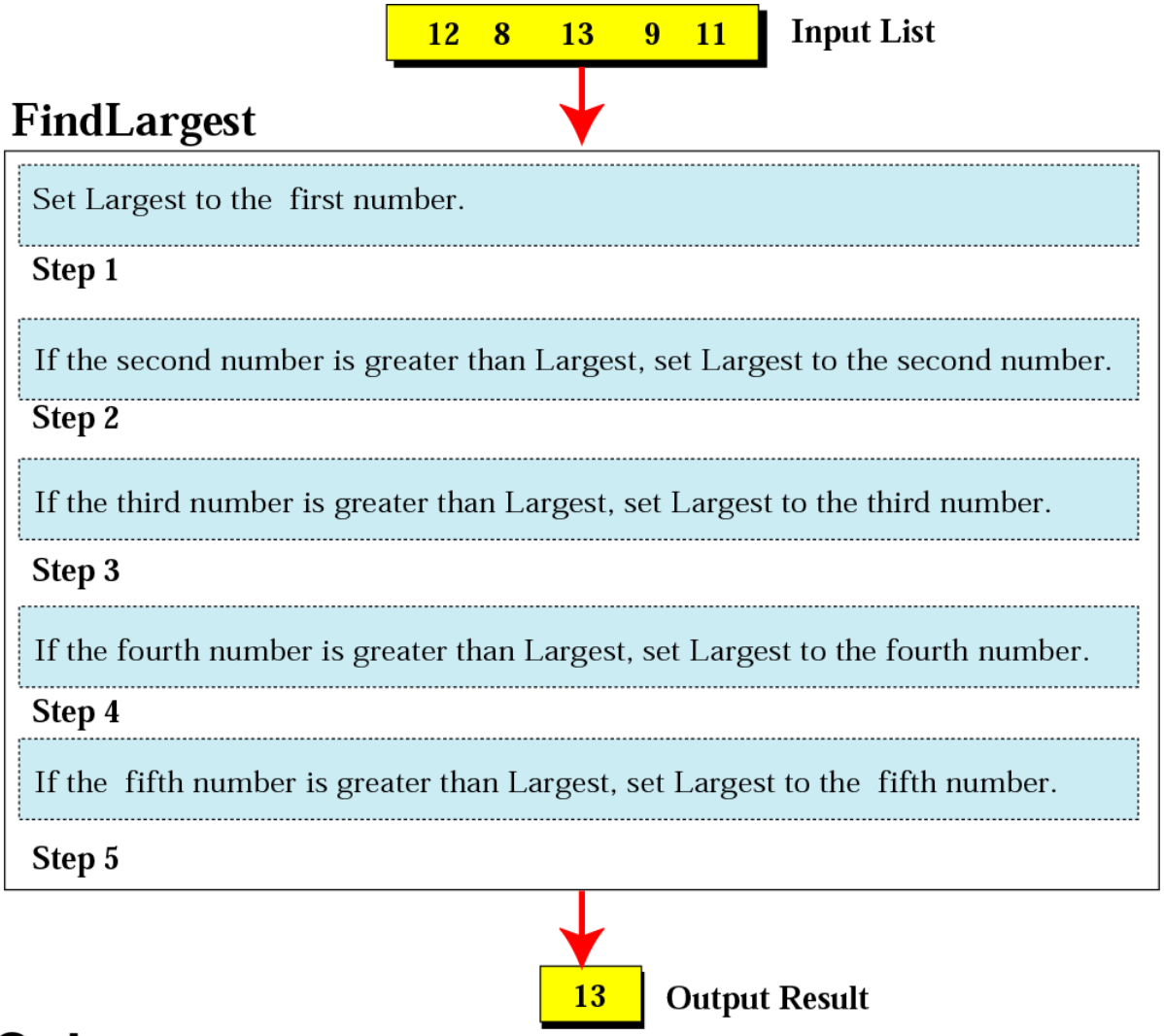
ในรูปที่ 8.2 ไม่ได้ระบุว่าในแต่ละขั้นตอนนั้นต้องทำอะไร เราสามารถเพิ่มรายละเอียดได้ดังนี้

ขั้นที่ 1: กำหนดให้ตัวแปร Largest เท่ากับอินพุทตัวแรก

ขั้นที่ 2 ถึง ขั้นที่ 5: เปรียบเทียบค่าที่เก็บอยู่ใน Largest กับค่าอินพุทตัวที่กำลังตรวจสอบอยู่ ถ้าตัวเลขที่กำลังตรวจสอบมีค่ามากกว่าค่าที่อยู่ใน Largest ให้กำหนดค่าตัวแปร Largest เป็นค่าที่กำลังตรวจสอบ (ตามรูปที่ 8.3)

รูปที่ 8-3

นิยามการกระทำในอัลกอริธึม FindLargest

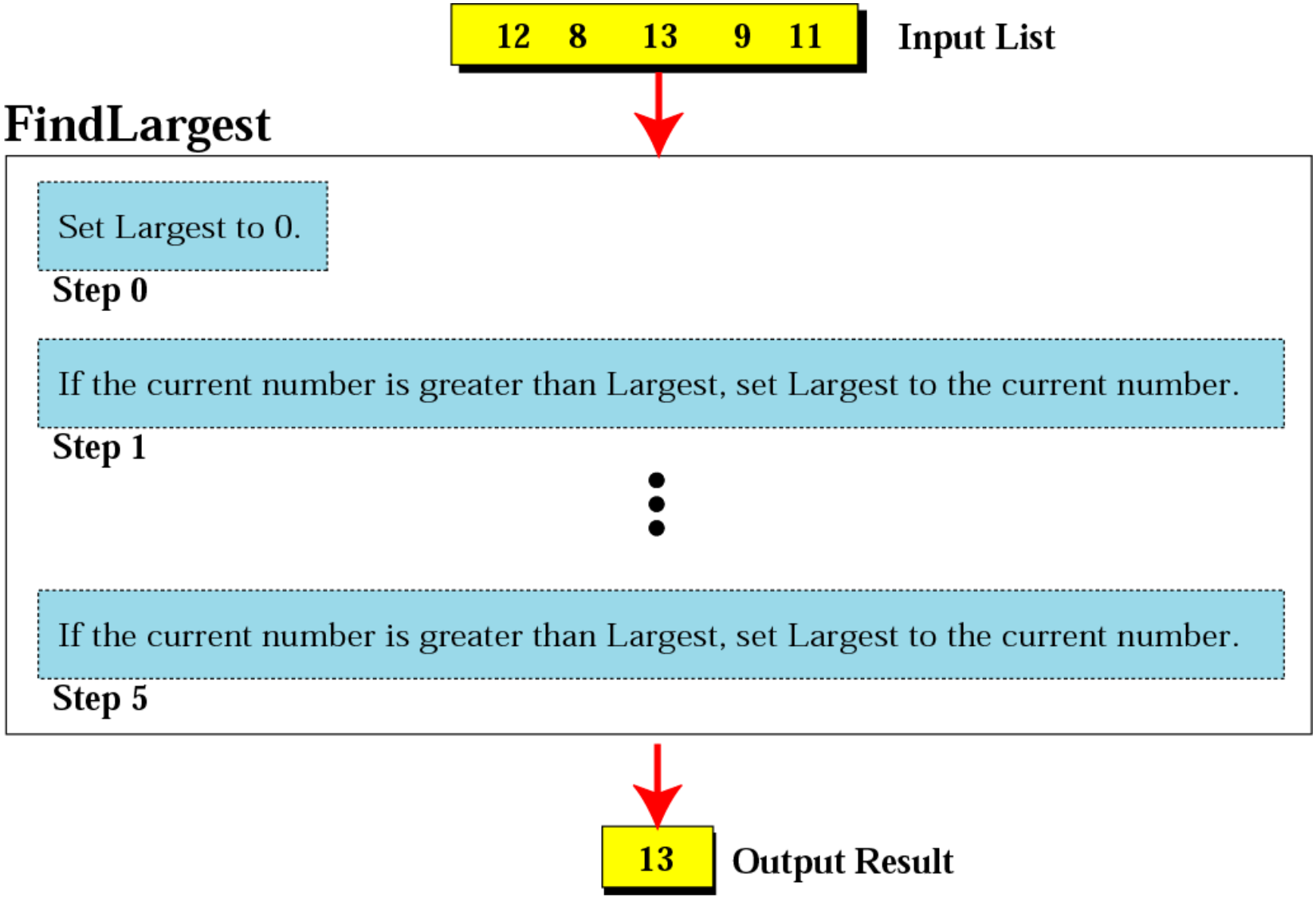


8.1.4 เพิ่มรายละเอียดให้มากขึ้น (Refinement)

อัลกอริธึมที่ผ่านมาจำเป็นที่จะต้องเพิ่มรายละเอียดให้มากขึ้นอีก เพื่อให้สามารถเปลี่ยนไปเป็นโปรแกรมได้โดยง่าย อย่างไรก็ดี ยังมีปัญหาหลักอยู่ 2 ประการคือ (1) การกระทำในขั้นที่ 1 แตกต่างจากการกระทำในขั้นที่ 2-5 และ (2) คำสั่งกระทำที่ใช้ในขั้นที่ 1 ก็แตกต่างจากจากขั้นที่ 2-5 เราสามารถปรับอัลกอริธึมเพื่อขจัดปัญหาทั้งสองประการได้ โดยการเปลี่ยนข้อความในขั้นที่ 2-5 “ถ้าตัวเลขที่กำลังตรวจสอบอยู่มีค่ามากกว่าค่าใน Largest แล้ว ให้กำหนดค่า Largest เท่ากับค่าตัวเลขที่กำลังตรวจสอบ”

เหตุผลที่ขั้นที่ 1 แตกต่างจากขั้นที่ 2-5 เพราะตัวแปร Largest ยังไม่ได้มีการกำหนดค่าเริ่มต้น ถ้าเรากำหนดค่าเริ่มต้นของ Largest = 0 (ไม่มีเลขจำนวนเต็มบวกใดน้อยกว่า 0) แล้วขั้นที่ 1 จะเหมือนกับขั้นที่ 2-5 ถ้าเราเพิ่มขั้นที่ 0 เพื่อกำหนดค่า Largest = 0 ก่อนเริ่มทำงาน จะได้ผลดังรูปที่ 8.4

รูปที่ 8-4 อัลกอริธึม FindLargest ที่ละเอียดมากขึ้น

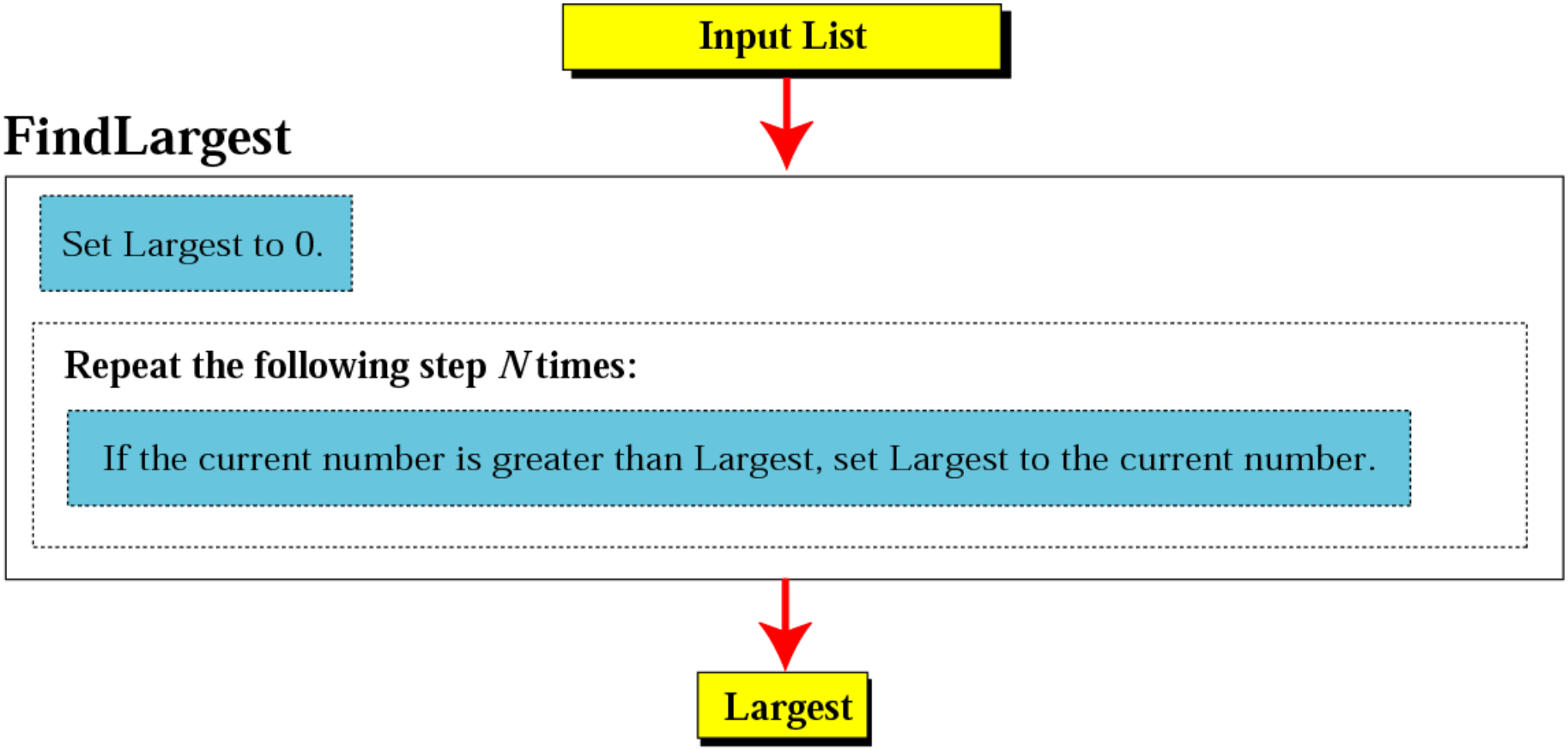


8.1.5 การทำให้เป็นกรณีทั่วไป (Generalization)

เราสามารถทำให้อัลกอริธึมทำงานได้กับเลขจำนวนเต็มบวกกี่จำนวนก็ได้ สมมติมีเลขจำนวนเต็มบวก N ตัว (N อาจจะเป็น 1,000 หรือ 10,000 หรือ 1,000,000 หรือมากกว่าก็ได้) ถ้าดูแบบผิวเผินเราอาจจะบอกว่าก็เขียนตามรูปที่ 8.4 โดยเขียนซ้ำๆเท่ากับ N ขั้นตอน เป็นไปได้แต่จะทำให้อัลกอริธึมยาวมาก มีวิธีที่ดีกว่า คือเราบอกให้คอมพิวเตอร์ทำซ้ำๆจำนวน N ครั้งดังรูปที่ 8.5

รูปที่ 8-5

กรณีทั่วไปของอัลกอริธึม FindLargest



8.2

โครงสร้าง 3 รูปแบบ

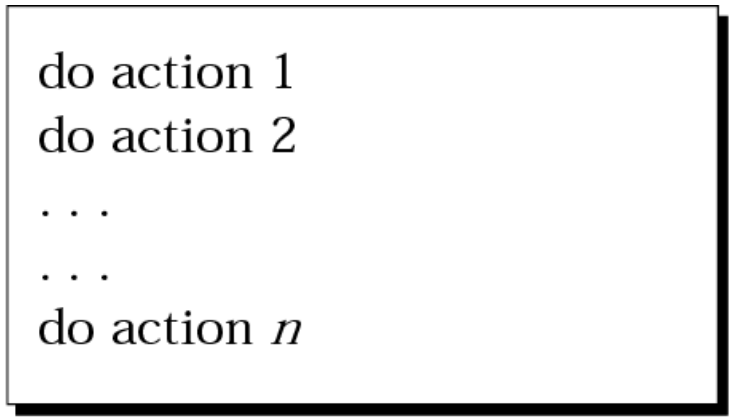
THREE CONSTRUCTS

8.2.1 โครงสร้างการเขียนอัลกอริธึม 3 รูปแบบ (Three Constructs)

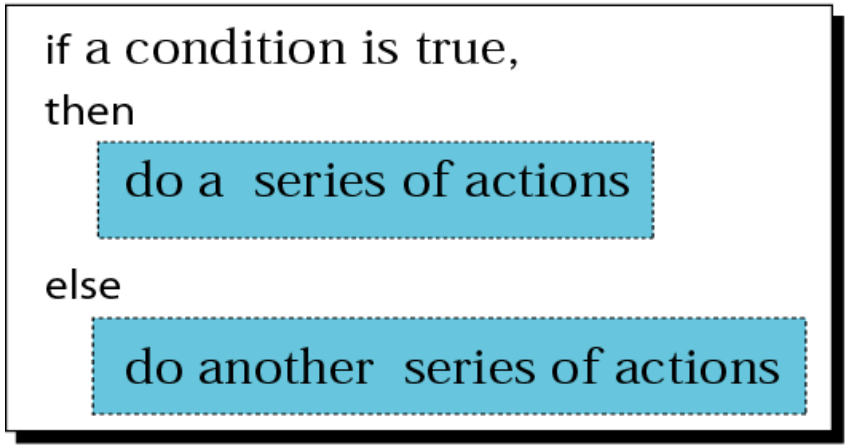
นักคอมพิวเตอร์ได้กำหนดโครงสร้าง 3 รูปแบบเพื่อใช้ในการเขียนโปรแกรมแบบโครงสร้างหรืออัลกอริธึม แนวคิดคือคอมพิวเตอร์โปรแกรมใดๆจะสามารถเขียนได้โดยใช้โครงสร้างเพียง 3 รูปแบบนี้ โครงสร้างทั้งสามรูปแบบได้แก่: (1) การทำตามลำดับ (sequence) (2) การตัดสินใจ (decision หรือ selection) และ (3) การทำซ้ำๆ (repetition) ดังรูปที่ 8.6

ได้มีการพิสูจน์แล้วว่าในการเขียนโปรแกรมคอมพิวเตอร์นั้นใช้โครงสร้างแค่ 3 รูปแบบนี้ก็เพียงพอ ไม่จำเป็นต้องใช้โครงสร้างแบบอื่น การใช้โครงสร้างทั้งสามในการเขียนโปรแกรมหรืออธิบายอัลกอริธึม จะทำให้โปรแกรมหรืออัลกอริธึมเข้าใจได้ง่าย แก้ไขได้ง่าย และเปลี่ยนแปลงได้ง่าย

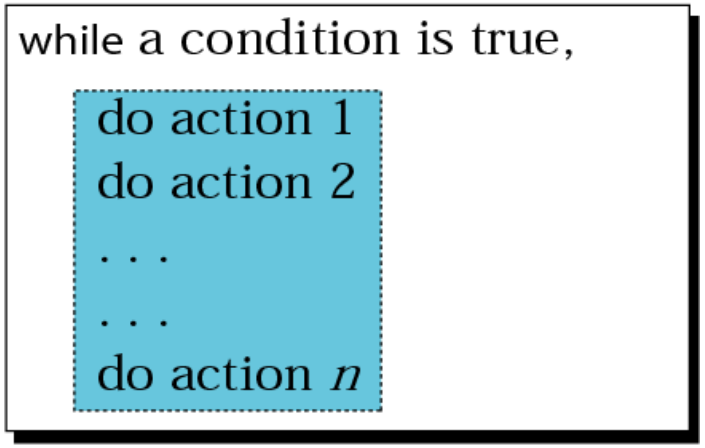
รูปที่ 8-6 โครงสร้าง 3 แบบ



a. Sequence



b. Decision



c. Repetition

8.3

การแทนอัลกอริธึม

ALGORITHM REPRESENTATION

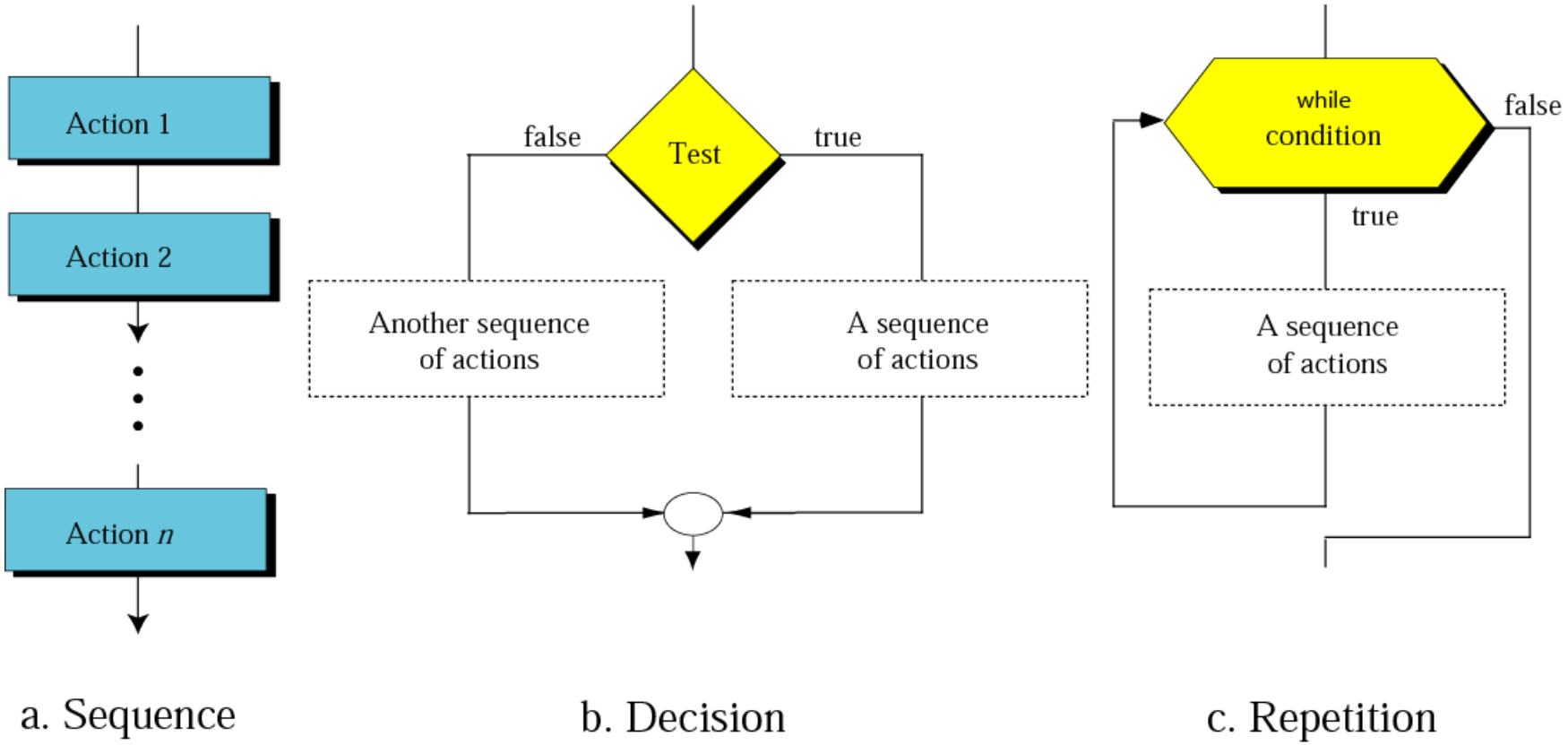
ปกติการแทนอัลกอริธึมเรานิยมใช้รูปภาพ เพราะเข้าใจง่ายและมีความชัดเจน โดยทั่วไปเราใช้ 2 รูปแบบคือ

8.3.1 ผังงาน (flowchart): มีลักษณะคล้ายรูปภาพ โดยจะเน้นภาพรวมของตรรกะของอัลกอริธึมมากกว่ารายละเอียด ผังงานจะแสดงถึงขั้นตอนการทำงานของอัลกอริธึมตั้งแต่ต้นจนจบ รายละเอียดของผังงานจะกล่าวถึงในตอนหลังๆ **(รูปที่ 8.8)**

8.3.2 รหัสเทียม (pseudocode): เป็นการแทนอัลกอริธึมโดยใช้ภาษาเขียน มีลักษณะคล้ายภาษาอังกฤษที่ใช้กันโดยทั่วไป แต่ระดับความละเอียดนั้นแต่ละคนก็ใช้แตกต่างกันไป ไม่มีมาตรฐานใดกำหนดชัดเจน บางคนเขียนคล้ายๆภาษาอังกฤษ บางคนเขียนโดยใช้รูปแบบคล้ายภาษาปาสคาล รายละเอียดจะอธิบายในบทต่อๆไป **(รูปที่ 8.8)**

รูปที่ 8-7

ผังงานของโครงสร้างทั้ง 3 แบบ



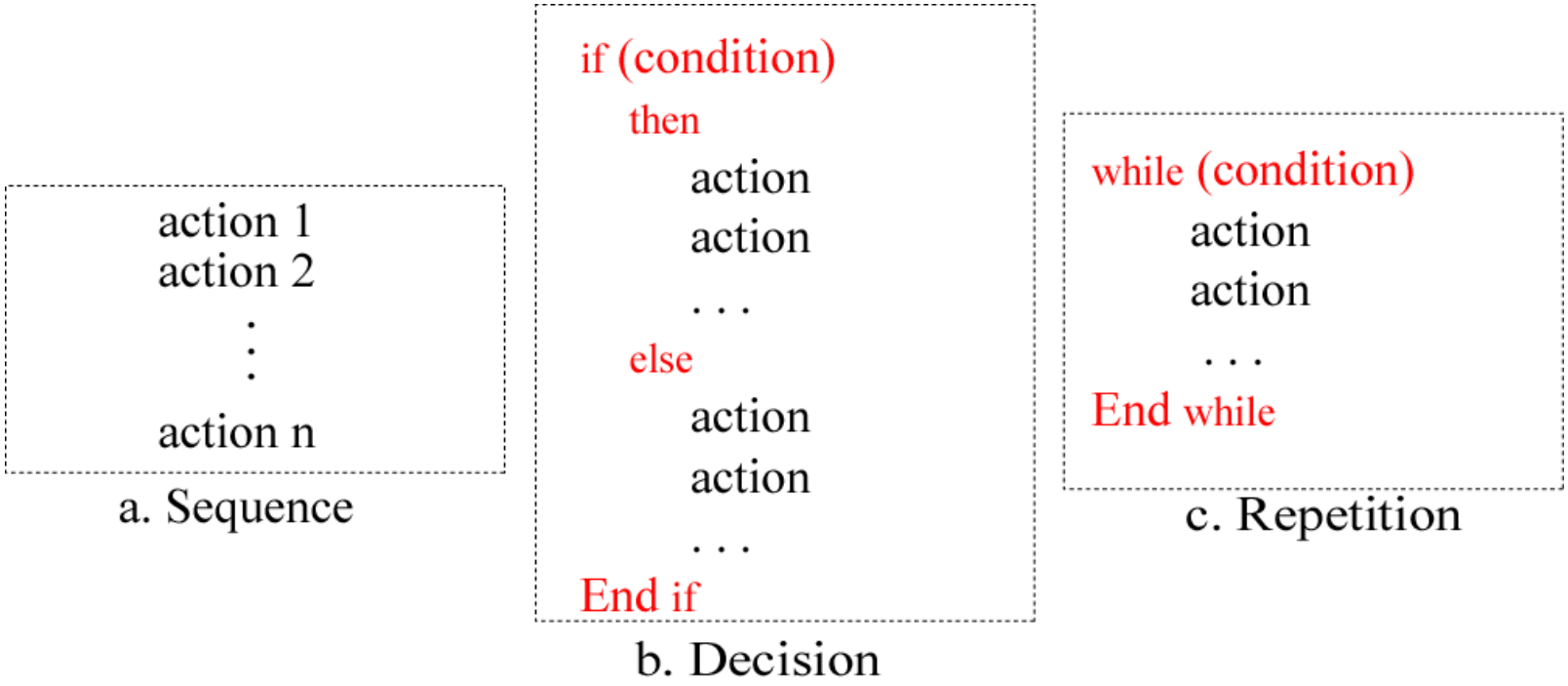
a. Sequence

b. Decision

c. Repetition

รูปที่ 8-8

Pseudocode สำหรับโครงสร้าง 3 แบบ



ตัวอย่างที่ 1

จงเขียน algorithm ในรูป pseudocode เพื่อหาค่าเฉลี่ยของเลข 2 จำนวน

วิธีทำ

Algorithm 8.1: Average of two

Input: Two numbers

1. Add the two numbers
2. Divide the result by 2
3. Return the result by step 2

ตัวอย่างที่ 2

จงเขียนอัลกอริธึมเพื่อเปลี่ยนค่าคะแนนให้เป็นเกรด ผ่าน/ไม่ผ่าน

วิธีทำ

Algorithm 8.2: Pass/No pass Grade

Input: One number

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

Return the grade

ตัวอย่างที่ 3

จงเขียนอัลกอริธึมเพื่อเปลี่ยนค่าคะแนนเป็นเกรดตัวอักษร A, B..

วิธีทำ

Algorithm 8.3: Letter Grade

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

/* ต่อหน้าถัดไป */

Algorithm 8.3: Letter Grade (ต่อ)

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

Algorithm 8.3: Letter Grade (ต่อ)

5. If (the number is less than 60) Then

5.1 Set the grade to “F”

End If

6. Return the grade

ตัวอย่างที่ 4

จงเขียนอัลกอริธึมเพื่อหาค่าตัวเลขที่มากที่สุดจากชุดของตัวเลขที่ไม่ทราบว่ามีกี่จำนวน

วิธีทำ

Algorithm 8.4: Find Largest

Input: A list of positive integers

Set Largest to 0

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

Return Largest

ตัวอย่างที่ 5 จงเขียนอัลกอริธึมเพื่อหาเลขที่มากที่สุดจากเลข 1000จำนวน

วิธีทำ

Algorithm 8.5: Find largest of 1000 numbers

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

Return Largest

8.4

นิยามอัลกอริธึมที่เป็นทางการมากขึ้น

MORE FORMAL DEFINITION

นิยาม: อัลกอริธึมคือเซตของขั้นตอนการแก้ปัญหาที่จัดเรียงลำดับอย่างชัดเจน (ordered set) ไม่คลุมเครือ (unambiguous steps) ก่อให้เกิดผลลัพธ์(produce a result) ภายในเวลาที่จำกัด (terminate in a finite time)

Ordered set: หมายความว่าคำสั่งทั้งหมดต้องเรียงลำดับ และ well-defined

Unambiguous steps: หมายความว่าแต่ละขั้นตอนต้องมีความหมายชัดเจน ไม่เปิดโอกาสให้ตีความได้มากกว่า 1 อย่าง

Produce result: หมายความว่าอัลกอริธึมจะต้องสร้างผลลัพธ์ที่เป็นรูปธรรม เช่นผลที่เป็นตัวเลข หรือผลที่เกิดจากการพิมพ์ เป็นต้น

Terminate in a finite time: หมายความว่าอัลกอริธึมจะต้องมีการจบสิ้นหรือจะต้องหยุดในที่สุด

8.5

อัลกอริธึมย่อย

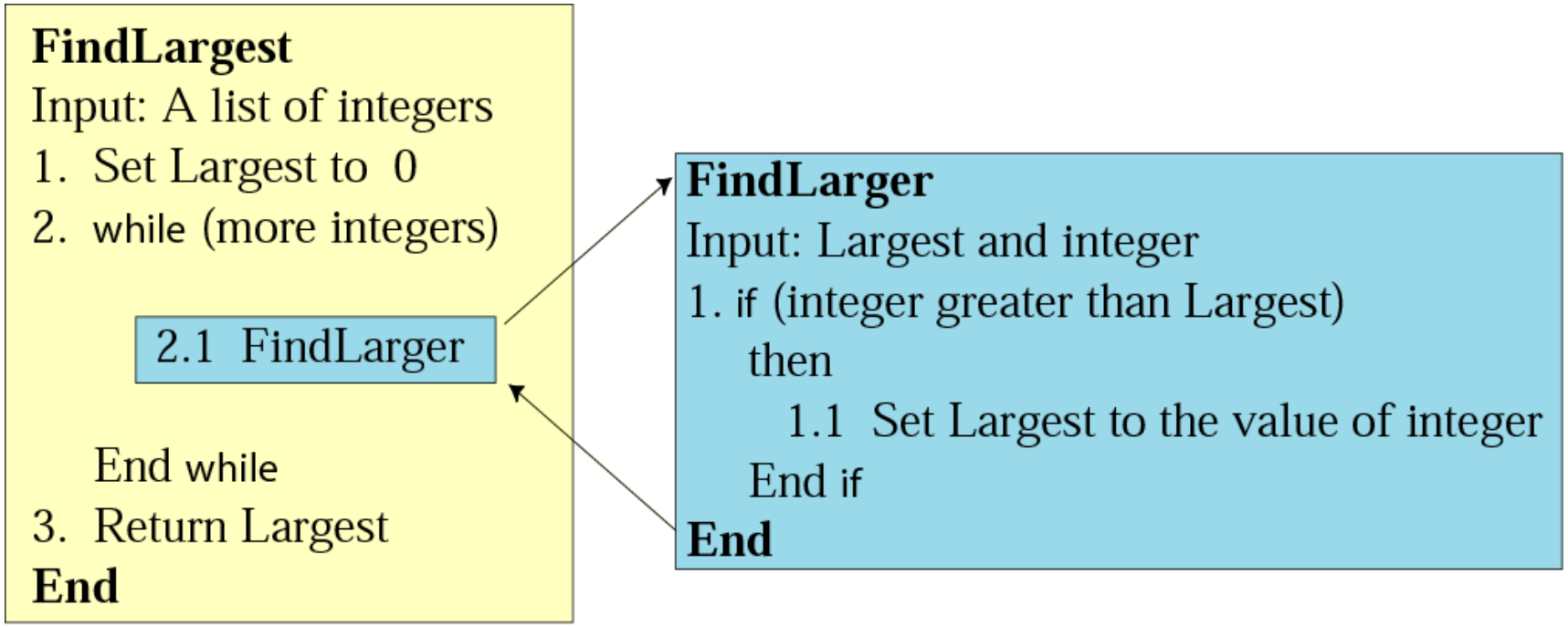
SUBALGORITHMS

หลักการของการเขียนโปรแกรมเชิงโครงสร้าง (structured programming) คือการแบ่งอัลกอริธึมออกเป็นหน่วยย่อยๆ แต่ละหน่วยก็สามารถแบ่งย่อยๆลงไปได้อีกจนกระทั่งแต่ละส่วนเป็น intrinsic คือเมื่ออ่านแล้วเข้าใจได้ทันที ทำงานเพียงอย่างเดียว แต่ละส่วนย่อยที่ถูกแบ่งนี้มีชื่อเรียกหลายแบบเช่น subprogram, subroutines, procedures, functions, methods, modules เป็นต้น

ตัวอย่าง: เราสามารถแบ่งอัลกอริธึม FindLargest ออกเป็น subalgorithm โดยตั้งชื่อว่า “FindLarger” เพื่อให้ทำหน้าที่หาตัวเลขที่มีค่ามากกว่าระหว่างเลข 2 จำนวนใดๆ อัลกอริธึมย่อย “FindLarger” จะถูกเรียกเพื่อให้ทำงานซ้ำๆในแต่ละรอบ ดังรูปที่ 8.9

รูปที่ 8-9

Concept of a subalgorithm



Concept of a subalgorithm

- อัลกอริธึม FindLargest จะทำการ execute แต่ละคำสั่งจนกระทั่งพบกับชื่อของอัลกอริธึมอื่น

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

ตัวอย่างอัลกอริธึมพื้นฐาน

BASIC ALGORITHMS

ตัวอย่างที่ 1: การหาผลรวมของเลขหลายๆจำนวน : SUMMATION

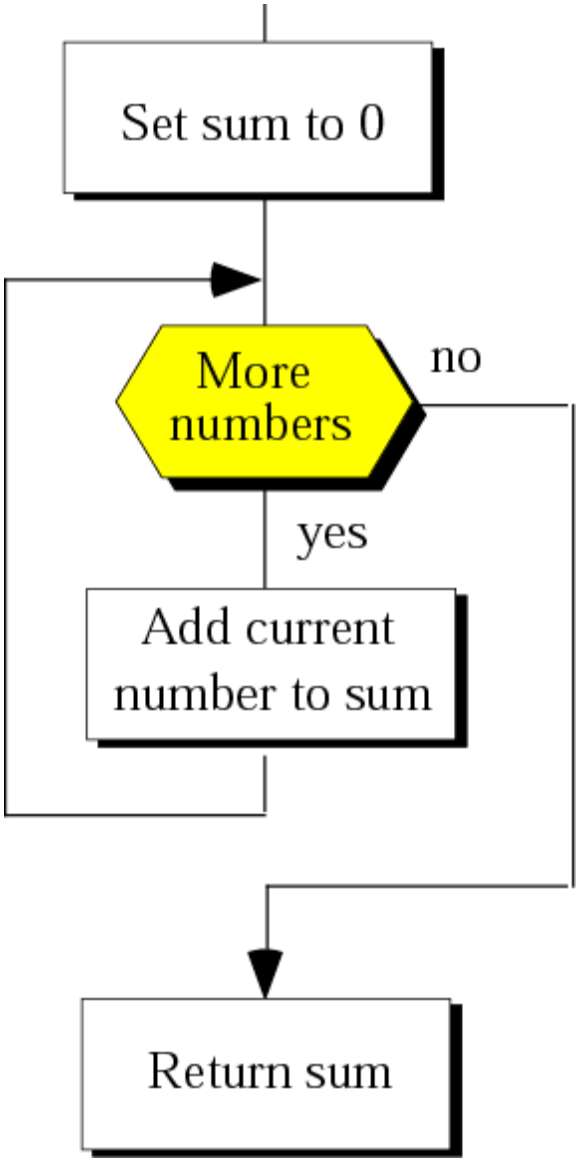
การหาผลรวมของเลขหลายๆจำนวนนับเป็นการกระทำพื้นฐานที่ใช้แก้ปัญหาหลายๆอย่างเช่นการหาค่าเฉลี่ย การหายอดรวมทางบัญชี การหารยอดรวมของรายรับ-จ่าย เป็นต้น ขั้นตอนการทำงานเป็นดังนี้

1. กำหนดค่าเริ่มต้นของ sum ให้เท่ากับ 0
2. Loop: ในแต่ละรอบ บวกเลขที่รับเข้ามากับค่าใน sum
3. ส่งผลลัพธ์สุดท้ายกลับหลังออกจาก Loop

แทนการทำงานอัลกอริธึม SUMMATION ดังรูปที่ 8.10

รูปที่ 8-10

Summation



ตัวอย่างที่ 2: การหาผลคูณของเลขหลายๆจำนวน : PRODUCT

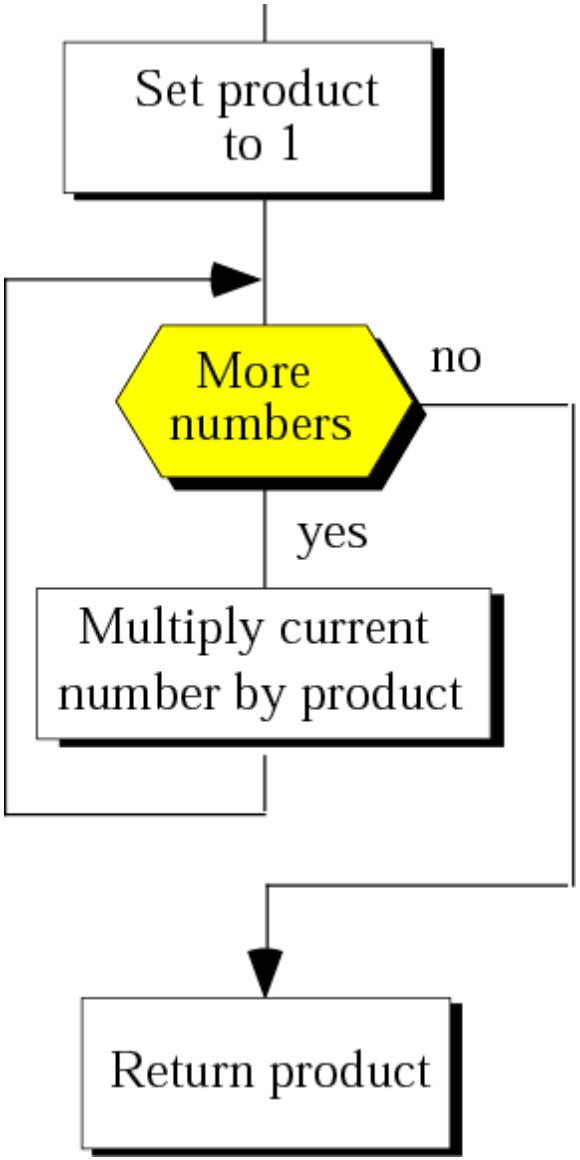
การหารผลคูณของเลขหลายๆจำนวนมีขั้นตอนการทำงานดังนี้

1. กำหนดค่าเริ่มต้นของผลคูณให้เท่ากับ 1
2. Loop: ในแต่ละรอบ ให้คูณเลขที่รับเข้ามาใหม่กับผลคูณเดิม
3. ส่งผลลัพธ์กลับเมื่อออกจาก Loop

แทนอัลกอริธึมโดยผังงานดังรูปที่ 8.11

รูปที่ 8-11

Product



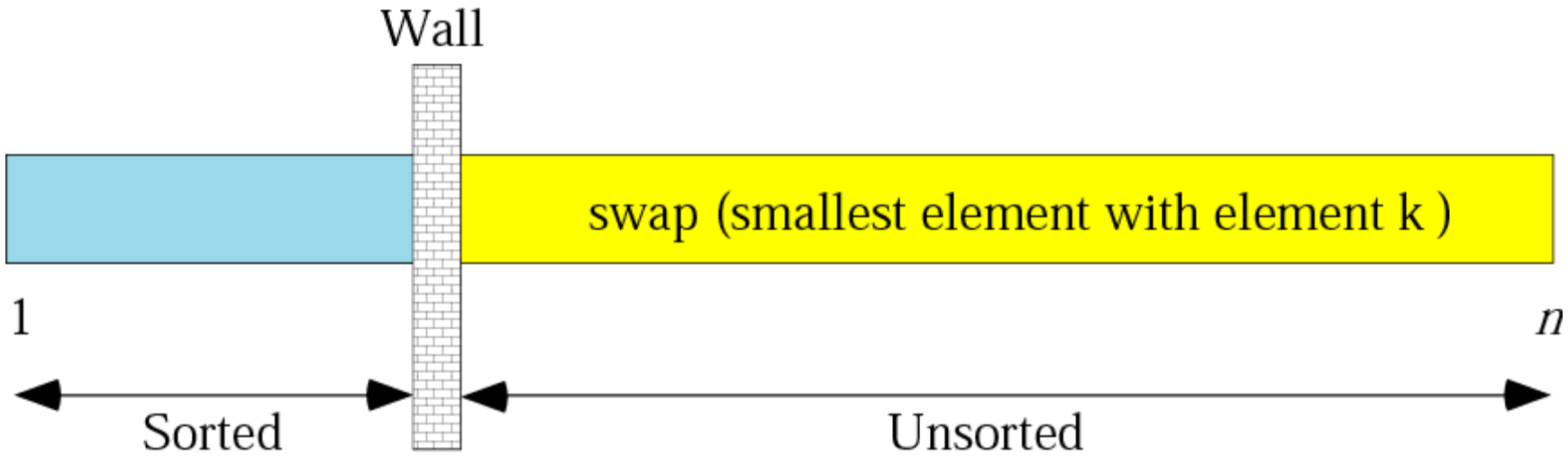
การเรียงลำดับ : SORTING

การเรียงลำดับกลุ่มของตัวเลขนับเป็นการประยุกต์ที่สำคัญยิ่งในทางคอมพิวเตอร์ วิธีการเรียงลำดับมีหลายแบบ การเลือกใช้ขึ้นอยู่กับประเภทและจำนวนของข้อมูล ตัวอย่างอัลกอริธึมที่นิยมใช้กันมากมี 3 อัลกอริธึมคือ

1. Selection Sort
2. Bubble Sort
3. Insertion Sort

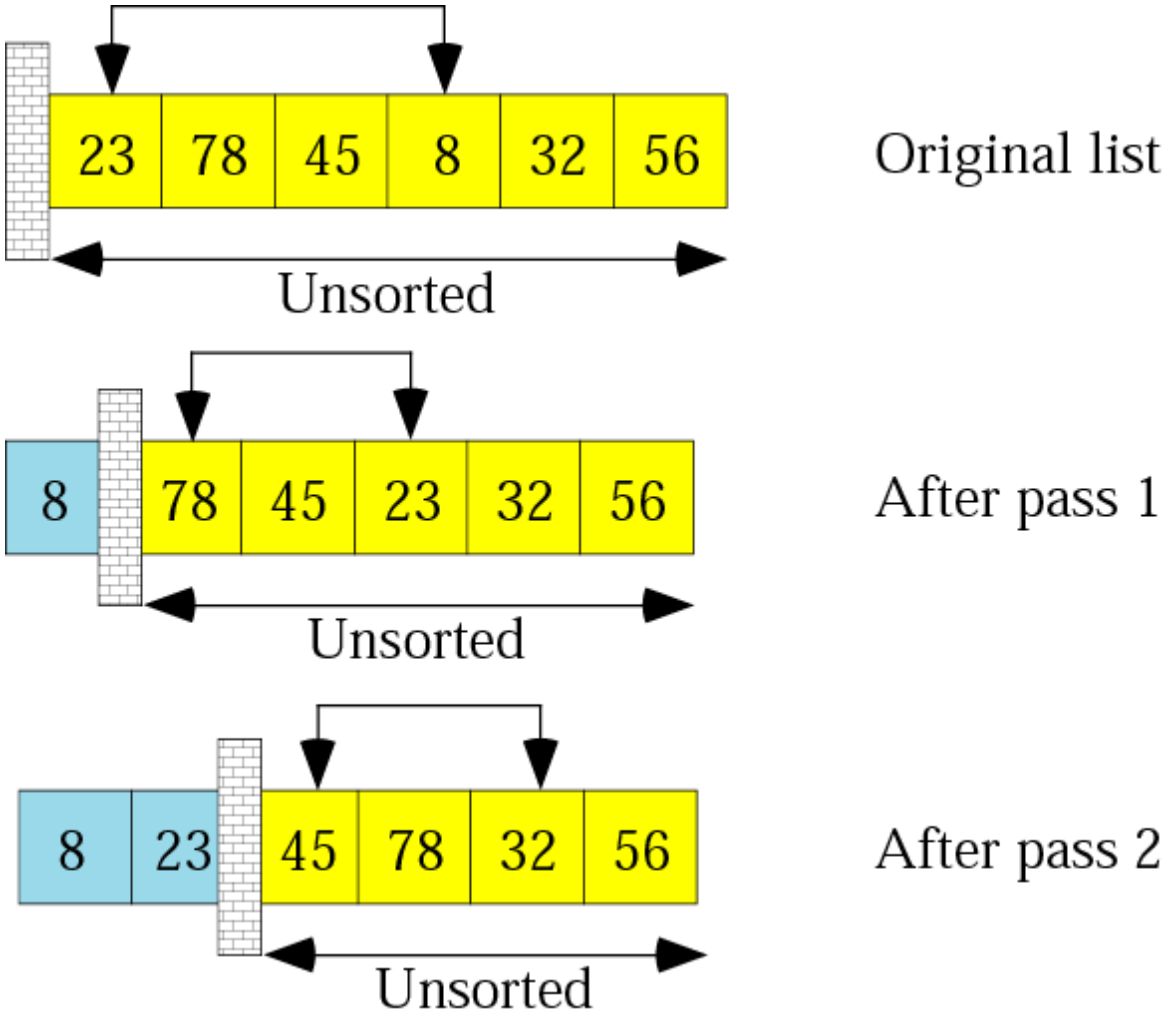
รูปที่ 8-12

Selection sort



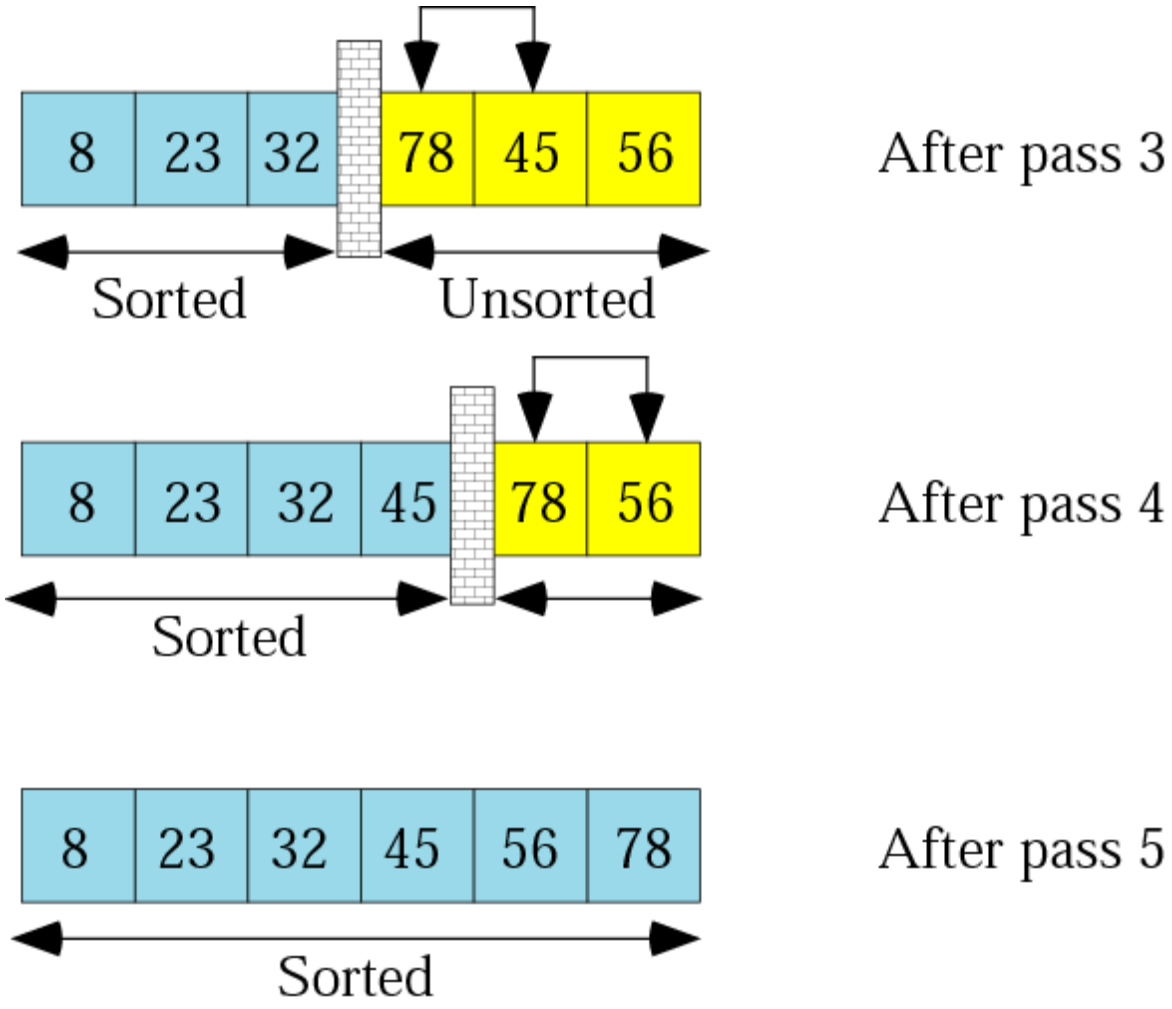
รูปที่ 8-13: ตอนที่ I

Example of selection sort



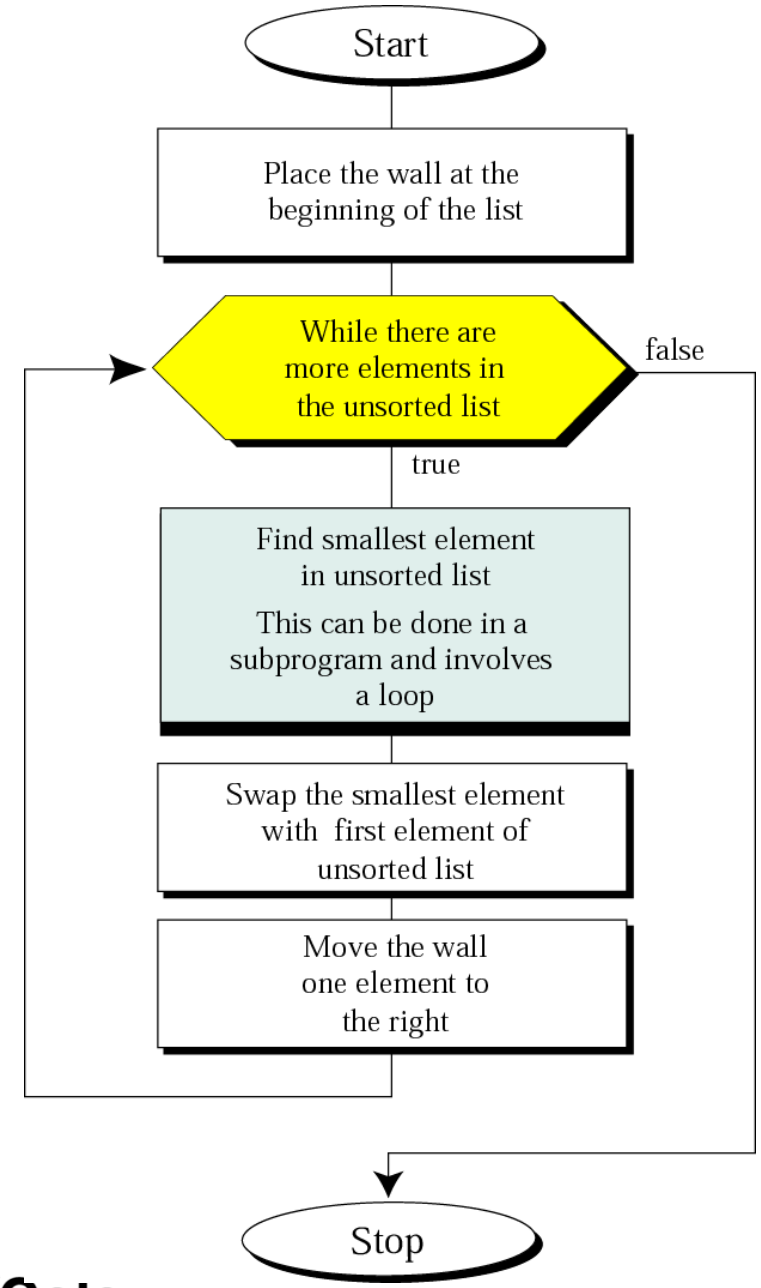
รูปที่ 8-13: ตอนที่ II

Example of selection sort



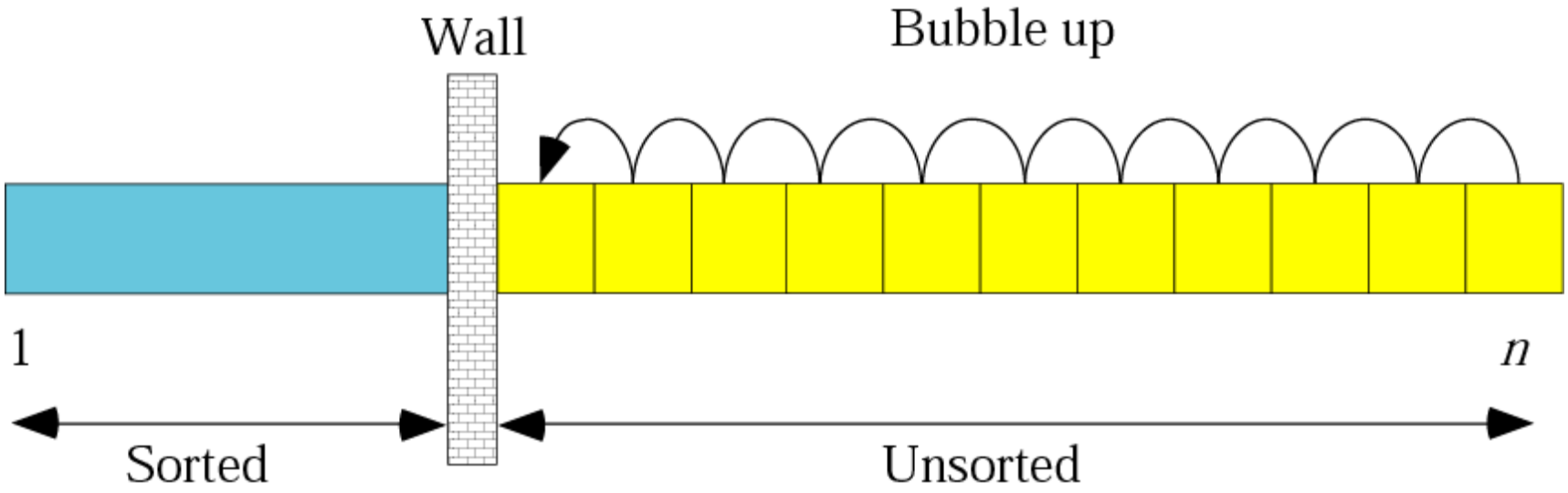
รูปที่ 8-14

Selection sort algorithm



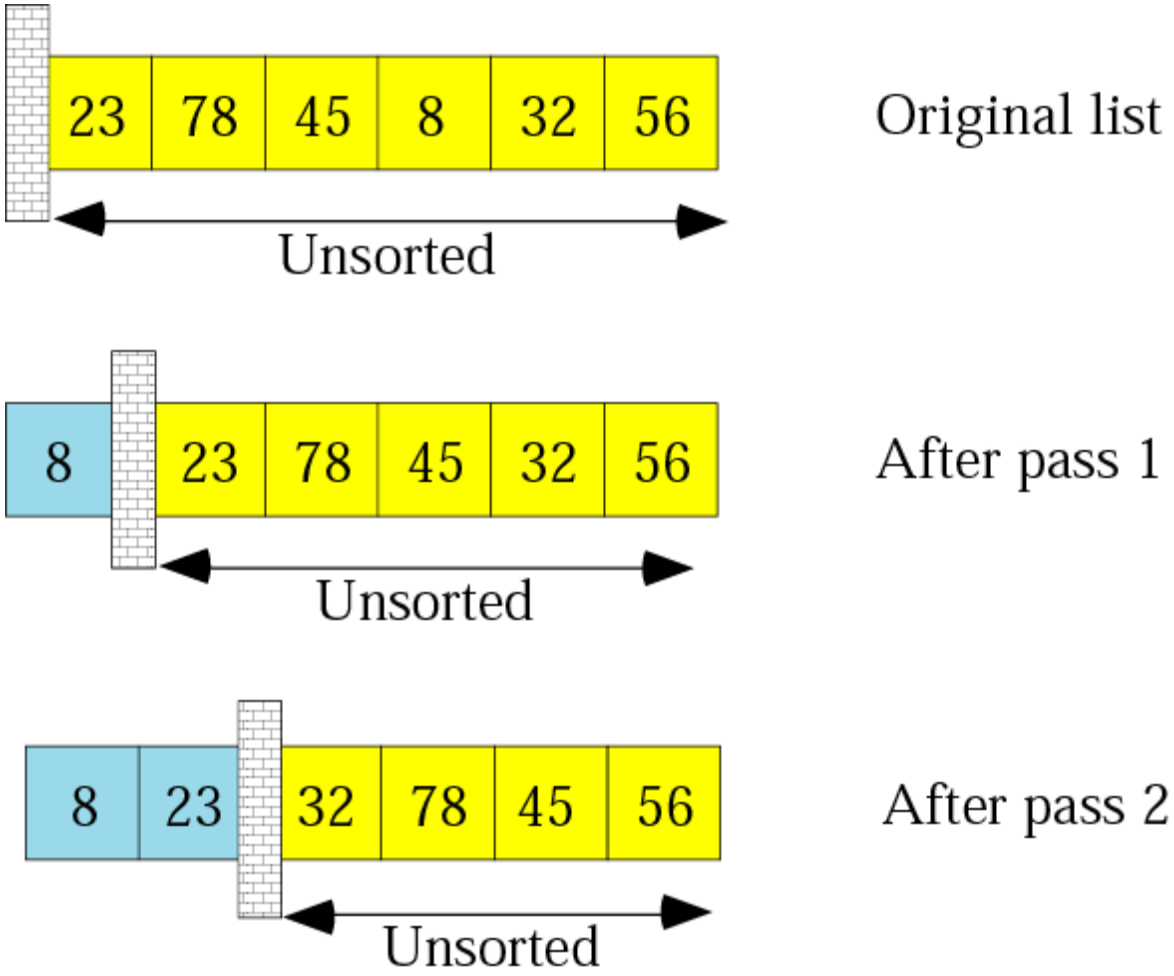
รูปที่ 8-15

Bubble sort



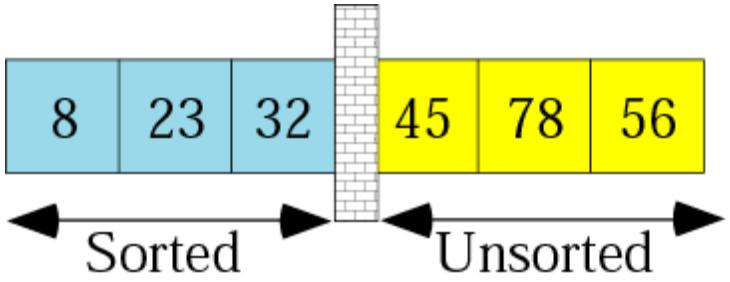
รูปที่ 8-16: ตอนที่ I

Example of bubble sort

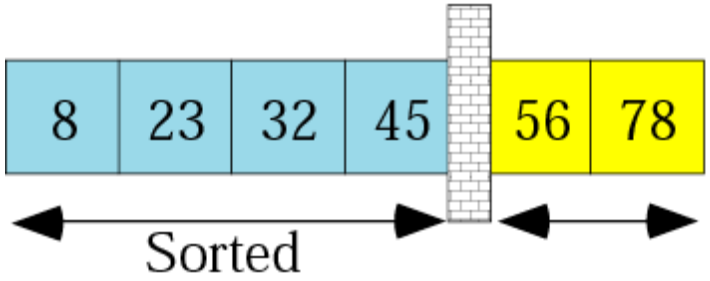


รูปที่ 8-16: ตอนที่ II

Example of bubble sort



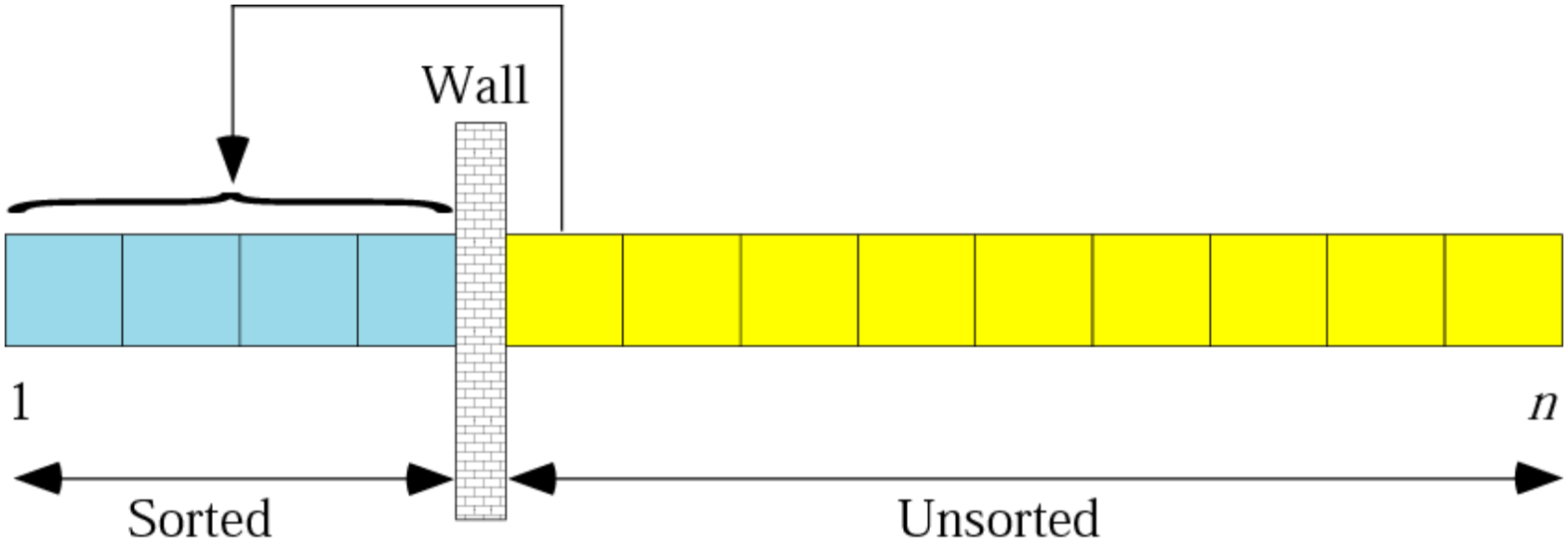
After pass 3



After pass 4
Sorted

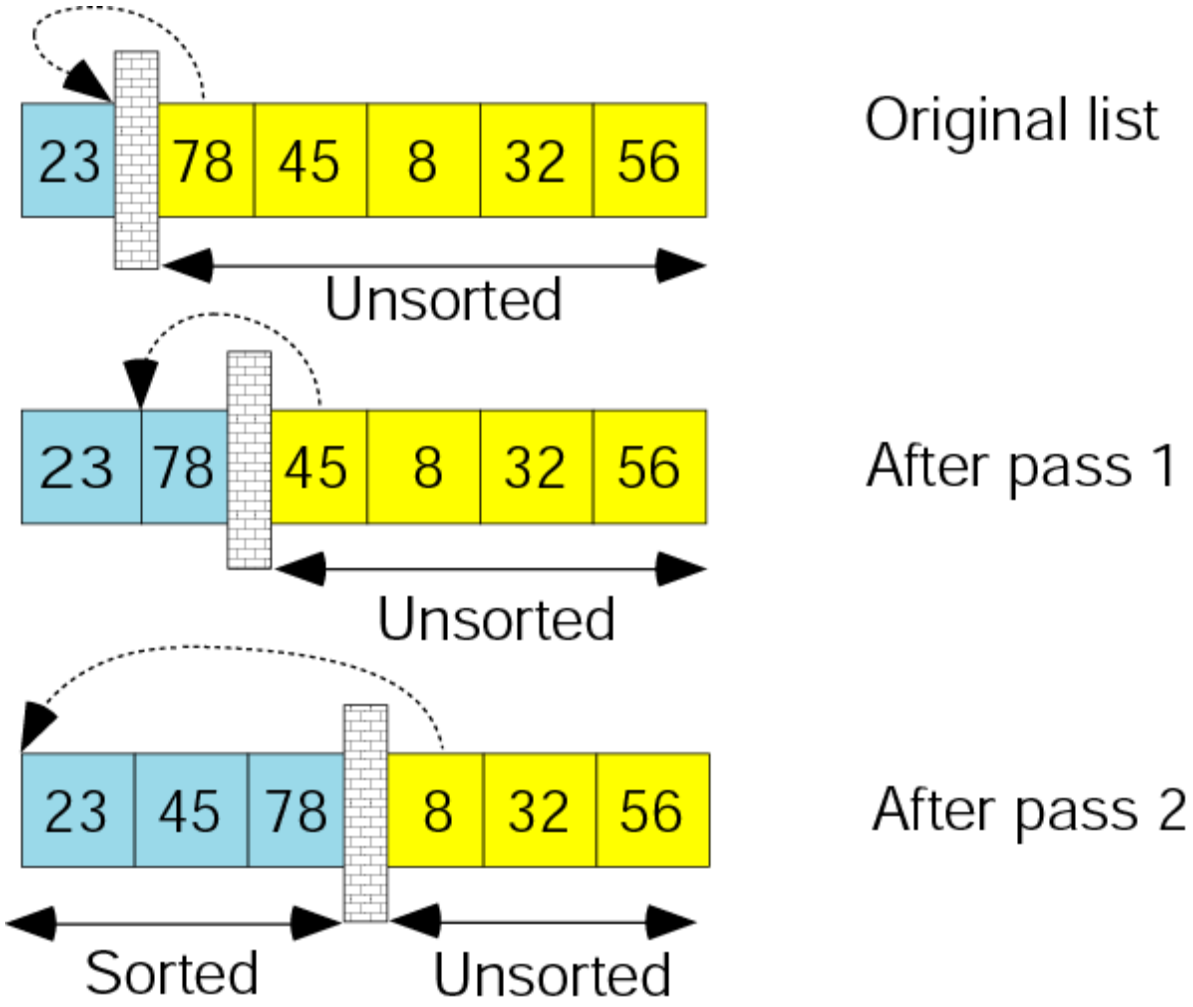
รูปที่ 8-17

Insertion sort



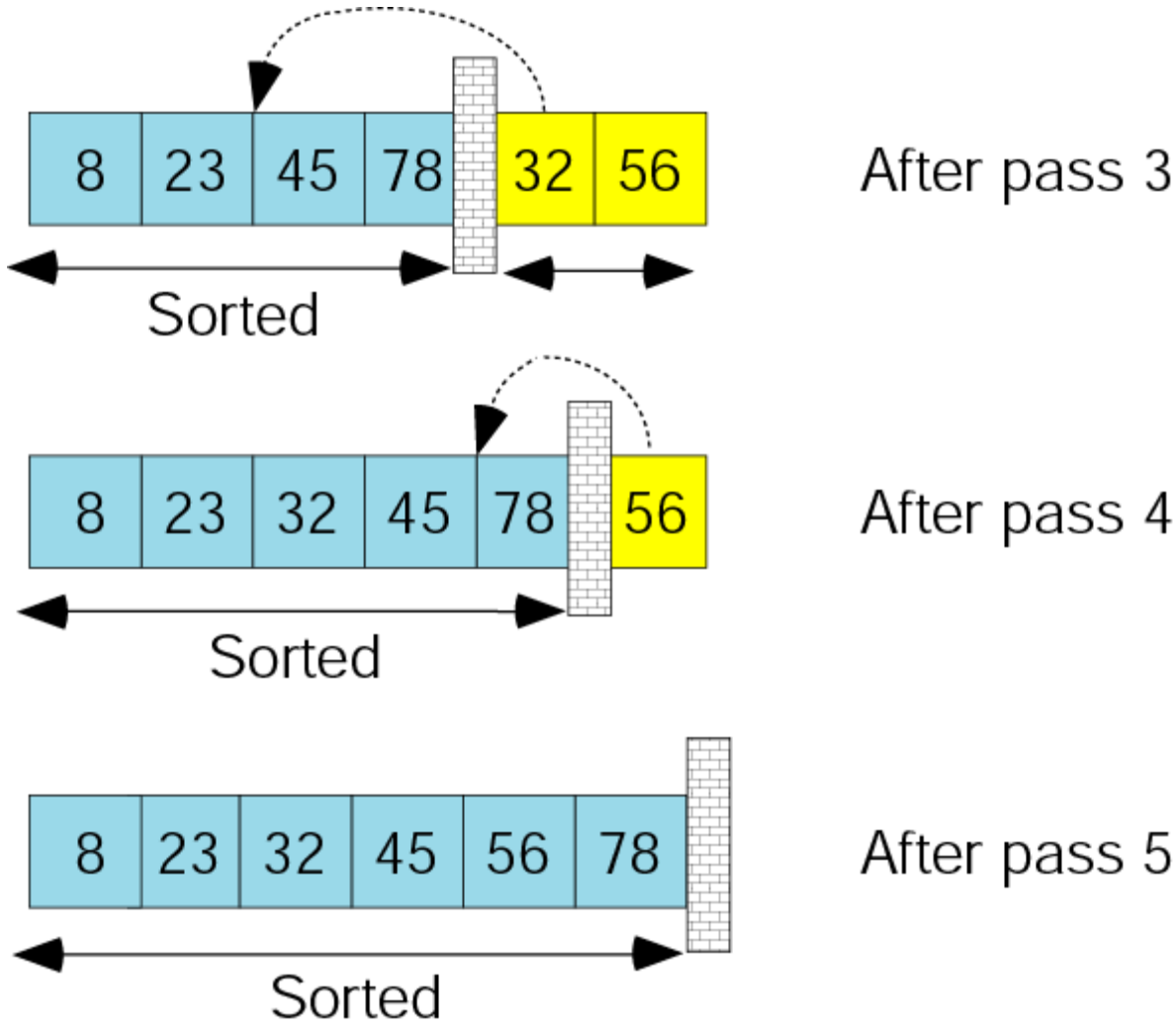
รูปที่ 8-18: ตอนที่ I

ตัวอย่างของ insertion sort



รูปที่ 8-18: ตอนที่ II

ตัวอย่างของ insertion sort



การค้นหา : SEARCHING

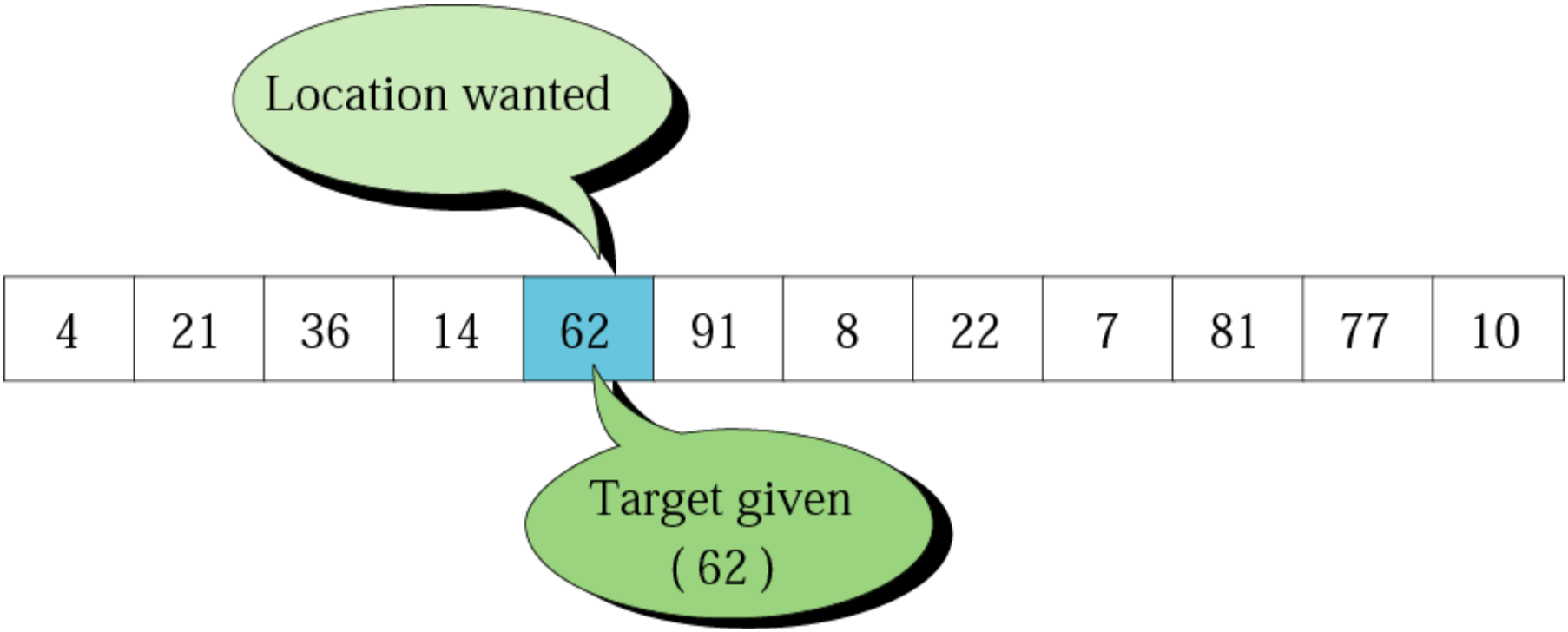
การค้นหาข้อมูลที่ต้องการจากกลุ่มของข้อมูลที่กำหนดให้ก็เป็นอีกการประยุกต์หนึ่งที่ใช้กันมาก มีวิธีการค้นหาข้อมูลหลายวิธีเช่นกัน ขึ้นอยู่กับประเภทและจำนวนของข้อมูลที่เกี่ยวข้อง วิธีการค้นหาที่ใช้กันมากได้แก่

1. Sequential Search

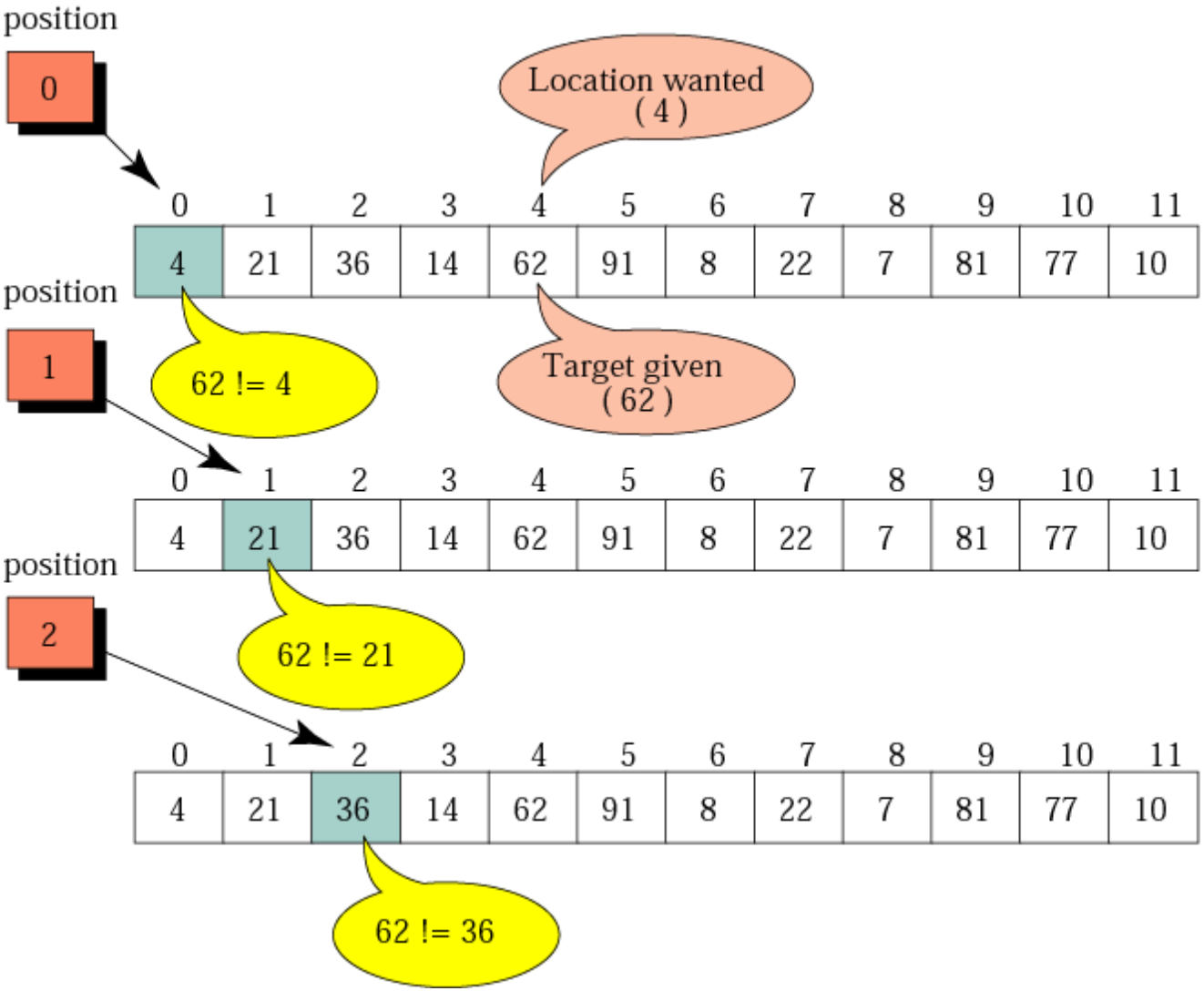
2. Binary Search

รูปที่ 8-19

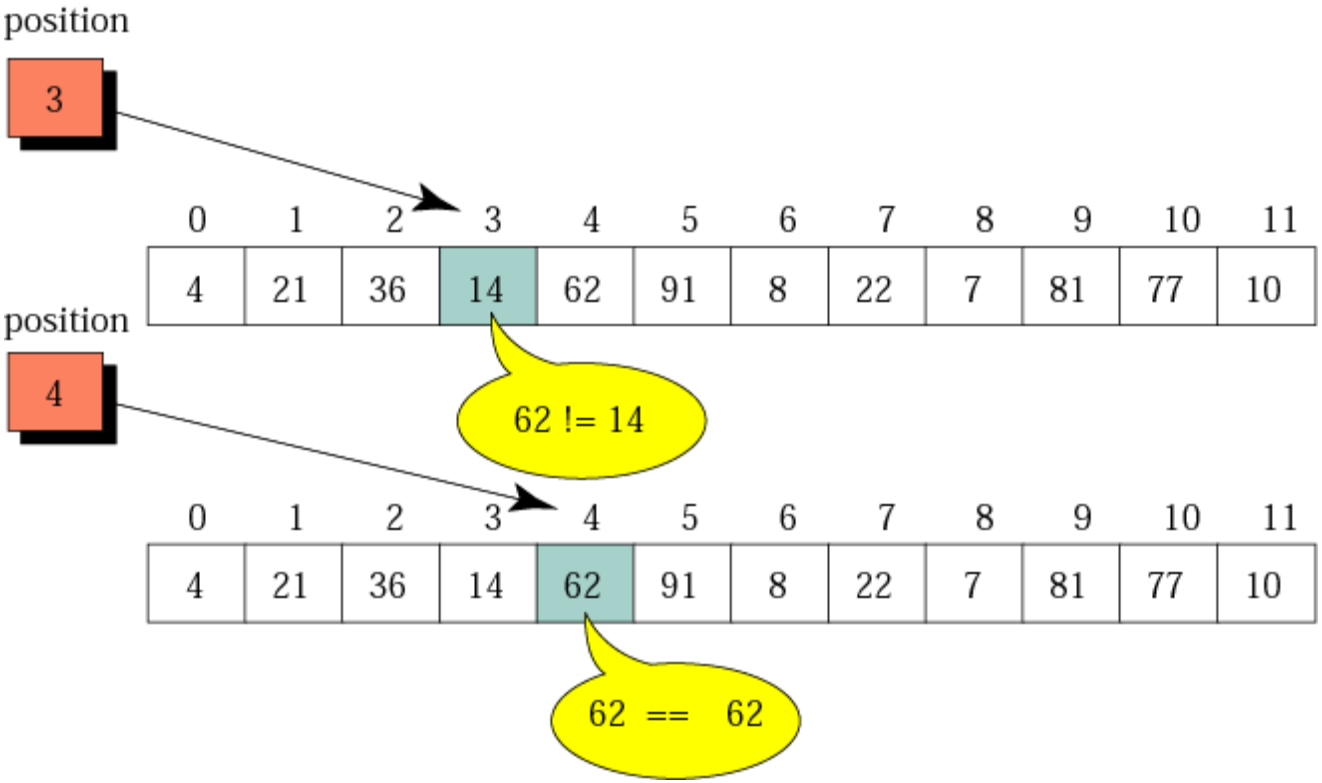
Search concept



ตัวอย่าง sequential search

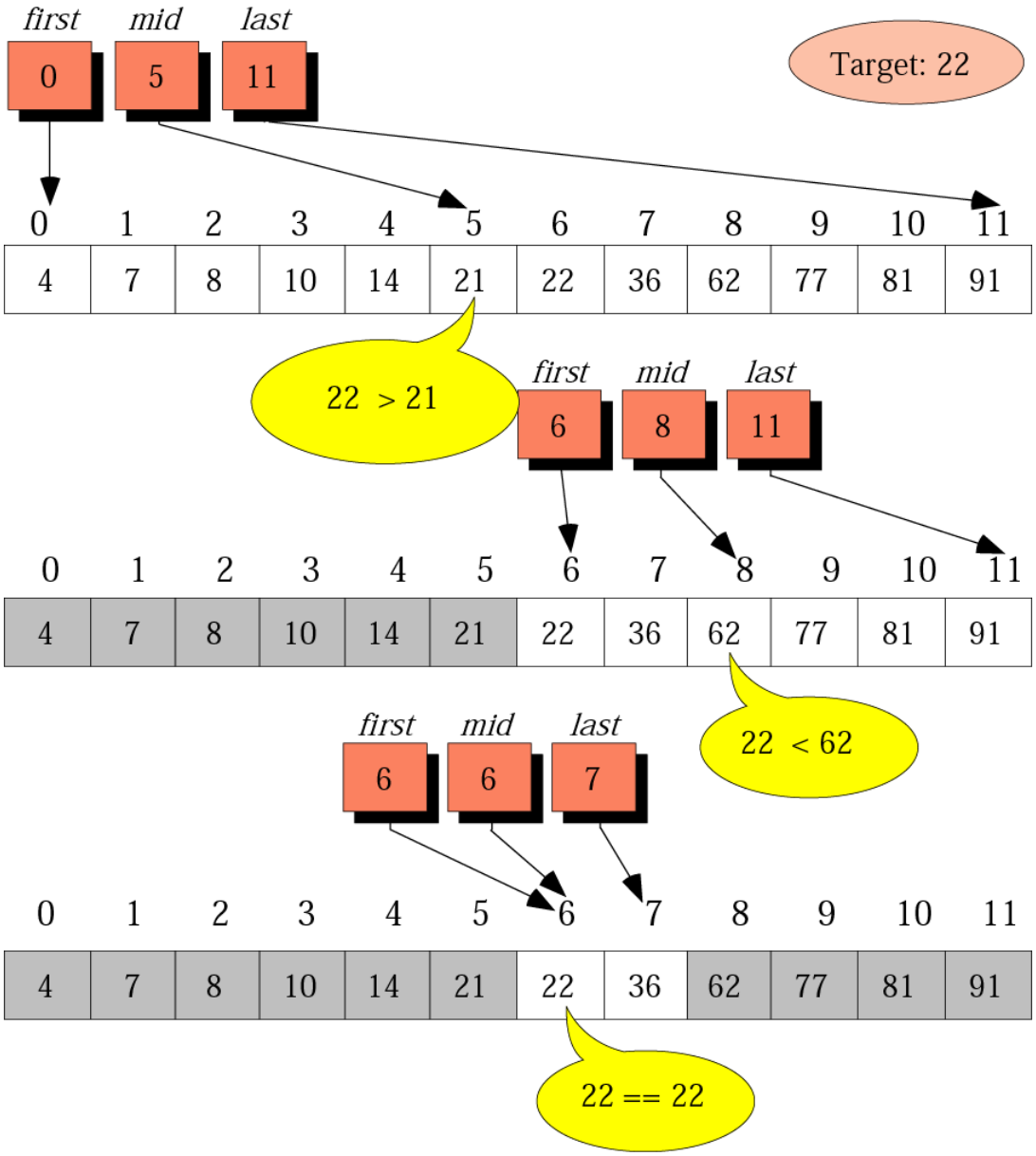


ตัวอย่าง sequential search



รูปที่ 8-21

ตัวอย่าง binary search



8.1

การเรียกตัวเอง
RECURSION

Iterative definition of factorial

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

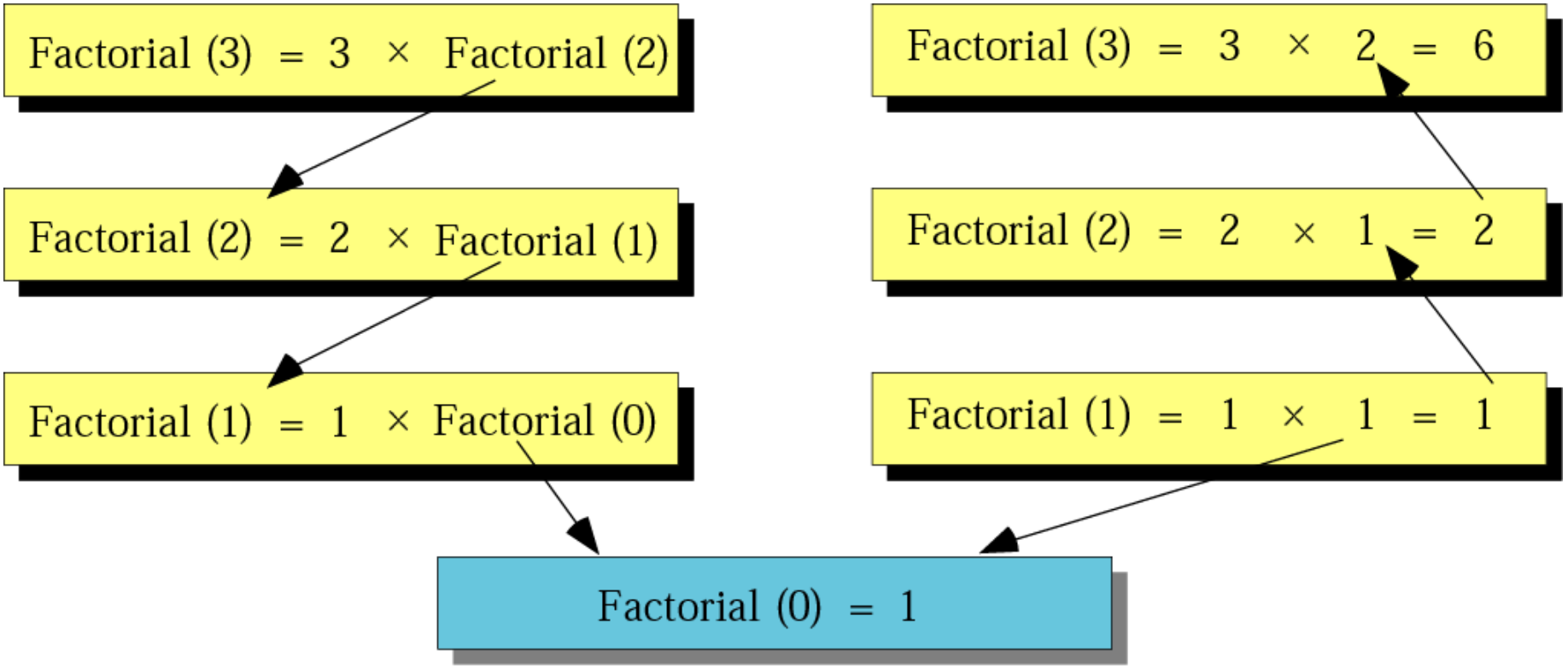
รูปที่ 8-23

Recursive definition of factorial

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

รูปที่ 8-24

Tracing recursive solution to factorial problem



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
```

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
```

คำสำคัญ

- Algorithm
- Binary Search
- Bubble Sort
- Flowchart
- Insertion Sort
- Pseudocode
- Selection Sort
- Sequential Search
- Module
- Subalgorithm
- Subprogram
- Subroutine