10301222 โครงสร้างข้อมูลและอัลกอริทึม

บทที่ 1: อัลกอริทึม

ผศ.ดร. ปวีณ เขื่อนแก้ว

สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยแม่โจ้

Data structure : โครงสร้างข้อมูล

ต้นแบบอัลกอริทึมการวิเคราะห์ โครงสร้างข้อมูลแบบเชิงเส้น โครงสร้างข้อมูลแบบไม่มีเชิงเส้น แถวลำดับ คิว ลิงค์ลิสต์ โครงสร้างต้นไม้ กราฟ การค้นข้อมูล การเรียงลำดับข้อมูล ฟังก์ชันของเวลาเชิงซับซ้อน การแบ่งแยกและเอาชนะ หลักการกำหนดการพลวัต อัลกอริทึมแบบละโมภ เอ็นพีบริบูรณ์

Data structure : โครงสร้างข้อมูล

มีอะไรในวิชานี้บ้าง ?

-การเก็บข้อมูล และจัดการข้อมูล

เก็บ / เรียกใช้ / ค้น / ลบ / เรียง / จัดรูปแบบ

แล้วเรียนไปเพื่ออะไร ?

-ทุกโปรแกรมต้องมี การจัดการข้อมูล

การใช้ตัวแปร ก็เป็นการจัดการข้อมูลรูปแบบหนึ่ง

Data structure : โครงสร้างข้อมูล

แล้วเรียนอะไร ?

- เรียนอัลกอริทึม การเก็บข้อมูล และจัดการข้อมูล

จำเป็นต้องเรียนเป็น 1 วิชาเลยหรือ ?

- เพราะไม่มีอัลกอริทึมไหนเท่าเทียมกัน
- เพราะอัลกอริทึมเทพ ไม่มีในโลก
- ศึกษาข้อดี และ ข้อเสีย ของแต่ละอัลกอริทึม

Data structure : โครงสร้างข้อมูล

ตัวอย่างเช่น

- โปรแกรมดิกชันนารี่

  ต้องค้นข้อมูลได้เร็วที่สุด แต่ไม่เน้นความเร็วในการใส่ข้อมูล

- โปรแกรมเก็บ log ตามพ.ร.บ.คอมพิวเตอร์

  เน้นความเร็วในการใส่ข้อมูล แต่ไม่เน้นความเร็วในการค้น

- โปรแกรมสมุดโทรศัพท์

  เน้นความเร็วในการเรียงข้อมูลตามลำดับตัวอักษร
  แต่ไม่เน้นความเร็วในการบันทึก

- โปรแกรมแผนที่

  เน้นความเร็วในการค้น และข้อมูลต้องมีความสัมพันธ์กัน

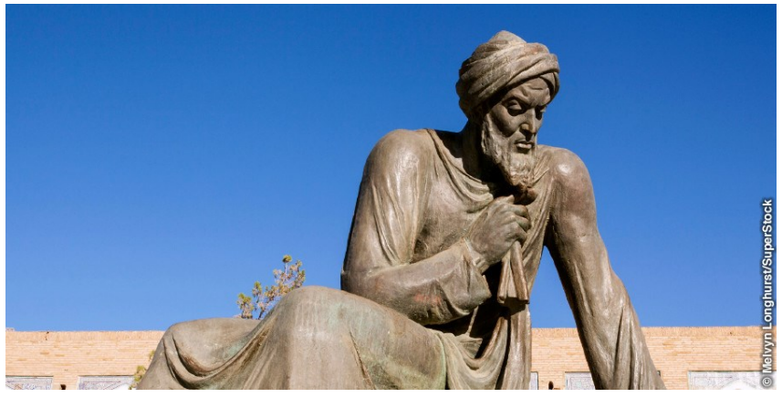
Data structure : โครงสร้างข้อมูล

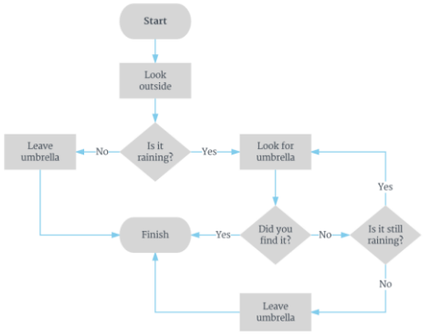
เรียนวิชานี้ต้องรู้อะไรมาก่อนบ้าง

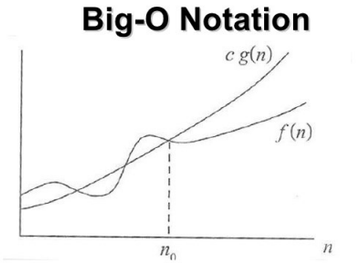
- การเขียนโปรแกรม (Java , C)
- อ่านอัลกอริทึมเป็น (pseudo code)
- เข้าใจความซับซ้อนของโปรแกรม (Big O)

Data structure : โครงสร้างข้อมูล

สัปดาห์ที่	หัวข้อ	ชั่วโมง
1	บทนำ	5
2	อัลกอริทึมเบื้องต้น	5
3	การวิเคราะห์ประสิทธิภาพของอัลกอริทึม	5





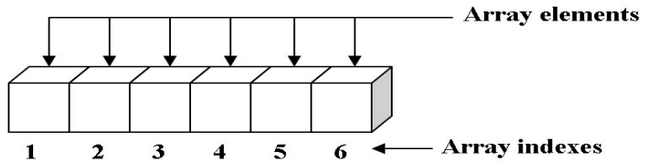


f(n) ϵ O(g(n))

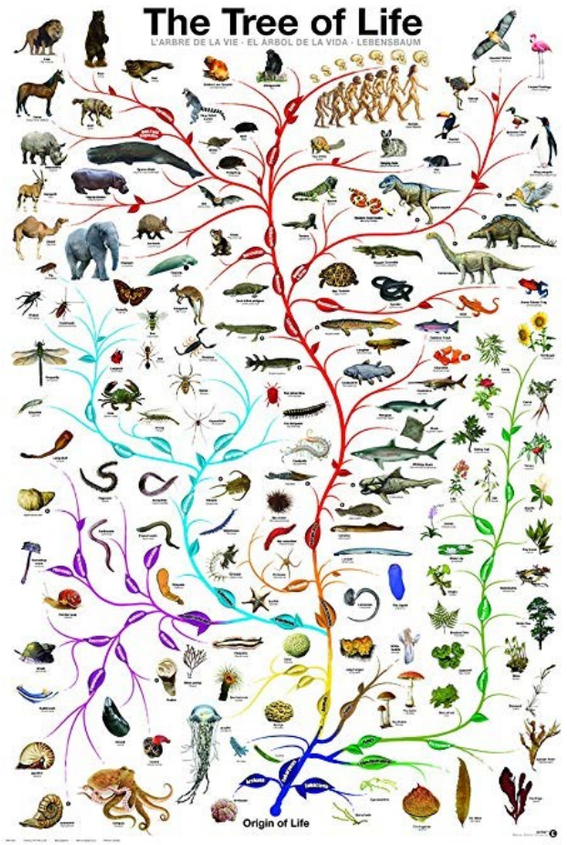
f(n) ≤(g(n))

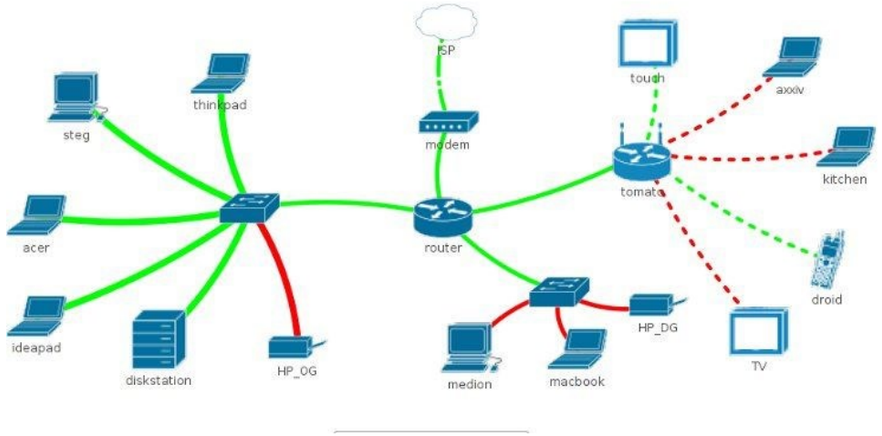
Data structure : โครงสร้างข้อมูล

4-5	โครงสร้างข้อมูลแบบเชิงเส้น โครงสร้างข้อมูลแบบไม่เชิงเส้น	10



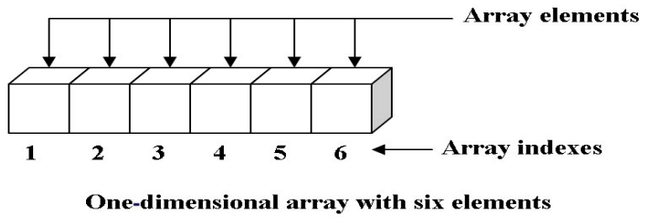
One-dimensional array with six elements



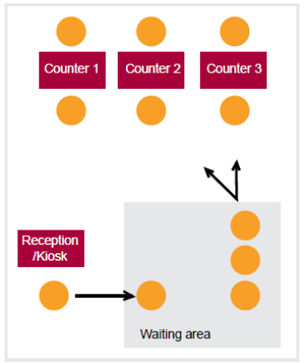


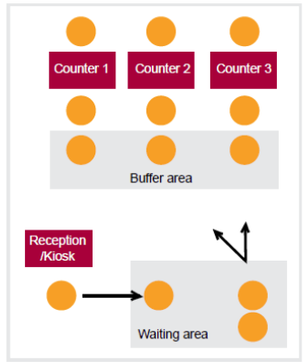
Data structure : โครงสร้างข้อมูล

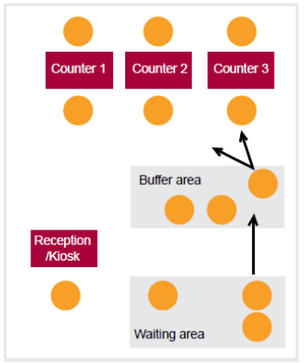
6	แถวลำดับ	5



One-dimensional array with six elements

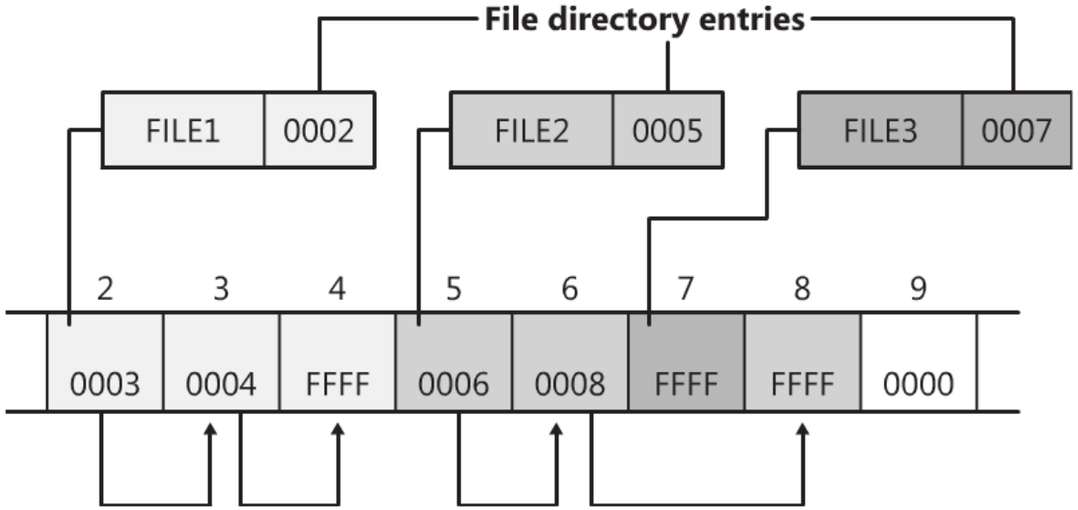


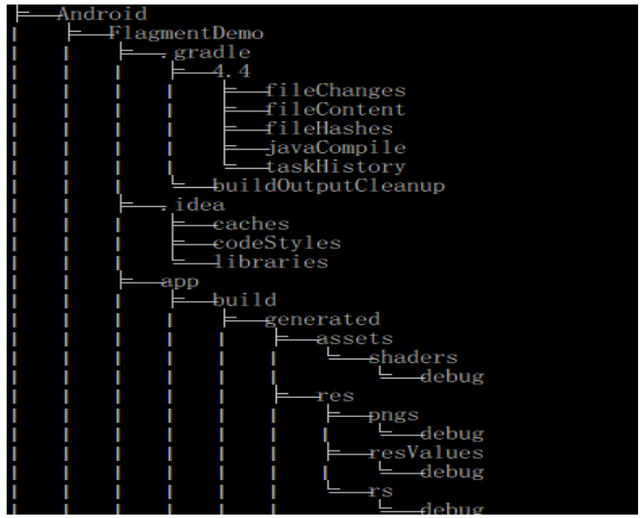


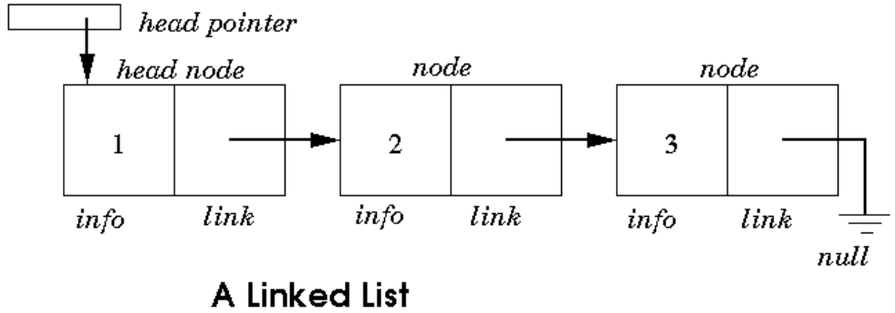


Data structure : โครงสร้างข้อมูล

7-9	ลิงค์ลิสต์	15



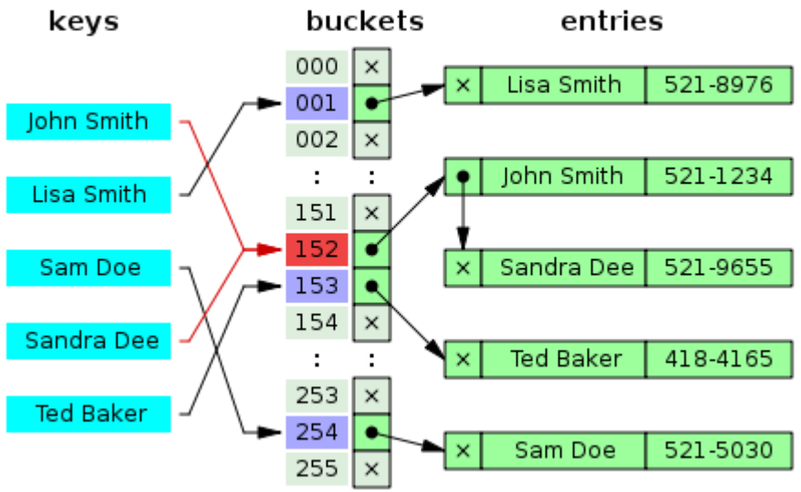




A Linked List

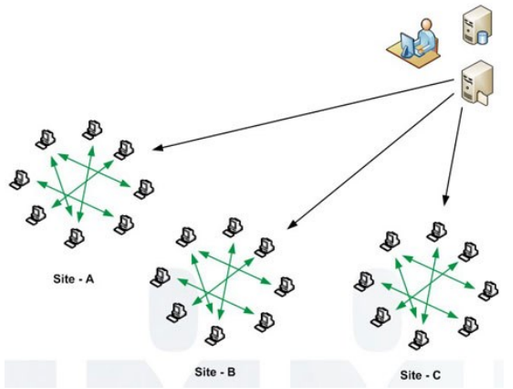
Data structure : โครงสร้างข้อมูล

10	ดิกชันนารี และ แฮชชิ่ง	5



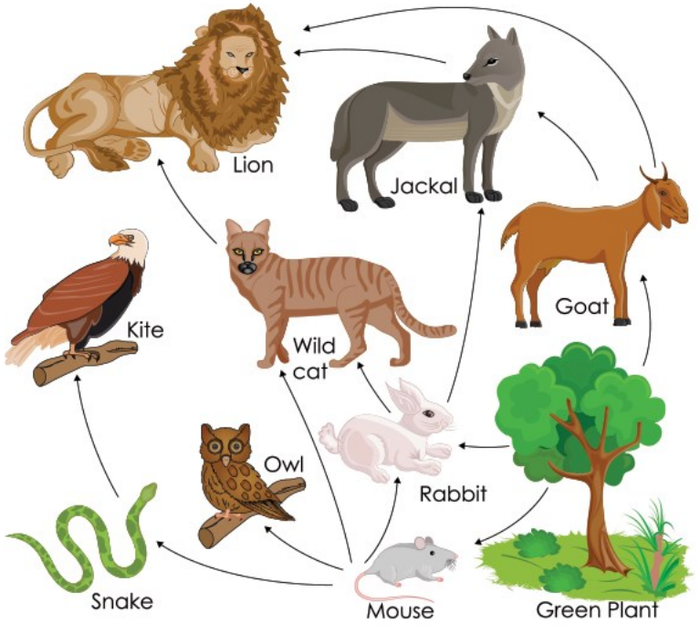
=

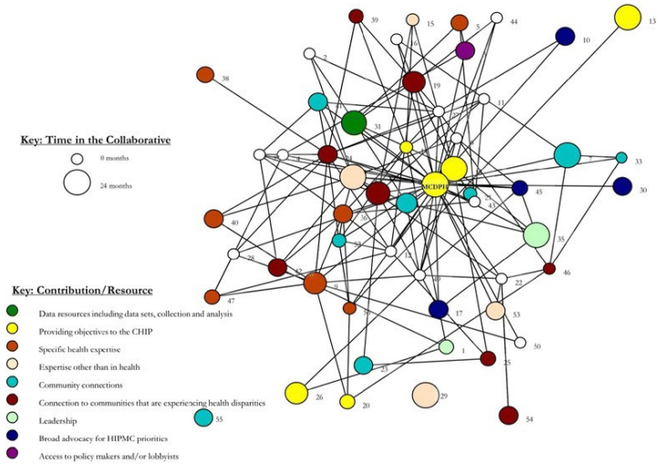
79054025
255fb1a2
6e4bc422
aef54eb4

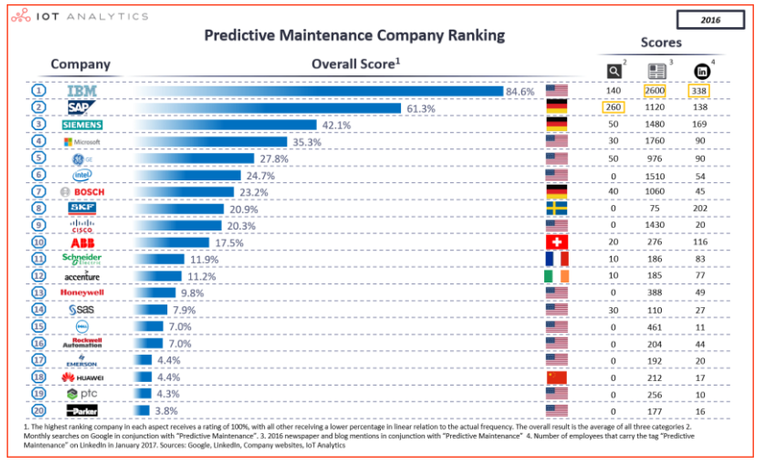


Data structure : โครงสร้างข้อมูล

11	โครงสร้างต้นไม้และ กราฟ	5
12	การค้นข้อมูล การเรียงลำดับข้อมูล	5

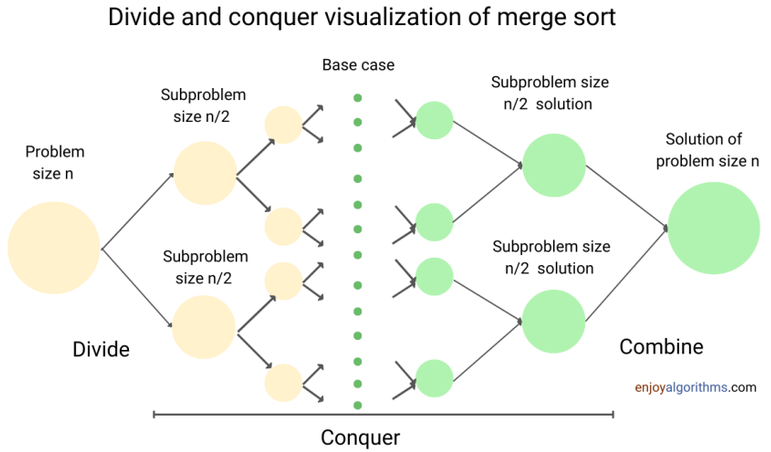


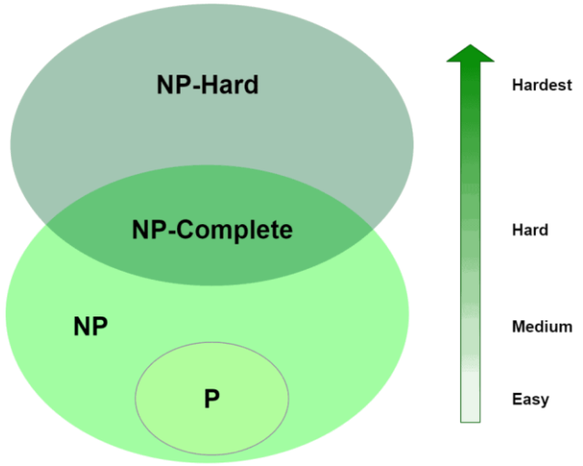


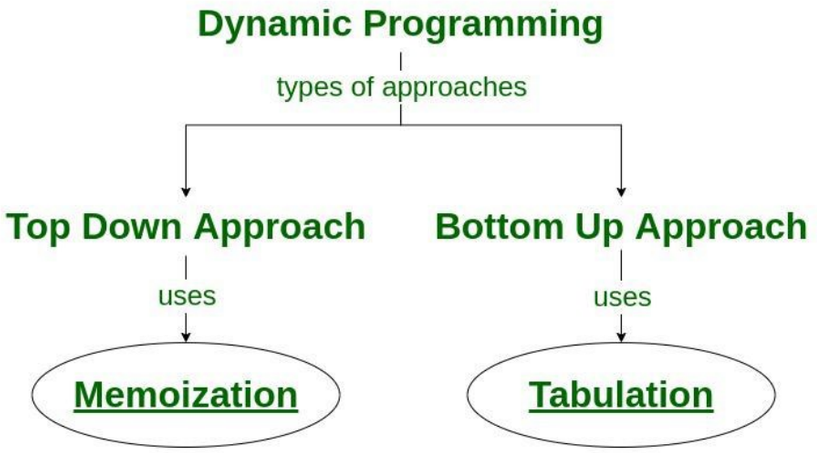


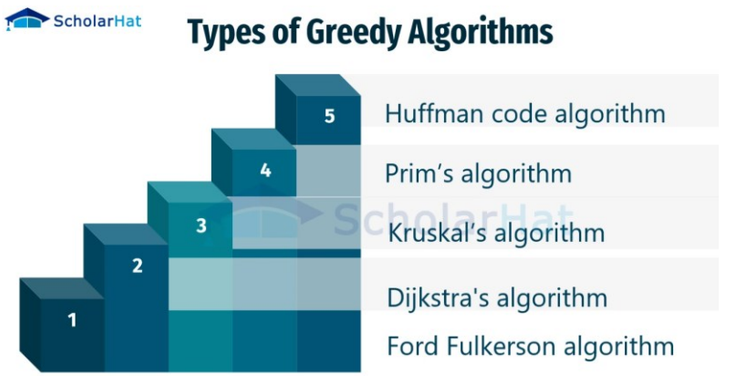
Data structure : โครงสร้างข้อมูล

13	ฟังก์ชันของเวลาเชิงซับซ้อน การแบ่งแยกและเอาชนะ	5
14	หลักการกำหนดการพลวัต อัลกอริทึมแบบละโมภ	5
15	เอ็นพีบริบูรณ์	5







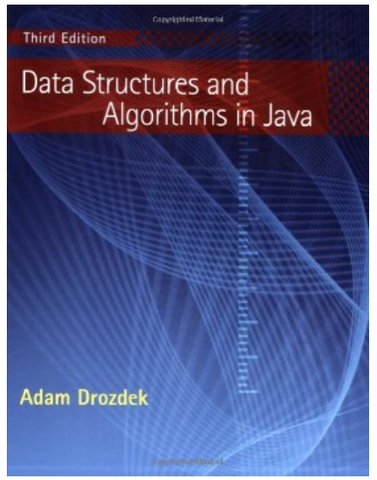


Data structure : โครงสร้างข้อมูล

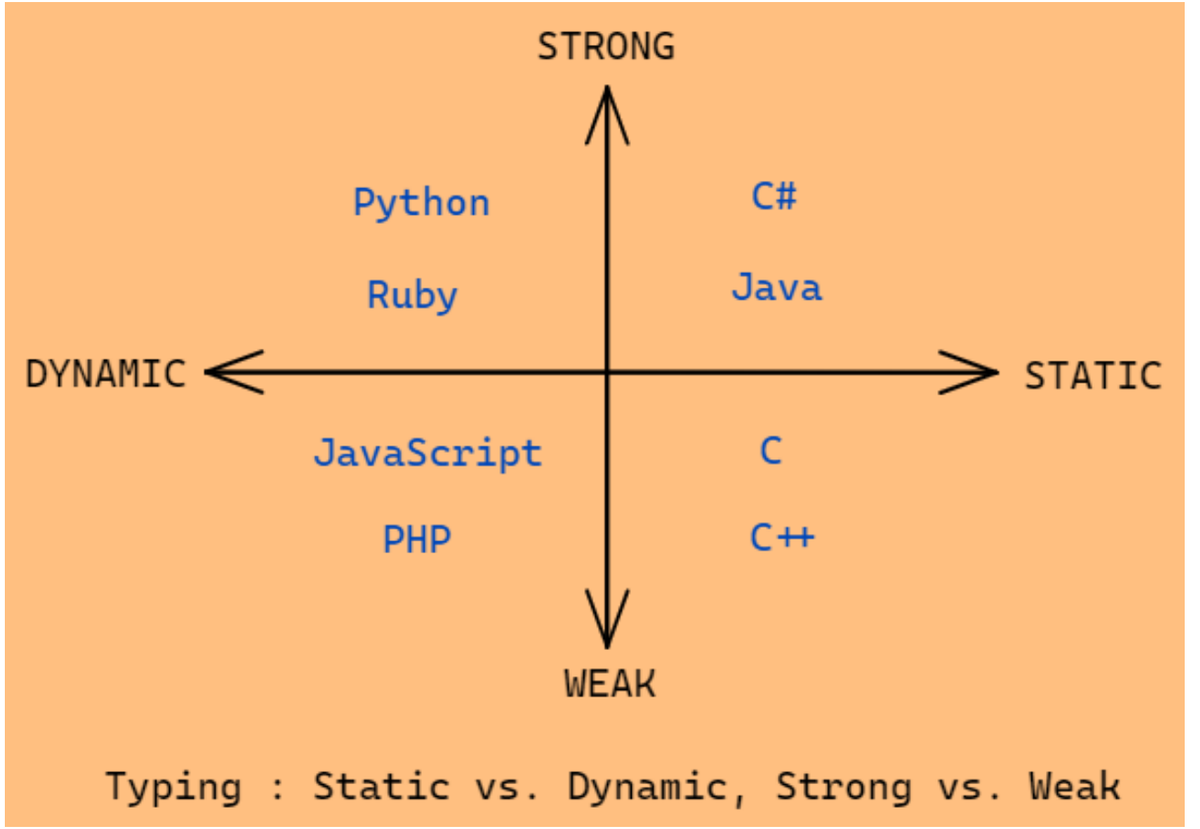
การให้คะแนน : สอบ 60%, แลป 20%, การบ้าน 10%, การเข้าเรียน 10%

เกณฑ์การประเมินผล (สามารถปรับเปลี่ยนได้ตามความเหมาะสม)				
	80 % ขึ้นไป	ระดับคะแนน A	60 – 64 %	ระดับคะแนน C
	75 – 79 %	ระดับคะแนน B+	55 – 59 %	ระดับคะแนน D+
	70 – 74 %	ระดับคะแนน B	50 – 54 %	ระดับคะแนน D
	65 – 69 %	ระดับคะแนน C+	ต่ำกว่า 50 %	ระดับคะแนน F

www.drpaween.com



Data structure : โครงสร้างข้อมูล



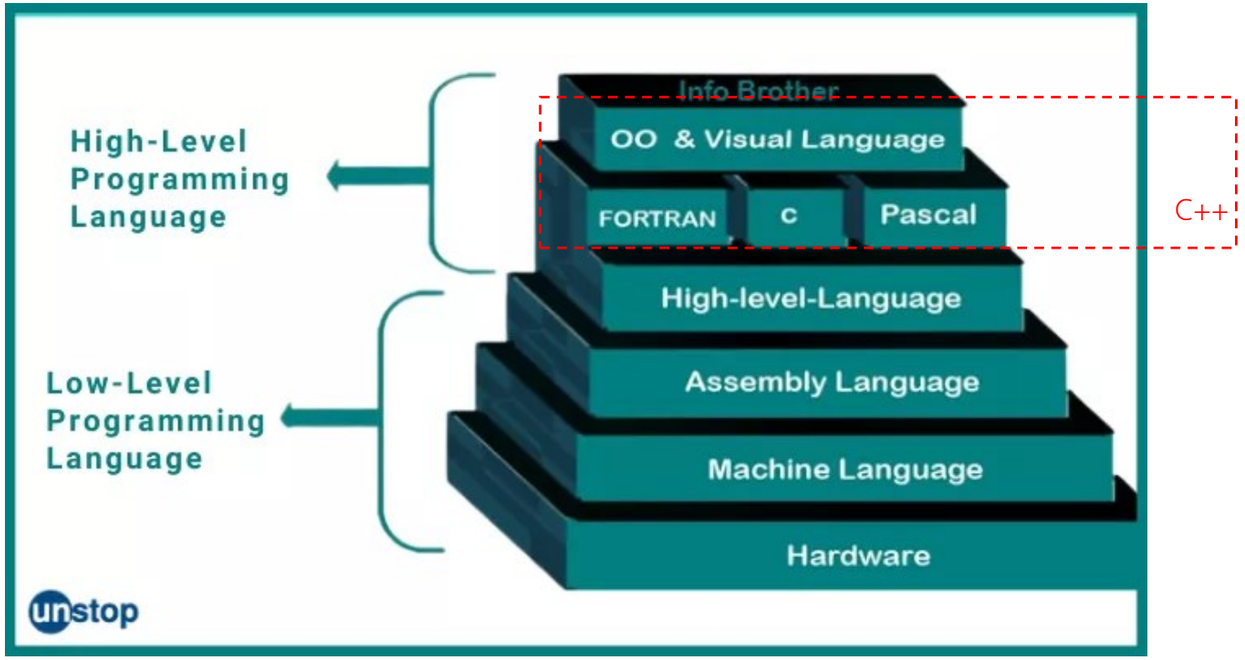
Strong type: Data type related rules and restrictions are Strictly maintained

Weak type: Data type related rules and restrictions are Loosely maintained

Static: Type checking is done at compile time

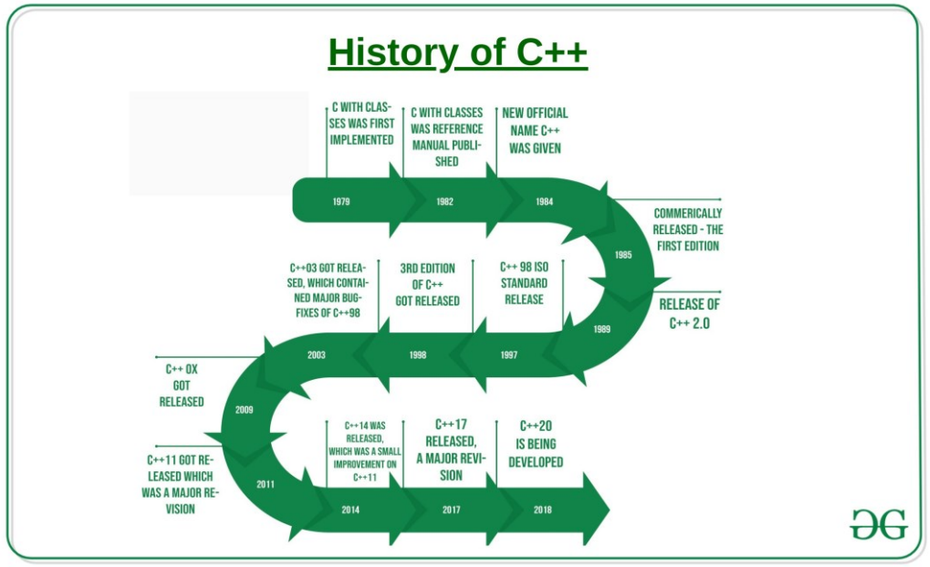
Dynamic Type: Type checking is done at runtime

Data structure : โครงสร้างข้อมูล



C++ สามารถเข้าถึงข้อมูล ในหน่วยความจำ และในรีจิสเตอร์ภายในไมโครโพรเซสเซอร์ได้
รองรับการโปรแกรมในรูปแบบเชิงวัตถุ ทำให้สามารถพัฒนาโปรแกรมที่มีขั้นตอนวิธีการประมวลผลซับซ้อนได้ง่าย

Data structure : โครงสร้างข้อมูล

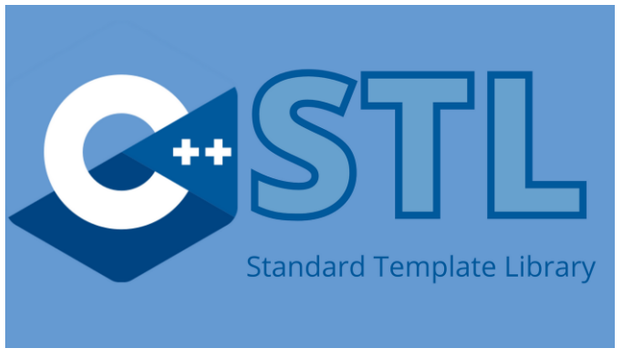


Data structure : โครงสร้างข้อมูล

Code::Blocks

MinGW-w64

A complete runtime environment for GCC & LLVM
for 32 and 64 bit Windows



Data structure : โครงสร้างข้อมูล

Visual Studio Code

MinGW-w64

A complete runtime environment for GCC & LLVM
for 32 and 64 bit Windows

MSYS2

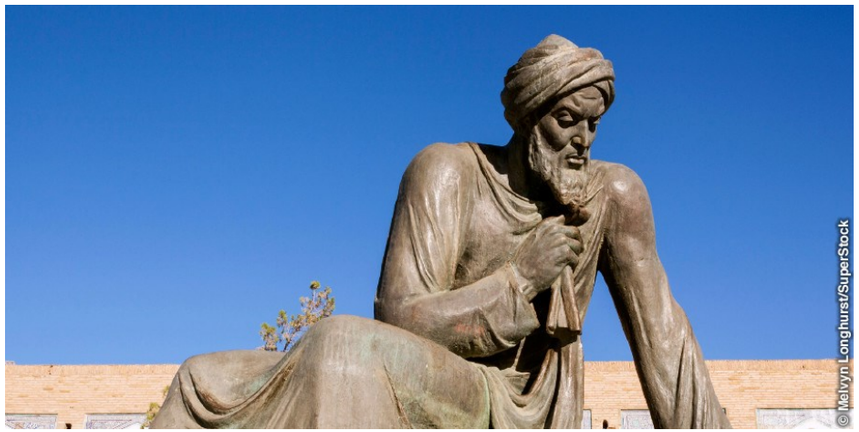
Data structure : โครงสร้างข้อมูล

อัลกอริทึม

ขั้นตอนวิธี หรือ **อัลกอริทึม** (อังกฤษ: algorithm) หมายถึง กระบวนการแก้ปัญหาที่สามารถเข้าใจได้ มีลำดับหรือวิธีการในการแก้ไขปัญหาใดปัญหาหนึ่งอย่างเป็นขั้นเป็นตอนและชัดเจน เมื่อนำเข้าอะไร แล้วจะต้องได้ผลลัพธ์เช่นไร

ซึ่งแตกต่างจากการแก้ปัญหาแบบสามัญสำนึก หรือฮิวริสติก (heuristic)

Algorithms

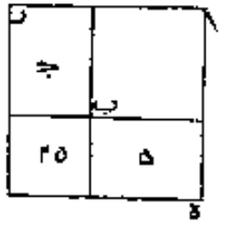


Muhammad ibn Musa al-Khwarizmi

محمد بن موسى الخوارزمى

(c. 780 – c. 850)

علي تسعة وثلثين ليتم السطح الاعظم الذي هو سطح ره فبلغ
ذلك كله اربعة وستين فاخذنا جذرها وهو ثمانية وهو احد
اضلاع السطح الاعظم فاذا نقصنا منه مثل ما زدنا عليه وهو
خمسة بقي ثلثة وهو ضلع سطح اب الذي هو المال وهو جذره
والمال تسعة وهذه صورته



واما مال واحد وعشرون درهما يعدل عشرة اجذاره فانا
نجعل المال سطحا مربعا مجهول الاضلاع وهو سطح اد ثم نضم
اليه سطحا متوازي الاضلاع عرضه مثل احد اضلاع سطح اد وهو
ضلع هن والسطح ده فصار طول السطحين جميعا ضلع جه
وقد علمنا ان طوله عشرة من العدد لان كل سطح مربع
متساوي الاضلاع والزوايا فان احد اضلاعه مضروبا في واحد جذر
ذلك السطح وفي اثنين جذراه فلما قال مال واحد وعشرون
يعدل عشرة اجذاره علمنا ان طول ضلع هج عشرة اعداد لان
ضلع جد جذر المال فقسمنا ضلع جه بنصفين علي نقطة

Algorithms

The compendious book on calculation by completion and balancing

อธิบายขั้นตอนของการหาคำตอบโจทย์การคำนวณต่าง ๆ ในชีวิตประจำวัน

- การแก้สมการ หาตัวแปรที่ไม่ทราบค่า
- ระบบสมการเชิงเส้น และ สมการกำลังสอง

A page from al-Khwārizmī's *Algebra*

ตัวอย่างการแก้สมการเชิงเส้นด้วย Cramer's rule

$$x + 4y - 2z = -3$$
$$-2x + 3y - 3z = 5$$
$$2y - 4z = 8$$

$$A = \begin{bmatrix} 1 & 4 & -2 & -3 \\ -2 & 3 & -3 & 5 \\ 0 & 2 & -4 & 8 \end{bmatrix}$$

$$|A_0| = \begin{vmatrix} 1 & 4 & -2 \\ -2 & 3 & -3 \\ 0 & 2 & -4 \end{vmatrix}$$

$|A_0| = 1\cdot3\cdot(-4) + 4\cdot(-3)\cdot0 + (-2)\cdot(-2)\cdot2 - (-2)\cdot3\cdot0 - 4\cdot(-2)\cdot(-4) - 1\cdot(-3)\cdot2 = -12+0+8+0-32+6 = -30$

$$|A_x| = \begin{vmatrix} -3 & 4 & -2 \\ 5 & 3 & -3 \\ 8 & 2 & -4 \end{vmatrix}$$

$|A_x| = (-3)\cdot3\cdot(-4) + 4\cdot(-3)\cdot8 + (-2)\cdot5\cdot2 - (-2)\cdot3\cdot8 - 4\cdot5\cdot(-4) - (-3)\cdot(-3)\cdot2 = 36-96-20+48+80-18 = 30$

$$|A_y| = \begin{vmatrix} 1 & -3 & -2 \\ -2 & 5 & -3 \\ 0 & 8 & -4 \end{vmatrix}$$

$|A_y| = 1\cdot5\cdot(-4) + (-3)\cdot(-3)\cdot0 + (-2)\cdot(-2)\cdot8 - (-2)\cdot5\cdot0 - (-3)\cdot(-2)\cdot(-4) - 1\cdot(-3)\cdot8 = -20+0+32+0+24+24 = 60$

$$|A_z| = \begin{vmatrix} 1 & 4 & -3 \\ -2 & 3 & 5 \\ 0 & 2 & 8 \end{vmatrix}$$

$|A_z| = 1\cdot3\cdot8 + 4\cdot5\cdot0 + (-3)\cdot(-2)\cdot2 - (-3)\cdot3\cdot0 - 4\cdot(-2)\cdot8 - 1\cdot5\cdot2 = 24+0+12+0+64-10 = 90$

$$x = \frac{|A_x|}{|A_0|}, \quad y = \frac{|A_y|}{|A_0|}, \quad z = \frac{|A_z|}{|A_0|}$$

$$x = \frac{30}{-30} = -1, \quad y = \frac{60}{-30}, \quad z = \frac{90}{-30} = -3$$

Algorithms

คุณสมบัติของอัลกอริทึมที่ดี

1.เป็นกระบวนวิธีการที่สร้างขึ้นจากกฎเกณฑ์

2.การเขียนอัลกอริทึมต้องไม่คลุมเครือ

3.ต้องมีลำดับขั้นตอนที่ชัดเจน

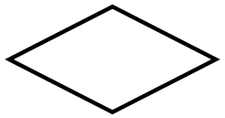
4.กระบวนวิธีการต้องให้ผลลัพธ์ตามที่กำหนดในปัญหา (ผลลัพธ์ถูกต้อง)

5.อัลกอริทึมต้องมีจุดสุดท้ายของการทำงาน

Algorithms

การอธิบายอัลกอริทึม

ผังงาน (Flow chart)

สัญลักษณ์	ความหมาย
	**จุดเริ่มต้นหรือจุดจบของโปรแกรม (Terminal)**
	**การรับและแสดงผลของข้อมูล (Input/Output)**
	**การประมวลผล (Process)**
	**การตัดสินใจ** (Decision /Selection)

รหัสเทียม (Pseudo code)

Algorithm: Grading

Input:SCORE
Output:GRADE

```
1. GRADE='F'
2. IF SCORE >49 THEN
                GRADE='D'
3. IF SCORE >59 THEN
                GRADE='C'
4. IF SCORE >69 THEN
                GRADE='B'
5. IF SCORE >79 THEN
                GRADE='A'
6. RETURN GRADE
```

Algorithms:รหัสเทียม

รหัสเทียม (อังกฤษ: pseudocode) ใช้เป็นภาษากลางในการอธิบายขั้นตอนการทำงานของโปรแกรมและขั้นตอนวิธี

รหัสเทียมไม่มีหลักเกณฑ์ตายตัว สำคัญเพียงแต่เขียนให้ผู้อ่านเข้าใจ

รหัสเทียมนั้นมักจะไม่ใส่ใจในรายละเอียดการเขียนมากนัก เช่น อาจไม่มีขั้นตอนการประกาศตัวแปร

เป้าหมายสำคัญของการเขียนรหัสเทียมคือทำลายกำแพงของภาษาลงไป

การเขียนรหัสเทียมจึงไม่ใส่ใจในการเขียนไวยากรณ์ให้ถูกต้องตามหลักภาษา แต่จะเป็นไปตามใจของผู้เขียนมากกว่า

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)

Algorithm: ไร้สาระ
Input: ไม่มี Output: ไม่มี
1. ตื่น 2. กิน 3. เล่น 4. นอน

Algorithm: ไร้สาระ
Input: ไม่มี Output: ไม่มี
Step 1) ตื่น Step 2) กิน Step 3) เล่น Step 4) นอน

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)

Algorithm: ตรวจสอบเลขคู่
Input: k Output: ans
1. IF (k mod 2) == 0 THEN ans=‘Yes’ ELSE ans=‘No’ 2. RETURN ans

Algorithm: ตรวจสอบเลขคู่
Input: k Output: ans

1. IF (k mod 2) == 0 THEN
ans=‘Yes’
ELSE
ans=‘No’
END IF
2. RETURN ans

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)

Algorithm: จีบ
Input: ไม่มี Output: ไม่มี
1. WHILE (มีแฟน == FALSE) หาแฟน

Algorithm: จีบ
Input: ไม่มี Output: ไม่มี
1. WHILE (มีแฟน == FALSE) หาแฟน END WHILE

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)
4. กระทำซ้ำเมื่อเป็นจริง (Repeat-Until)

Algorithm: ง้อ
Input: ไม่มี Output: ไม่มี
1. REPEAT ง้อ UNTIL (หายงอน == TRUE)

Algorithm: Login
Input: password Output: ไม่มี
1. REPEAT READ ข้อความ UNTIL (ข้อความ == password)

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)
4. กระทำซ้ำเมื่อเป็นจริง (Repeat-Until)
5. วนซ้ำแบบมีการนับ (For)

Algorithm: หาผลรวมสะสม
Input: a,b Output: ans
1. FOR i=a to b ans=ans+i 2. Return ans

Algorithm: หาผลรวมสะสม
Input: A Output: ans
1. For each i in A ans=ans+1 2. Return ans

- ตัวแปรที่เป็น set ตัวอย่างเช่น array หรือ matrix มักนิยมเขียนด้วยตัวพิมพ์ใหญ่ หรือตัวหนา
- ตัวแปรที่เป็น scalar หรือ atomic (มีค่าได้เพียงค่าเดียวในเวลานั้น) นิยมเขียนด้วยตัวพิมพ์เล็ก หรือตัวเอียง

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)
4. กระทำซ้ำเมื่อเป็นจริง (Repeat-Until)
5. วนซ้ำแบบมีการนับ (For)

Algorithm: หาผลรวมสะสม
Input: A Output: ans
1. For each i in A ans=ans+1 2. Return ans

ans=1+2+3+4+5+6

K=[1,2,3,4,5,6]

x= หาผลรวมสะสม(K)

x=21

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)
4. กระทำซ้ำเมื่อเป็นจริง (Repeat-Until)
5. วนซ้ำแบบมีการนับ (For)
6. เงื่อนไข (Case)

Algorithm: Driving
Input: traffic light Output: none
1. CASE traffic light OF red: Stop orange: Slow down Green: Speed up default: Stop

Algorithms:รหัสเทียม

การสร้างรหัสเทียมจะใช้คำสั่งเพียง 6 ชนิดคือ

1. กำหนดลำดับ (Sequence)
2. เงื่อนไข (IF-THEN-ELSE)
3. กระทำเมื่อเป็นจริง (While)
4. กระทำซ้ำเมื่อเป็นจริง (Repeat-Until)
5. วนซ้ำแบบมีการนับ (For)
6. เงื่อนไข (Case)

มีเพียง 6 คำสั่งนี้ ก็มากพอที่จะใช้อธิบาย ทุก ๆโปรแกรมและอัลกอริทึมที่มีในโลกได้

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0  2. For each human in image n=n+1 mark X over that human 3. Return n

n=0

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image ⬅ n=n+1 mark X over that human 3. Return n

n=0

เลือกหยิบมา 1 คน

ขั้นตอนวิธีนี้ไม่ได้ระบุลำดับ ดังนั้นจึงเลือกคนใหนมาก็ได้ แต่ต้องไม่ซ้ำกับที่เคยเลือกมาแล้ว

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 ⬅ mark X over that human 3. Return n

n=1

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=1

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=1

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

n=2

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

n=3

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=3

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=3

วนซ้ำเรื่อยๆ จนครบทุกคน

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=8

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการนับจำนวนคนในรูปภาพ

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

n=8

ขั้นตอนวิธีนี้ตอบ 8

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

เริ่มจับ

หยุดจับ

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

ในการวิเคราะห์ความเร็วในการทำงานของขั้นตอนวิธี เราไม่นิยมจับเวลาจริงๆ เพราะค่าที่วัดได้ ไม่มีประโยชน์

คอมพิวเตอร์แต่ละเครื่องไม่เหมือนกัน จึงใช้เวลาไม่เท่ากัน

โปรแกรมเดียวกันแต่เขียนคนละภาษา ก็ใช้เวลาไม่เท่ากัน

ดังนั้นเมื่อพูดถึงเวลาในการทำงานของขั้นตอนวิธี เราหมายถึง เวลาในทางทฤษฎี

ขั้นตอนวิธีที่มีความซับซ้อนสูง (High Complexity) จะใช้เวลาในการทำงานสูง

ขั้นตอนวิธีที่มีความซับซ้อนน้อย (Low Complexity) จะใช้เวลาในการทำงานต่ำ

ซับซ้อน = เวลา

ในทางทฤษฎี

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ขั้นตอนวิธี Counting นี้ มีขั้นตอนไหน ใช้เวลามากที่สุด ?

1 ครั้ง

ไม่รู้ว่าต้องวนกี่รอบ

1 ครั้ง

ขั้นตอนที่ 2 เป็นตัวกำหนดเวลาที่ใช้ในการทำงาน

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ไม่รู้ว่าต้องวนกี่รอบ

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

ถ้าเปลี่ยนภาพ เวลาในการทำงานเปลี่ยนหรือไม่ ?

เวลาที่ใช้ในการทำงานเป็นฟังก์ชัน (function)

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

เวลาที่ใช้ในการทำงานขึ้นอยู่กับ จำนวนคน

จำนวนคน = n

ดังนั้นขั้นตอนวิธีนี้ ใช้เวลาการทำงาน n

หรือมีความซับซ้อนทางเวลา = n

เนื่องเวลาการทำงานเป็นฟังก์ชั้น เรานิยมเรียกฟังก์ชันนี้ว่า

ฟังก์ชั่น O

อ่านว่า Big - O

ขั้นตอนวิธีนี้ใช้มีเวลาการทำงานเป็น O(n)

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ขั้นตอนวิธีนี้ใช้เวลาในการทำงานเท่าไหร่ ?

ตอบ ขั้นตอนวิธีนี้ใช้เวลาการทำงานเป็น O(n)

แล้วนำไปใช้อย่างไร ?

ตัวอย่างเช่น: ต้องการนับภาพที่มี 100 คน

ด้วยเครื่องที่ประมวลผลคำสั่งละ 1 วินาที จะใช้เวลาเท่าใด

วิธีทำ ขั้นตอนวิธีนี้ใช้เวลาการประมวลผล O(n)

ภาพมี 100 คน n=100

ต้องวนซ้ำ O(100) = 100 ครั้ง

ต้องประมวลผลคำสั่ง 100 ครั้ง จึงใช้เวลา 100 x 1

= 100 วินาที

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0  2. For each human in image n=n+1 mark X over that human 3. Return n 

แล้วทำไมไม่เอาขั้นตอนที่ 1 และ 3 มาคิด ?

จะคิดก็ได้ ไม่ผิด จะได้เป็น O(n) +1 +1 = O(n)+2

เนื่องจาก เวลาในการทำงานของ (1) และ (3) มีค่าเท่าเดิม ไม่ได้เปลี่ยนแปลงตาม input

การประมวลผลที่เวลาไม่เปลี่ยนตามข้อมูล เราเรียกว่า เวลาคงที่ (Constant) หรือ c

จะได้เป็น O(n) + c

แล้วทำไมไม่เอา c มาคิด

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0  2. For each human in image n=n+1 mark X over that human 3. Return n 

แล้วทำไมไม่เอา c มาคิด ?

ลองพิจารณาตัวอย่างนี้ดู ขั้นตอนวิธีนี้ c=2

จำนวนคน	n	c	เวลารวม (t)
1	1	2	3
100	100	2	102
1000	1000	2	1002
10000	10000	2	10002
100000	100000	2	100002
1000000	1000000	2	1000002
∞	∞	2	∞ + 2

จะว่า t ≈ n ในกรณีที่ n มีค่ามาก ๆ

เราจึงไม่นิยมเอา c มาคิดในการวิเคราะห์ Big-O

คิดเล่นๆอยู่บ้านไม่ว่ากัน แต่ถ้าตอนสอบเอาค่า c มาตอบ จะได้ 0 คะแนน

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

แสดงว่า Big-O พิจารณาเฉพาะ loop วนซ้ำใช่หรือไม่ ?

ตอบ: ใช่ และพิจารณาเฉพาะ loop ที่จำนวนการวนเปลี่ยนตาม input ด้วย

ตัวอย่างเช่น การวน loop เพื่อล้างค่าตัวแปร ที่มีจำนวนครั้งการวนเท่าเดิมทุกครั้ง ก็ไม่นำมาพิจารณา

เท่าเดิม = Constant = c

Algorithms:การออกแบบขั้นตอนวิธี

ตั้งชื่อขั้นตอนวิธีว่า Counting

กำหนดให้ image คือภาพ

n คือจำนวนคน

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

ถ้าปรับปรุงขั้นตอนวิธี ให้นับทีละ 2 คน (ทีละคู่)

เวลาทำงานจะเหลือครึ่งเดียวใช่ไหม ?

Algorithm: Counting2
Input: image Output: n
1. n=0 2. For each pair of human in image n=n+2 mark X over that pair 3. If human remain in image Then n=n+1 4. Return n

Algorithms:การออกแบบขั้นตอนวิธี

Algorithm: Counting2
Input: image Output: n
1. n=0 2. For each pair of human in image n=n+2 mark X over that pair 3. If human remain in image Then n=n+1 4. Return n

ถ้าปรับปรุงขั้นตอนวิธี ให้นับทีละ 2 คน (ทีละคู่)
เวลาทำงานจะเหลือครึ่งเดียวใช่ไหม ?

$=O(n/2)$

$=O(n) \times O(1/2)$

ค่าคงที่

$=O(n) \times O(0.5)$

$=O(n) \times c$ ← ค่าคงที่ ไม่เอามาคิด

$=O(n)$

จะนับทีละคู่ก็เร็วเท่าเดิม

งงเด๊!

Algorithms:การออกแบบขั้นตอนวิธี

Counting1

จำนวนคน	n	c	เวลารวม (t)
1	1	2	3
100	100	2	102
1000	1000	2	1002
10000	10000	2	10002
100000	100000	2	100002
1000000	1000000	2	1000002
∞	∞	2	∞ + 2

Counting2

จำนวนคน	n	c	เวลารวม (t)
1	1	3	4
100	100	3	54
1000	1000	3	504
10000	10000	3	5004
100000	100000	3	50004
1000000	1000000	3	500004
∞	∞	3	(∞ / 2) + 4

ถ้า n มาค่ามาก ๆ ขั้นตอนวิธีทั้งสอง ก็ใช้เวลาเท่ากัน

Algorithms:การออกแบบขั้นตอนวิธี

Algorithm: Counting
Input: image Output: n
1. n=0 2. For each human in image n=n+1 mark X over that human 3. Return n

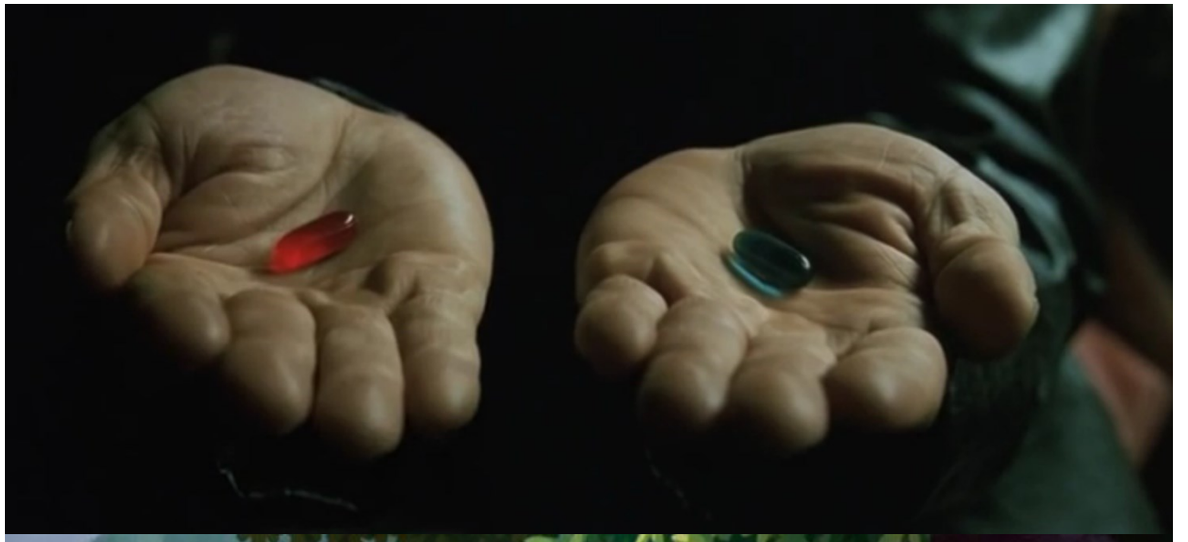
Algorithm: Counting2
Input: image Output: n
1. n=0 2. For each pair of human in image n=n+2 mark X over that pair 3. If human remain in image Then n=n+1 4. Return n

Engineer / Programmer จะเชื่อว่า Counting2 เร็วกว่า

ยังไงก็ไปไม่ถึง ∞

Scientist จะเชื่อว่าขั้นตอนวิธีทั้ง 2 เร็วเท่ากัน

∞ ยังเล็กไป



Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = ∞

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = ∞

n = 9

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = ∞

n = 9

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 9

n = 9

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 9

n = 6

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 9

n = 6

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 6

n = 6

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 6

n = 5

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 6

n = 5

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 5

n = 5

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 5

n = 3

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 5

n = 3

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 3

n = 3

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 3

n = 2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 3

n = 2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 9

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 9

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 8

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 8

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N=[9, 6, 5, 3, 2, 9, 8]

y = 2

n = 8

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการหาค่าน้อยสุด

ตั้งชื่อขั้นตอนวิธีว่า Min

กำหนดให้ N คือข้อมูล

y คือค่าน้อยสุด

Algorithm: Min
Input: N Output: y
1. y = ∞ 2. For each n in N IF n < y THEN y = n 3. Return y

N loops

ขั้นตอนวิธีนี้มีความซับซ้อนทางเวลาเท่าใด ?

O(n)

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

N=[9, 6, 5, 3, 2, 9, 8]

i=1

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

N=[9, 6, 5, 3, 2, 9, 8]

i=1 j=1

Algorithms:การออกแบบขั้นตอนวิธี

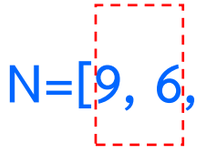
จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y



N=[9, 6, 5, 3, 2, 9, 8]

i=1 j=1

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

N=[9, 6, 5, 3, 2, 9, 8]

i=1 j=1

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

N=[6, 9, 5, 3, 2, 9, 8]

i=1 j=1

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

N=[6, 9, 5, 3, 2, 9, 8]

i=1 j=2

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

N=[6, 9, 5, 3, 2, 9, 8]

i=1 j=2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

N=[6, 5, 9, 3, 2, 9, 8]

i=1 j=2

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

N=[6, 9, 5, 3, 2, 9, 8]

i=1 j=3

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

ขั้นตอนวิธีนี้มีความซับซ้อนทางเวลาเท่าใด ?

```
i  j           N
1, 1, [6, 9, 5, 3, 2, 9, 8]
1, 2, [6, 5, 9, 3, 2, 9, 8]
1, 3, [6, 5, 3, 9, 2, 9, 8]
1, 4, [6, 5, 3, 2, 9, 9, 8]
1, 5, [6, 5, 3, 2, 9, 9, 8]
1, 6, [6, 5, 3, 2, 9, 8, 9]
2, 1, [5, 6, 3, 2, 9, 8, 9]
2, 2, [5, 3, 6, 2, 9, 8, 9]
2, 3, [5, 3, 2, 6, 9, 8, 9]
2, 4, [5, 3, 2, 6, 9, 8, 9]
2, 5, [5, 3, 2, 6, 8, 9, 9]
2, 6, [5, 3, 2, 6, 8, 9, 9]
3, 1, [3, 5, 2, 6, 8, 9, 9]
3, 2, [3, 2, 5, 6, 8, 9, 9]
3, 3, [3, 2, 5, 6, 8, 9, 9]
3, 4, [3, 2, 5, 6, 8, 9, 9]
3, 5, [3, 2, 5, 6, 8, 9, 9]
3, 6, [3, 2, 5, 6, 8, 9, 9]
4, 1, [2, 3, 5, 6, 8, 9, 9]
4, 2, [2, 3, 5, 6, 8, 9, 9]
4, 3, [2, 3, 5, 6, 8, 9, 9]
4, 4, [2, 3, 5, 6, 8, 9, 9]
4, 5, [2, 3, 5, 6, 8, 9, 9]
4, 6, [2, 3, 5, 6, 8, 9, 9]
5, 1, [2, 3, 5, 6, 8, 9, 9]
5, 2, [2, 3, 5, 6, 8, 9, 9]
5, 3, [2, 3, 5, 6, 8, 9, 9]
5, 4, [2, 3, 5, 6, 8, 9, 9]
5, 5, [2, 3, 5, 6, 8, 9, 9]
5, 6, [2, 3, 5, 6, 8, 9, 9]
6, 1, [2, 3, 5, 6, 8, 9, 9]
6, 2, [2, 3, 5, 6, 8, 9, 9]
6, 3, [2, 3, 5, 6, 8, 9, 9]
6, 4, [2, 3, 5, 6, 8, 9, 9]
6, 5, [2, 3, 5, 6, 8, 9, 9]
ตอบ 6, 6, [2, 3, 5, 6, 8, 9, 9]
```

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort

กำหนดให้ N คือข้อมูล

Y คือข้อมูลที่เรียงแล้ว

ขั้นตอนวิธีนี้มีความซับซ้อนทางเวลาเท่าใด ?

O(n) x O(n)

$= O(n^2)$

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

Loop ที่ 2 วน N-1 รอบ

Loop ที่ 1 วน N-1 รอบ

Algorithms:การออกแบบขั้นตอนวิธี

จงออกแบบวิธีการเรียงข้อมูลจากน้อยไปมาก

ตั้งชื่อขั้นตอนวิธีว่า Sort
กำหนดให้ N คือข้อมูล
Y คือข้อมูลที่เรียงแล้ว

$O(n^2)$

$O(n^2)$ มีหน้าตาอย่างไร ?

Algorithm: Sort
Input: N Output: y
1. For i = 1 to N-1 For j=1 to N-1 If N[j+1] < N[j] Then Swap(N[j] , N[j+1]) 2 Y=N 3. Return Y

n (จำนวนข้อมูล)	n^2 (จำนวนการวนซ้ำ)
1	1
10	100
100	10000
1000	1000000
10000	100000000
100000	10000000000
1000000	1E+12
10000000	1E+14
100000000	1E+16

ขั้นตอนวิธีการเรียงข้อมูลแบบต่าง ๆ

Name	Best	Average	Worst	Memory
Bubble sort	n	n^2	n^2	1
Cocktail sort	n	n^2	n^2	1
Comb sort	$n \log n$	n^2	n^2	1
Gnome sort	n	n^2	n^2	1
Insertion sort	n	n^2	n^2	1
Library sort	n	$n \log n$	n^2	n
Odd–even sort	n	n^2	n^2	1
Strand sort	n	n^2	n^2	n
UnShuffle Sort[10]	n	kn	kn	In-place for linked lists. n × sizeof(link) for array.
Quicksort	$n \log n$ variation is n	$n \log n$	n^2	$\log n$ on average, worst case space complexity is n; Sedgewick variation is $\log n$ worst case.
Cycle sort	n^2	n^2	n^2	1
Selection sort	n^2	n^2	n^2	1
In-place merge sort	—	—	$n \log^2 n$ See above, for hybrid, that is $n \log n$	1
Shell sort	$n \log n$	Depends on gap sequence	Depends on gap sequence; best known is $n^{4/3}$	1
Shell sort	$n \log n$	Depends on gap sequence	Depends on gap sequence; best known is $n^{4/3}$	1
Block sort	n	$n \log n$	$n \log n$	1
Cubesort	n	$n \log n$	$n \log n$	n
Patience sorting	n	—	$n \log n$	n
Smoothsort	n	$n \log n$	$n \log n$	1
Timsort	n	$n \log n$	$n \log n$	n
Binary tree sort	$n \log n$	$n \log n$	$n \log n$ (balanced)	n
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$
Merge sort	$n \log n$	$n \log n$	$n \log n$	n A hybrid block merge sort is O(1) mem.
Tournament sort	$n \log n$	$n \log n$	$n \log n$	n[9]
Heapsort	n If all keys are distinct, $n \log n$	$n \log n$	$n \log n$	1
Franceschini's method[11]	—	$n \log n$	$n \log n$	1

จากนี้ไปเราจะศึกษาขั้นตอนวิธีที่เกี่ยวข้องกับการเก็บข้อมูล

วิธีการที่ดีที่สุดสำหรับ

เพิ่มข้อมูล

ลบข้อมูล

ค้นหาข้อมูล

จัดรูปแบบ

ในสถานการณ์ต่าง ๆ

การบ้านที่ 1

จงออกแบบขั้นตอนวิธีในการหาค่าที่น้อยที่สุดเป็นลำดับ 2 โดยกำหนดให้

N คือ input เป็นจำนวนเต็ม มีจำนวนสมาชิกตั้งแต่ 2 จำนวนขึ้นไป

y คือค่าของคำตอบ

ให้แสดง Pseudo code พร้อมทั้งวิเคราะห์ความซับซ้อนทางเวลา (Big-O)

ตัวอย่างเช่น:

input: 1 2 3 4 5 6 7 8 9 10 ตอบ 2

Input: 10 9 5 3 2 2 2 2 1 5 9 3 3 0 0 1 ตอบ 1

Input: 11 15 32 0 5 5 5 7 7 6 ตอบ 5

ส่งเป็นแฟ้ม PDF หรือ jpeg เท่านั้น

พิมพ์ ,เขียนบนกระดาษแล้วถ่ายรูป ได้หมด ส่งทาง Microsoft Team