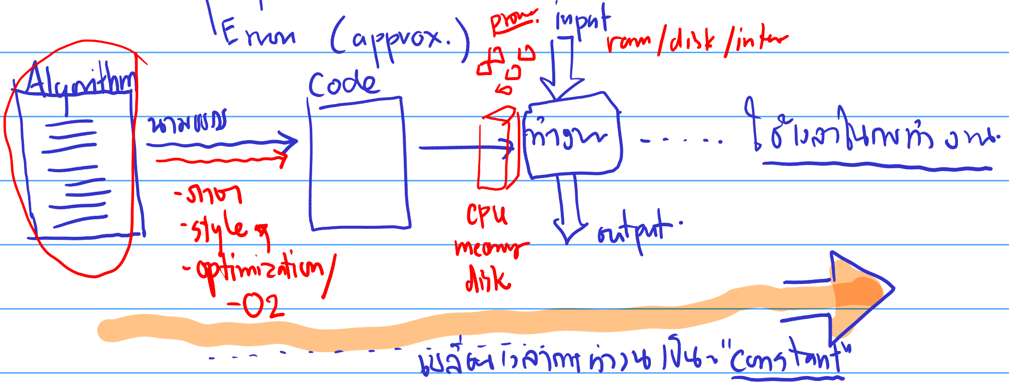
Algorithm Analysis
- ความถูกต้อง *
- เวลาการทำงาน (ประสิทธิภาพ-ด้านเวลา) → big O : ex. $O(n^2)$, $O(n \log n)$
- Space.
- Enum (approx.)



..... ใช้เวลาในการทำงาน.

ไม่สนใจเวลาการทำงานเป็น "constant"

แย่สุด

if

ex หาค่าสูงสุด

array

Input: A[1, 2, ..., n], integer n

Output. $Z = \max_{i=1}^{n} A[i]$

Best case ← Assumption → ของ input

Worst case

"Average case"

```
max ← A[1]
for i ← 2, 3, ..., n
    n-1   if A[i] > max
              ✓ max = A[i]
return max
```

ขอบเวลา ↓ Assumption

C_1

C_5 ต่อครั้ง n ครั้ง

C_4 สมมติว่า if ทำงานทุก...

C_3

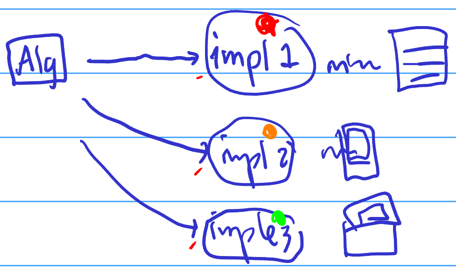
C_2

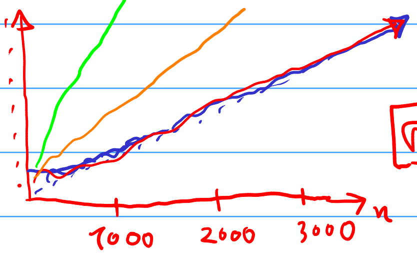
(C_4+C_3)

amortized

Running Time $C_1 + n \cdot C_5 + (n-1) \cdot [C_3 + C_4] + C_2$

$= n(C_5 + C_4 + C_3) + \cancel{\bigcirc}$





ควรจะพบ ค่า n มากพอ

เมื่อเพิ่ม n จาก X

เป็น 2X

เวลาจะเพิ่ม ~ 2 เท่า

Asymptotic analysis - อัตราการโต ของ function

- ค่าลิมิตของ algorithm

$\Theta(n)$, $\Theta(n \log n)$, $\Theta(n^2)$

ค่า "n มากพอ" ← จะดีกว่ามาก ← ดีกว่า / ที่แย่

$O(n)$ ทำงานเร็วกว่า $O(n^2)$

เมื่อ n ใหญ่มากพอ

$f: \mathbb{N} \to \mathbb{R}$

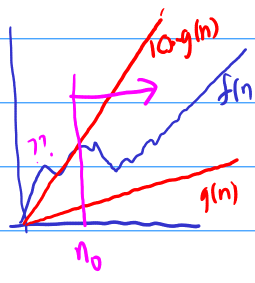
Definitions:

จะกล่าวว่า $f(n)$ เป็น $O(g(n))$ [เขียน $f(n) = O(g(n))$]

ถ้ามีค่าคงที่ c และ n_0 ที่

$$f(n) \le c \cdot g(n)$$

เมื่อ $n \ge n_0$



$\boxed{ex}$ จงแสดงว่า $n^2+200n+5000 = O(n^2)$

พิจารณาว่า $n^2+200n+5000 \le n^2+200n^2+5000n^2$

$= \boxed{5201}\, n^2$ เมื่อ $n \ge \boxed{1}$

ดังนั้นมี $c=5201$, $n_0=1$ ที่สอดคล้องกับนิยาม^{c}ว่า $n^2+200n+5000 = O(n^2)$.^{n_0}

$\boxed{ex}$ ให้ $f(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_0$ เป็น polynomial บน n ที่ degree d. $f(n) = O(n^d)$. (h.w.)

$\boxed{ex}$ พิสูจน์ว่า $f(n), g(n), h(n)$

(1) ถ้า $f(n)=O(h(n))$, $g(n)=O(h(n))$ จะได้ว่า $f(n)+g(n)=O(h(n))$

(2) ถ้า $f(n)=O(g(n))$, $g(n)=O(h(n))$ จะได้ว่า $f(n)=O(h(n))$ (transitive)

(1) เนื่องจาก

$f(n)=O(h(n))$ จึงมี c_1, n_1 ที่

$f(n) \le c_1 \cdot h(n)$ เมื่อ $n \ge n_1$.

$g(n) = O(h(n))$ จึงมี c_2, n_2 ที่

$g(n) \le c_2 \cdot h(n)$ เมื่อ $n \ge n_2$.

ดังนั้นจะได้ว่า

$$f(n)+g(n) \le c_1\cdot h(n) + c_2 \cdot h(n) \quad \text{เมื่อ } n \ge \underset{n_0}{\boxed{\max(n_1,n_2)}}$$
$$= \underset{c}{\boxed{(c_1+c_2)}} \cdot h(n)$$

O, Ω

Θ

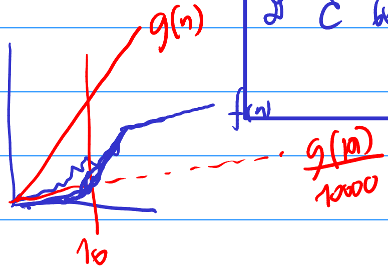
"o", "ω"

(2) (h.w.)

<u>Definitions</u>

$f(n)$ เป็น $\Omega(g(n))$ $[\, f(n) = \Omega(g(n)) \,]$ ถ้า มี c และ n_0 ที่

$f(n) \ge c \cdot g(n)$ เมื่อ $n \ge n_0$



ถ้า $f(n) = O(g(n))$ และ $f(n) = \Omega(g(n))$ จะได้ว่า

$f(n) = \Theta(g(n))$.

Properties

(1) $f(n) = O(g(n))$ จะได้ว่า $g(n) = \Omega(f(n))$

ถ้า $f(n)$ เป็น polynomial degree d ที่มีสัมประสิทธิ์ตัวหน้าสุดของพจน์ n^d มากกว่า 0,

$$f(n) = \Omega(n^d)$$

และ $f(n) = \Theta(n^d)$

ถ้า $f(n) = O(g(n))$, $h(n) = O(g'(n))$, $\Rightarrow f(n) \cdot h(n) = O(g(n) \cdot g'(n))$

Polynomials, exponentials, logarithms.

for any $c > 0$

$$\log^c n = O(n^{\varepsilon}) \quad \text{for any } \varepsilon > 0$$

for any $c > 0$

$$n^c = O(a^n) \quad \text{for any } a > 1.$$

Sorting algorithms

Input: Array $A[1, \dots, n]$

Selection sort

```
for $i \leftarrow 1, \dots, n-1$          // ทำ n-1 รอบ
    $j \leftarrow$ minIndex$(A, i, n)$    $\boxed{\frac{O(n-i)}{O(n)}}$ = , $O(n)$
    swap$(A[i], A[j])$
Total: $O(n^2)$
```

ของที่น้อยที่สุด จาก $A[i]$ — $A[n]$

$\le O(n-i) \le c(n-i)$
สำหรับ c

Insertion sort:

```
for $i \leftarrow 2, \dots, n$          // ทำ n-1 รอบ
    $k \leftarrow i$                     $\boxed{O(i) = O(n)}$
    insert $A[k]$ into sorted list $A[1], \dots, A[i-1]$
Total: $O(n^2)$
```

$f = \Theta(g)$

$h = \Theta(g)$ ←

$e = \Theta(g)$

$f + h + e \neq O(g)$

$$\sum_{i=1}^{n-1} c(n-i) = \underbrace{\overset{c}{O}(n)}_{O(n)} + \underbrace{\overset{c}{O}(n-1)}_{O(n)} + \underbrace{\overset{c}{O}(n-2)}_{O(n)} + \cdots + O(1)$$

$$\cancel{= O(n)}$$

$$= c \sum_{i=1}^{n-1} (n-i) = c \left[\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i \right]$$

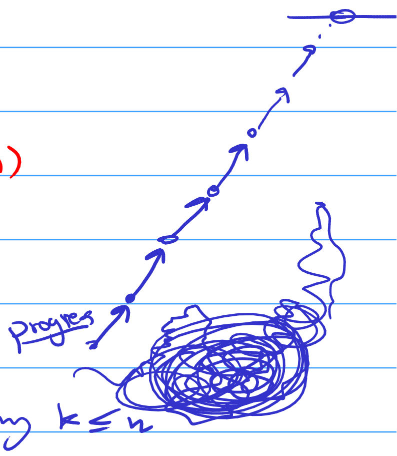
$$\sum_{i=1}^{n} c \cdot i = c \frac{n(n+1)}{2} = \Theta(n^2)$$

$$= cn^2 - c\left(\frac{n(n+1)}{2}\right)$$

$$= \Theta(n^2)$$

Bubble Sort

```
done ← false
while not done :            ← ทำกี่รอบ?
    done ← true
    for i ← 1, ..., n−1                                  | Θ(n)
        if A[i] > A[i+1]   ✓                   | Θ(1)
            swap(A[i], A[i+1])       | Θ(1)
            done ← false
```



ALGORITHM ⟶ "PROGRESS"

Claim: หลังจากทำงาน while loop k รอบ for any $k \le n$
ข้อมูลมากสุด $\ge k$ ตัว จะอยู่ในตำแหน่งที่ถูกต้อง.
(prove by induction)

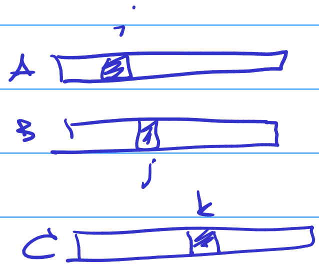
Cor: while loop ทำงานไม่เกิน n รอบ และ:
Bubble sort ใช้เวลาทำงานในกรณีที่แย่สุด $O(n^2)$

Merge

Input : A $[1, 2, \dots, n]$ sorted น้อยไปมาก
B $[1, 2, \dots, m]$ sorted น้อยไปมาก

Output : C $[1, \dots, n+m]$ ผลการ merge A & B sorted น้อยไปมาก

```
i ← 1 ; j ← 1 ; k ← 1
while i ≤ n & j ≤ m
    if (j > m) or ((i ≤ n) and (A[i] ≤ B[j]))                          Θ(1)
        → C[k] ← A[i] ;   k++ ; i++ ;     | Θ(1)
    else
        → C[k] ← B[j] ;   k++ ; j++ ;     | Θ(1)
```



Stable matching

PROGRESS (1) i ไม่ลด , j ไม่ลด , หลังเปรียบเทียบ i & j เพิ่ม
⇒ ไม่มีคู่ A[i] , B[j] โดนเปรียบเทียบเกิน 1 ครั้ง
if เปรียบเทียบไม่เกิน จน.คู่ $= n \cdot m$ คู่ → $O(n \cdot m)$
ถูก แต่ไม่ tight

(2) ทุกครั้งที่เปรียบเทียบ จะมี
ข้อมูล 1 ตัว ย้ายไปใส่ใน $\boxed{C}$
แต่ จน.ข้อมูลใน C เริ่มต้น $= 0$, จบที่ $n+m$ ตัว
if จะเพิ่มได้ไม่เกิน $n+m$ ตัว ; loop ทำไม่เกิน $n+m$ รอบ
เวลา $\boxed{O(n+m)}$ ←