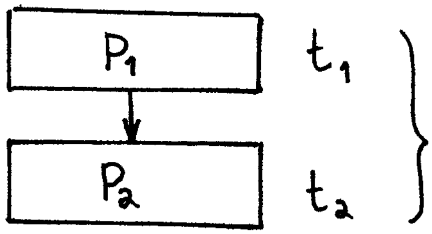
Analysis of Algorithms

Control Structures

- Sequencing



$$t_1 + t_2 = \Theta(\max(t_1, t_2))$$

- "For" loop

```
for i ← 1 to m do P(i)
```

ถ้า P(i) ใช้เวลา t_i $\rightarrow \sum_{i=1}^{m} t_i$

ถ้า P(i) ใช้เวลา t $\rightarrow \sum_{i=1}^{m} t = mt = \Theta(mt)$

- "While" loop

```
BinarySearch(T[1..n], x)
    i ← 1 ; j = n
    while i < j do
        m ← (i+j)/2
        case  x < T[m] :  j ← m-1
              x > T[m] :  i ← m+1
              x = T[m] :  i, j ← m
    return i
```

ให้ d_k คือค่าของ $j-i+1$ หลังจบรอบที่ k

$$d_k \le d_{k-1}/2 \qquad k \ge 1$$

$$d_0 = n$$

ได้ $d_k \le n/2^k$

while loop จบเมื่อ $d \le 1$

$\Big\}\ k \le \lceil \lg n \rceil$

การทำงานในแต่ละรอบใช้เวลาคงที่ $\quad c \cdot \lceil \lg n \rceil$

$\therefore$ Binary search ใช้เวลาเป็น $O(\log n)$

<u>ต.ย.</u> <u>Euclid's algorithm</u>

หา ห.ร.ม. (Greatest common divisor)

```
GCD( m, n )
    while m > 0 do
        t ← m
        m ← n mod m
        n ← t
    return n
```

$O(\log n)$

ให้ $n \ge m$ จะพิสูจน์ว่า $n \bmod m < n/2$ เสมอ.

① ถ้า $m > n/2$ จะได้ $1 \le n/m < 2$ ดังนั้น $n \div m = 1$

$$\begin{aligned} \therefore\ n \bmod m &= n - m \times (n \div m) \\ &= n - m \\ &< n - n/2 = n/2 \end{aligned}$$

② ถ้า $m \le n/2$ จะได้ $n \bmod m < m \le n/2$

```
GCD(m, n)
    i ← min(m, n) + 1
    repeat
        i ← i - 1
    until i divides both m and n
            exactly
    return i
```

n	m	t
52	16	
16	4	16
4	0	4

n	m	t
128	84	
84	44	84
44	40	44
40	4	40
4	0	4

n	m	t
88	128	
128	88	128
88	40	88
40	8	40
8	0	8

n	m	t
55	34	
34	21	34
21	13	21
13	8	13
8	5	8
5	3	5
3	2	3
2	1	2
1	0	1

ต.ย. Selection sort

```
Select ( T[1..n] )
    for i ← n downto 2 do
        maxj ← 1
        for j ← 1 to i do
            if T[j] > T[maxj] then maxj ← j
        Swap( T, maxj , i )
```

เลือก "barometer" :

คำสั่งที่ทำงานเป็นจำนวนครั้งไม่น้อยกว่าคำสั่งอื่นๆ

$$T[j] > T[maxj]$$

$$\sum_{i=2}^{n} i = 2+3+4+\dots+n = \frac{n(n+1)}{2} - 1$$

$$= \frac{n^2}{2} + \frac{n}{2} - 1 \quad \in \quad \Theta(n^2)$$

Average-case Analysis

```
Insertion_Sort ( A[1..n] )
  for j <- 2 to n
     do key <- A[j]
        i <- j-1
        while i > 0 and A[i] > key
           do A[i+1] <- A[i]
              i <- i-1
        A[i+1] <- key
```

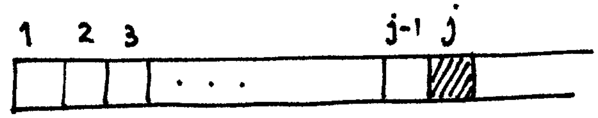
barometer : "$i > 0$ and $A[i] > key$"

worst case : $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1 = \Theta(n^2)$

best case : $\sum_{j=2}^{n} 1 = n - 1 = \Theta(n)$

average case :

① $\sum \frac{1}{n!} \times$ (เวลาในการเรียง instance i)

② 

รอบที่ j ทำงานคำสั่ง "$i > 0$ and $A[i] > key$"

ตั้งแต่ 1 ครั้ง ถึง j ครั้ง

$$\therefore = \sum_{j=2}^{n} \left(\sum_{k=1}^{j} \frac{1}{j} \times k \right) = \sum_{j=2}^{n} \left(\frac{j+1}{2} \right) = \frac{(n-1)(n+4)}{2} = \Theta(n^2)$$

Amortized Analysis

$$\boxed{\text{for } i \leftarrow 1 \text{ to } n \text{ do } P}$$

ถ้า P ใช้เวลา $\Theta(\log n)$ worst case

$\therefore$ ใช้เวลาทั้งหมด $O(n \log n)$ สำหรับ วงวนข้างบน.

แต่ อาจใช้เวลา เร็วกว่า ใน worst case

ต.ย. Binary counter : Increment

	b_3	b_2	b_1	b_0		
	0	0	0	0		0
	0	0	0	1	1	1
	0	0	1	0	2	3
	0	0	1	1	1	4
	0	1	0	0	3	7
	0	1	0	1	1	8
	0	1	1	0	2	10
	0	1	1	1	1	11
	1	0	0	0	4	15
	1	0	0	1	1	16
	1	0	1	0	2	18
	1	0	1	1	1	19
	1	1	0	0	3	22
	1	1	0	1	1	23
	1	1	1	0	2	25
	1	1	1	1	1	26
1	0	0	0	0	5	31

- ถ้ามี m บิต

  worst case : $\Theta(m)$

  (เปลี่ยนทุกบิต)

- Increment n ครั้ง

  $$\boxed{O(nm)}$$

- Increment n ครั้ง

  b_0 เปลี่ยน n ครั้ง

  b_1 " $\lfloor n/2 \rfloor$ "

  b_2 " $\lfloor n/4 \rfloor$ "

  $\vdots$

  b_{m-1} " $\lfloor n/2^{m-1} \rfloor$

  $$\sum_{i=0}^{m-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

  $$\boxed{O(n)}$$

Recursive calls / Recurrences

```
FibRec(n)
    if n < 2 then return n
    else return ( FibRec(n-1) + FibRec(n-1) ) % 1024
```

$$T(n) = \begin{cases} a & \text{if } n=0 \text{ or } n=1 \\ T(n-1)+T(n-2)+b & \text{otherwise} \end{cases}$$

$$T(n) = T(n-1)+T(n-2)+O(1) \qquad n>1$$

```
BinSearch(T, l, r, x)
    m <- (l+r)/2
    case x < T[m]: return BinSearch(T, l, m-1, x)
         x > T[m]: return BinSearch(T, m+1, r, x)
         x = T[m]: return m
```

$$T(n) = T(n/2) + O(1)$$

$$T(n) = T(n/2) + 1$$

```
Select ( T, n )
    if (n = 1 ) return
    maxj ← 1
    for j ← 2 to n do
        if T[j] > T[maxj] then maxj ← j
    Swap( T, maxj , n )
    Select ( T, n-1 )
```

$$T(n) = T(n-1) + \Theta(n)$$

$$T(n) = T(n-1) + n$$

```
Pow ( x , n )                              x^n
    if (n = 0)  return 1
    if (n = 1)  return x
    if (even(n))
        return( Pow (x*x, n/2))
    else
        return( Pow( x*x , n/2 ) * x )
```

$x^{62} = (x^2)^{31}$ $\qquad$ $x^{63} = (x^2)^{31} \cdot x$

$$T(n) = T(n/2) + 1$$