

# **Design and Analysis of Algorithms**

**การออกแบบและวิเคราะห์อัลกอริทึม**

สมชาย ประสิทธิ์จุตระกูล

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

2541

## ເລື່ອງນັ້ນ

1. Brassard and Bratley

"Fundamentals of Algorithmics", Prentice-Hall, 1996

2. Cormen, Leiserson, and Rivest

"Introduction to Algorithms", The MIT Press , 1990

## ໂທ: 11616

ການປົກ 20%

ສອບການງານ 40%

ສອບປາກງານ 40%

## ຄວາມຮູ້ເປື້ອງຕົງ

- ການໄດ້ຮັບໄປໃຫຍ່ການງານທາງຄວາມຮູ້ເປື້ອງຕົງ
- ຖອນກົດກຳສະໜັບ .

## ການສົ່ງ

somchaip@chula.ac.th

ທີ່ມະນຸຍາ ປະຈິບປະຈິບ

# Analysis & Design of Algorithms

## Unit

1. Elementary Algorithmics
2. Analysis of Algorithms
3. Data Structures
4. Greedy Algorithms
5. Divide-and-Conquer
6. Dynamic Programming
7. Exploring Graphs
8. Probabilistic Algorithms
9. Computational Complexity
10. Approximation Algorithms

# Algorithms (アルゴリズム, ᴬʳ्गोरिझମ୍)

- a tool for solving a well-specified computational problem.
  - a sequence of computational steps that transform the input into the output.
- 

## Sorting problem:

Input: a sequence of  $n$  numbers

$$\langle a_1, a_2, \dots, a_n \rangle$$

Output: a reordering  $\langle a'_1, a'_2, \dots, a'_n \rangle$

of the input sequence such that

$$a'_1 \leq a'_2 \leq \dots \leq a'_n$$

e.g.  $\langle 31, 41, 59, 26, 41, 58 \rangle \leftarrow$



$$\langle 26, 31, 41, 41, 58, 59 \rangle$$

an instance  
of the sorting  
problem.

---

An algorithms is correct if, for every input instance, it halts with the correct output.

## Elementary Operation

m.v. Wilson's theorem :

" the integer  $n$  divides  $(n-1)! + 1$   
if and only if  $n$  is prime for all  $n > 1$ "

Wilson(  $n$  )

if  $n$  divides  $(n-1)! + 1$  exactly then return true  
else return false

m.v.  $x \leftarrow \min \{ T[i] \mid 1 \leq i \leq n \}$

Elementary operation :

operation whose execution time can be bounded  
above by a constant depending only on the  
particular implementation used

- machine
- programming lang.
- compiler

m.v. នូវសារធិនា តាម  $\mu s$   
msg =  $\dots$   $t_m$   $\dots$   
assignment =  $t_s$   $\dots$

ចំណែកអេឡិចត្រូនុយ៉ាងៗ នូវសារ ដែល  $\mu s$ , msg និង assignment គឺ

$$\begin{aligned}
 t &= a \cdot t_a + m \cdot t_m + s \cdot t_s \\
 &\leq \max(t_a, t_m, t_s) \times (a+m+s) \\
 &\propto (a+m+s)
 \end{aligned}$$

∴ ມີຄວາມສໍາເລັດໃນການອະນຸຍາກ (ໃຫຍ່ລົກ) ທີ່ມີກົງ  
ທີ່ມີບໍ່ elementary operations ນີ້ນີ້ຈະ

---

### Q.8. Fibonacci number

$$f_n = \begin{cases} n & \text{if } n=0,1 \\ f_{n-1} + f_{n-2} & n \geq 2. \end{cases}$$

n	0	1	2	3	4	5	6	7	8	9
$f_n$	0	1	1	2	3	5	8	13	21	34

FibRec( n )

```

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

FibIter( n )

$i \leftarrow 1 ; j \leftarrow 0$

```

for k  $\leftarrow 1$  to n do j  $\leftarrow i+j$ 
                           i  $\leftarrow j-i$ 
    
```

return j

$n$	10	20	30	50	100	III Sem 2
FibRec	8 ms.	1 s.	2 min.	21 days	$10^9$ yrs.	$\phi^n$
FibIter	$\frac{1}{6}$ ms.	$\frac{1}{3}$ ms.	$\frac{1}{2}$ ms.	$\frac{3}{4}$ ms.	$1\frac{1}{2}$ ms.	$n$

ခြေခံနည်

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^3$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^4$$

တောက်  $f_n$   $u_n$   $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$

တော်များ  $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$  အတွက် 150

က.ပ.  $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{20} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{10} \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{10}$

$\boxed{\text{III Sem 2}} \lg n$   $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{10} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^5 \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^5$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^5 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

# Sorting Algorithms

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Heap sort
- Merge sort
- Quick sort

INSERTION-SORT ( A )

	Times
1    for $j \leftarrow 2$ to $\text{length}[A]$	$n$
2       do $\text{key} \leftarrow A[j]$	$n - 1$
3 $i \leftarrow j - 1$	$n - 1$
4              while $i > 0$ and $A[i] > \text{key}$	$\sum_{j=2}^n t_j$
5                  do $A[i+1] \leftarrow A[i]$	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i - 1$	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow \text{key}$	$n - 1$

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j + c_5 \sum_{j=2}^n (t_j - 1)$$

$$+ c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1)$$


---

លោកស្រាវជ្រាវ  $T(n)$  នឹងបានចំណាំ

បន្ថែម: ស្ថាបន្ទាត់ទិន្នន័យ

Best case: ក្រឡាក់ទិន្នន័យដែលត្រូវបានបញ្ជាក់

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1)$$

$$= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

"linear function"

Worst case: ក្រឡាក់ទិន្នន័យដែលត្រូវបានបញ្ជាក់ (បានរាយការណ៍)

$$t_j = j$$

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

$$\sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4\left(\frac{n(n+1)}{2} - 1\right) + c_5\left(\frac{n(n-1)}{2}\right)$$

$$+ c_6\left(\frac{n(n-1)}{2}\right) + c_7(n-1)$$

$$= \left(\frac{c_4 + c_5 + c_6}{2}\right)n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4 - c_5 - c_6 + c_7}{2}\right)n$$

$$- (c_2 + c_3 + c_4 + c_8) \quad \text{"Quadratic function"}$$

# Growth of Functions

- gives a simple characterization of the algorithm's efficiency
- allows us to compare the relative performance of alternative algorithms

## Asymptotic Notation

- permits substantial simplification
- deals with the behaviour of functions in the limit, that is for sufficiently large values of its parameter.

"the order of"

Let  $t: \mathbb{N} \rightarrow \mathbb{R}^{>0}$  and  $f: \mathbb{N} \rightarrow \mathbb{R}^{>0}$

e.g.  $t(n) = 27n^2 + \frac{355}{133}n + 12$

$$f(n) = n^2$$

$t(n)$  is in  
the order of  $n^2$

$$\begin{aligned} t(n) &\leq 27n^2 + \frac{355}{133}n^2 + 12n^2 \\ &= 42\frac{16}{133}n^2 \\ &= 42\frac{16}{133}f(n) \end{aligned} \quad \left. \right\} t(n) \leq c f(n)$$

## O-notation

"asymptotic upper bound"

$O(f(n)) = \{t(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq t(n) \leq cf(n) \text{ for all } n \geq n_0\}$

Q.U.

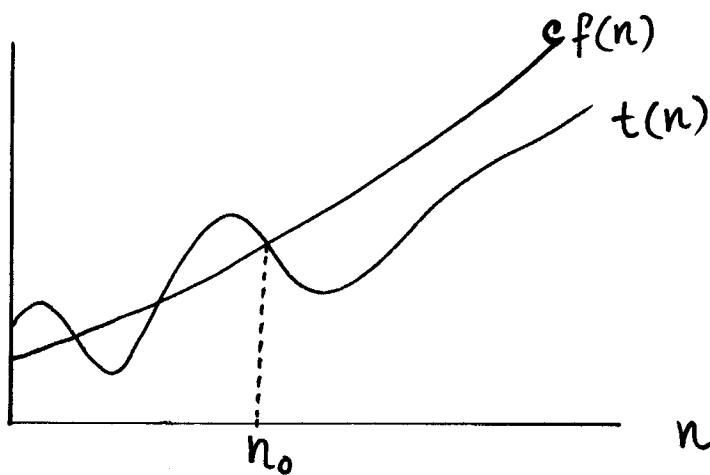
$$\overline{O}(n^2) = \{n, \frac{1}{5}n, \sqrt{n}, 30n^2, 7n^2 + 5n - 2, \dots\}$$

Running time of insertion sort is  $O(n^2)$

"—" FibRec is  $O(\phi^n)$

"—" FibIter is  $O(n)$

"—" FibIter is  $O(n^9)$



$$t(n) \in O(f(n))$$

## $\Omega$ -notation

"asymptotic lower bound"

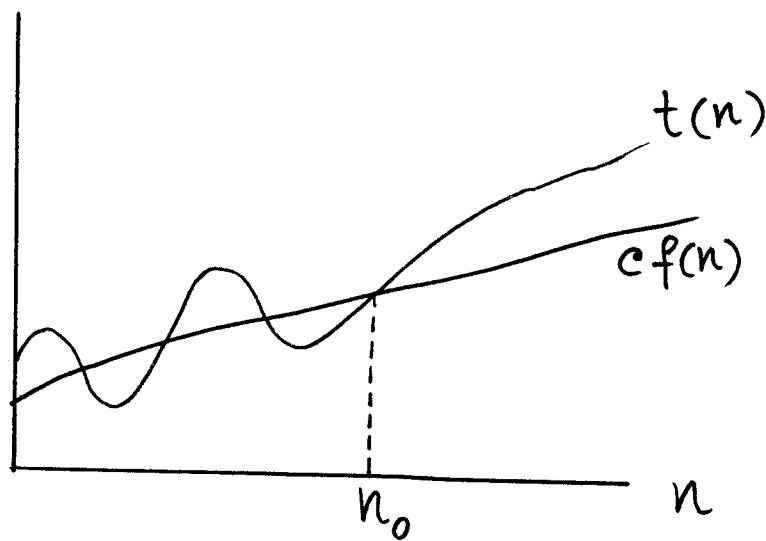
$\Omega(f(n)) = \{ t(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq c \cdot f(n) \leq t(n) \text{ for all } n \geq n_0 \}$

M.V.

$$\Omega(n^2) = \{ n^2, 5n^2 - 7n, n^{99}, \frac{1}{1000}n^9, \dots \}$$

Running time of insertion sort is  $\Omega(n)$

$$t(n) \in \Omega(f(n)) \text{ if and only if } f(n) \in O(t(n))$$



$$t(n) \in \Omega(f(n))$$

## $\Theta$ -notation

"asymptotically tight bound"

$\Theta(f(n)) = \{ t(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 f(n) \leq t(n) \leq c_2 f(n) \text{ for all } n \geq n_0 \}$

Ex.

$$\Theta(n^2) = \{ 3n^2, n^2 - \log n, \frac{1}{1000}n^2 + 100n + 1, \dots \}$$

$\frac{1}{2}n^2 - 3n \in \Theta(n^2)$  ? un  $c_1, c_2$  and  $n_0$

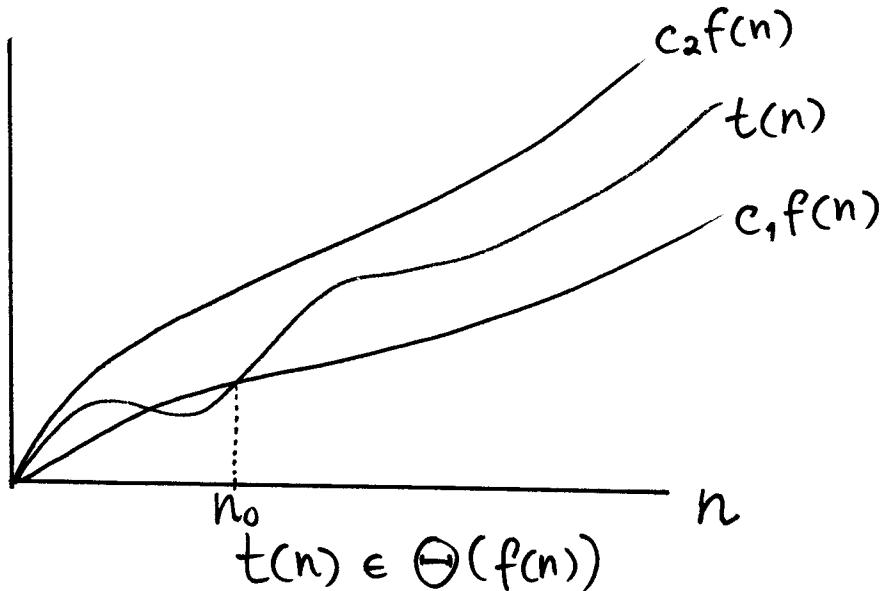
$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$$

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

$$\textcircled{1} \quad \frac{1}{2} - \frac{3}{n} \leq c_2 \quad \text{for } n \geq 1 \quad \text{so } c_2 \geq \frac{1}{2}.$$

$$\textcircled{2} \quad c_1 \leq \frac{1}{2} - \frac{3}{n} \quad \text{for } n \geq 7 \quad \text{so } c_1 \leq \frac{1}{14}.$$

$$\therefore \text{such } c_1 = \frac{1}{14}, c_2 = \frac{1}{2}, \text{ for } n_0 = 7$$



For any two functions  $f(n)$  and  $g(n)$ ,

$f(n) = \Theta(g(n))$  if and only if  $f(n) = O(g(n))$   
and  $f(n) = \Omega(g(n))$

### O-notation

$O(f(n)) = \{t(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq t(n) < cf(n) \text{ for all } n \geq n_0\}$

$$\lim_{n \rightarrow \infty} \frac{t(n)}{f(n)} = 0$$

Ex.  $2n \in O(n^2)$

$2n^2 \notin O(n^2)$

---

### $\omega$ -notation

$\omega(f(n)) = \{t(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq cf(n) < t(n) \text{ for all } n \geq n_0\}$

$$\lim_{n \rightarrow \infty} \frac{t(n)}{f(n)} = \infty$$

Ex.  $\frac{n^2}{2} \in \omega(n)$ ,  $\frac{n^2}{2} \notin \omega(n^2)$

M.V.  $\sum_{i=1}^n i^k \in \Theta(n^{k+1})$

พิสูจน์ ①  $\sum_{i=1}^n i^k \leq \sum_{i=1}^n n^k = n^{k+1} = O(n^{k+1})$

②  $\sum_{i=1}^n i^k \geq \sum_{i=\lceil \frac{n}{2} \rceil}^n i^k$  <sup>10</sup>

$$\geq \sum_{i=\lceil \frac{n}{2} \rceil}^n \left(\frac{n}{2}\right)^k$$

$$\geq \frac{n}{2} \cdot \left(\frac{n}{2}\right)^k = \frac{n^{k+1}}{2^{k+1}} = \Omega(n^{k+1})$$


---

M.V.  $n^b$  ที่สูงกว่า  $a^n$   $a, b - \text{real constants}$   
 $a > 1$

$$\lim_{n \rightarrow \infty} \frac{n^b}{a^n} = 0$$

$$\therefore n^b = o(a^n)$$

สรุปผล any positive exponential function ( $a^n$ ) grows faster than any polynomial.

Note: A polynomial in  $n$  of degree  $d$  is a function

$$p(n) = \sum_{i=0}^d a_i n^i \quad a_d \neq 0$$


---

Q.E.D.

$$\lg^b n \quad \text{is} \leq n^c$$

$$\lg^b n = (\lg n)(\lg n) \dots (\lg n)$$

*b times*

$$\text{Then } \lim_{n \rightarrow \infty} \frac{n^b}{2^n} = 0$$

$$\text{thus } n \leq \lg n$$

$$\text{thus } a \leq 2^c$$

$$\text{So } \lim_{n \rightarrow \infty} \frac{\lg^b n}{(2^c)^{\lg n}} = \frac{\lg^b n}{n^{\lg 2^c}} = \frac{\lg^b n}{n^c} = 0$$

$$\therefore \lg^b n = o(n^c)$$

Result for any constant  $c > 0$ ,  
any positive polynomial function  
grows faster than any polylogarithmic function.

---

$$\textcircled{1} \quad f, g: \mathbb{N} \rightarrow \mathbb{R}^{>0} \quad O(f(n) + g(n)) = O(\max(f(n), g(n)))$$

$$\textcircled{2} \quad \sum_{k=1}^n O(f(k)) = O\left(\sum_{k=1}^n f(k)\right)$$

$$\textcircled{3} \quad \log n! \in \Theta(n \log n)$$

$$\text{Stirling's formula} \quad n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

$$\textcircled{4} \quad O(1)$$

$$\textcircled{5} \quad 2^n, n^n, n!$$

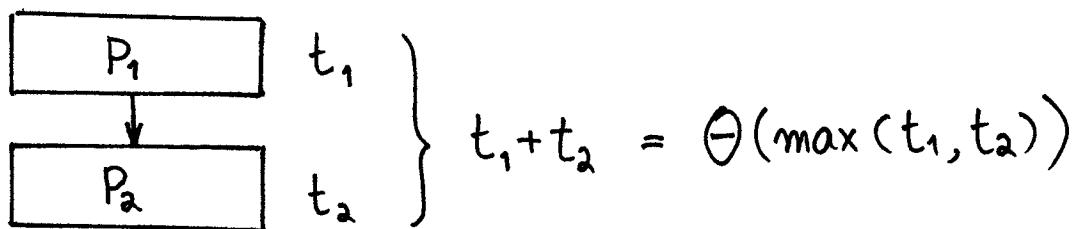
$$\textcircled{6} \quad 2^{n+1} \in O(2^n) ?$$

$$2^{2n} \in O(2^n) ?$$

# Analysis of Algorithms

## Control Structures

- Sequencing



- "For" loop

for  $i \leftarrow 1$  to  $m$  do  $P(i)$

$$\text{in } P(i) \text{ from } t_i \rightarrow \sum_{i=1}^m t_i$$

$$\text{in } P(i) \text{ from } t \rightarrow \sum_{i=1}^m t = mt = \Theta(mt)$$

- "while" loop

BinarySearch( $T[1..n]$ ,  $x$ )

$i \leftarrow 1 ; j = n$

while  $i < j$  do

$m \leftarrow (i+j)/2$

case  $x < T[m] : j \leftarrow m-1$

$x > T[m] : i \leftarrow m+1$

$x = T[m] : i, j \leftarrow m$

return  $i$

ຖី  $d_k$  តើទ្វារា  $j-i+1$  អស់ចាប់ដូចជា  $k$

$$d_k \leq d_{k-1}/2 \quad k \geq 1$$

$$d_0 = n$$

យើរ  $d_k \leq n/2^k$       }  $k \leq \lceil \lg n \rceil$   
while loop នូវខ្លះ  $d \leq 1$

ក្រសក៍ការងារ 96 || ផែនវឌ្ឍន៍ 96 គឺជាថាក់       $c \cdot \lceil \lg n \rceil$

∴ Binary search 96 គឺជាឧម្យ  $O(\log n)$

### ៣.៤. Euclid's algorithm

ឬ G.C.D. (Greatest common divisor)

GCD( $m, n$ )

```
while m > 0 do
    t ← m
    m ← n mod m
    n ← t
return n
```

$O(\log n)$

ឬ  $n \geq m$  នៅពេលណាដំឡូងវិញ  $n \mod m < n/2$  នៅទេ.

① កំពុង  $m > n/2$  នៅពេង  $1 \leq n/m < 2$  ទៀតនៅ  $n \div m = 1$

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

② កំពុង  $m \leq n/2$  នៅពេង  $n \mod m < m \leq n/2$

$\text{GCD}(m, n)$

$i \leftarrow \min(m, n) + 1$

repeat

$i \leftarrow i - 1$

until  $i$  divides both  $m$  and  $n$   
exactly

return  $i$

<u><math>n</math></u>	<u><math>m</math></u>	<u><math>t</math></u>
52	16	
16	4	16
④	0	4

<u><math>n</math></u>	<u><math>m</math></u>	<u><math>t</math></u>
88	128	
128	88	128
88	40	88
40	8	40
⑧	0	8

<u><math>n</math></u>	<u><math>m</math></u>	<u><math>t</math></u>
128	84	
84	44	84
44	40	44
40	4	40
④	0	4

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

## 3.5. Selection sort

Select (  $T[1..n]$  )

for  $i \leftarrow n$  down to 2 do

$\text{max}_j \leftarrow 1$

        for  $j \leftarrow 1$  to  $i$  do

            if  $T[j] > T[\text{max}_j]$  then  $\text{max}_j \leftarrow j$

        Swap( $T, \text{max}_j, i$ )

ເລືອນ "barometer":

ດຳສັ່ງກໍ່ກຳນາໄປເປົ້າຈຳນວນກວ່າໃໝ່ແລ້ວກຳນົດດຳສັ່ງອັນງ

$T[j] > T[\text{max}_j]$

$$\begin{aligned} \sum_{i=2}^n i &= 2+3+4+\dots+n = \frac{n(n+1)}{2}-1 \\ &= \frac{n^2}{2} + \frac{n}{2} - 1 \in \Theta(n^2) \end{aligned}$$

## 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
    
```

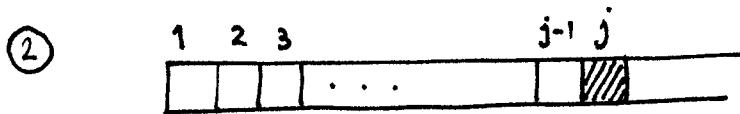
barometer : "  $i > 0$  and  $A[i] > \text{key}$ "

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



สมมุติ  $j$  ทำหน้าที่ "  $i > 0$  and  $A[i] > \text{key}$ "

ต้องมี 1 ตัว ถ้า  $i < j$

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

# Amortized Analysis

for  $i \leftarrow 1$  to  $n$  do  $P$

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

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

แต่จริงๆ ใช้เวลาเร็วกว่า worst case

## 7.5. Binary counter : Increment

$b_3$	$b_2$	$b_1$	$b_0$			
0	0	0	0	1	0	
0	0	0	1	2	1	• ถ้า $m$ บิต
0	0	1	0	1	3	worst case : $\Theta(m)$
0	0	1	1	3	4	(เปลี่ยนทุกปี)
0	1	0	0	1	7	
0	1	0	1	2	8	• Increment $n$ ครั้ง
0	1	1	0	1	10	$O(nm)$
0	1	1	1	4	11	• Increment $n$ ครั้ง
1	0	0	0	1	15	$b_0$ เป็นหนึ่ง $n$ ครั้ง
1	0	0	1	2	16	$b_1$ " $[n/2]$ "
1	0	1	0	1	18	$b_2$ " $[n/4]$ "
1	0	1	1	3	19	:
1	1	0	0	1	22	$b_{m-1}$ " $[n/2^{m-1}]$
1	1	0	1	2	23	
1	1	1	0	1	25	$\sum_{i=0}^{m-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$
1	1	1	1	5	26	
1	0	0	0		31	$O(n)$

## Recursive calls / Recurrences

FibRec( $n$ )

if  $n < 2$  then return  $n$

else return ( $\text{FibRec}(n-1) + \text{FibRec}(n-2)$ ) % 1024

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

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

BinSearch( $T, l, r, x$ )

$$m \leftarrow (l+r)/2$$

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

$\max_j \leftarrow 1$

for  $j \leftarrow 2$  to  $n$  do

    if  $T[j] > T[\max_j]$  then  $\max_j \leftarrow j$

Swap( $T$ ,  $\max_j$ ,  $n$ )

Select( $T$ ,  $n-1$ )

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

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

Pow(  $x$  ,  $n$  )

if ( $n = 0$ ) return 1

if ( $n = 1$ ) return  $x$

if (even( $n$ ))

    return ( Pow( $x*x$ ,  $n/2$ ) )

else

    return ( Pow( $x*x$ ,  $n/2$ ) \*  $x$  )

$$x^{62} = (x^2)^{31}$$

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

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

# Solving Recurrences

## Master method

$$T(n) = aT(n/b) + f(n) \quad a \geq 1, b > 1$$

---

① If  $f(n) = O(n^{\log_b a - \epsilon})$  for some constant  $\epsilon > 0$

$$T(n) = \Theta(n^{\log_b a})$$

---

② If  $f(n) = \Theta(n^{\log_b a})$

$$T(n) = \Theta(n^{\log_b a} \lg n)$$

---

③ If  $f(n) = \Omega(n^{\log_b a + \epsilon})$  for some constant  $\epsilon > 0$

and if  $af(n/b) \leq cf(n)$  for some constant  $c < 1$

and all sufficiently large  $n$

$$T(n) = \Theta(f(n))$$

---

## Master Method

M.I.U.  $T(n) = 9T(n/3) + n$

- $a=9, b=3, f(n)=n$
- $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
- $f(n) = O(n^{\log_3 9 - \epsilon}) \quad \epsilon=1 \rightarrow \textcircled{1}$

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


---

M.I.U.  $T(n) = T(2n/3) + 1$

- $a=1, b=3/2, f(n)=1$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
- $f(n) = \Theta(n^{\log_b a}) = \Theta(1) \rightarrow \textcircled{2}$

$$T(n) = \Theta(\lg n)$$


---

M.I.U.  $T(n) = 3T(n/4) + nlgn$

- $a=3, b=4, f(n)=nlgn$
- $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
- $f(n) = \Omega(n^{\log_4 3 + \epsilon}) \quad \epsilon \approx 0.2 \rightarrow \textcircled{3}?$
- $af(n/b) = 3(n/4) \lg(n/4) \leq 3/4 nlgn = cf(n)$

$\rightarrow \textcircled{3}$

$$T(n) = \Theta(nlgn)$$


---

# Data Structures

- List
  - Tree
  - Table
  - Heaps
  - Disjoint Set
  - Graph.
- 

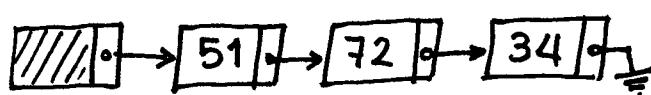
## Lists

Implementations :

- Contiguous



- linked



Q. V. Contiguous + Sorted (ascending)

Find : Binary search :  $O(\log n)$

Insert : ចំនួនចំណុះ :  $O(n)$

Delete : . . . :  $O(n)$

Find Min: ចំនួនការងារបែកស្ថិត :  $O(1)$

### ຕ.ບ. Contiguous + unordered

Find : Sequential Search :  $O(n)$

Insert : ចំណាំ :  $O(1)$

Delete : ដំឡើងលើក្រឡាត្រូវការ :  $O(1)$

FindMax : ចិត្តរីងប្រើបាយកែវយកចុង :  $\Theta(n)$

---

### ຕ.ບ. linked list + ordered

Find : Sequential search :  $O(n)$

Insert : ផ្ទៀងផ្ទាត់ ក្នុងលេខបន្ថែម  
pointers

Delete : ឬ — . :  $O(1)$

FindMax : ចិត្តរីងប្រើបាយកែវយកចុង :  $\Theta(n)$

---

### ន.ស.

#### Stack

#### Queue

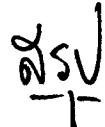
Push :  $O(1)$

Enqueue :  $O(1)$

Pop :  $O(1)$

Dequeue :  $O(1)$

---

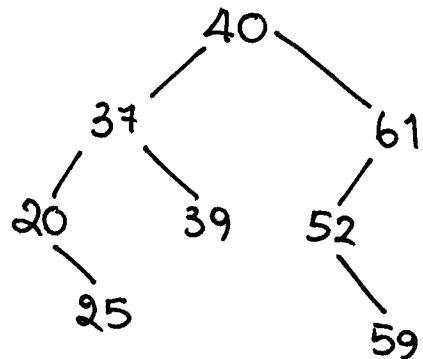


ទីនេះបាន operation ទៅ list នៅក្នុង  $O(n)$   
( Find, Insert, Delete )

---

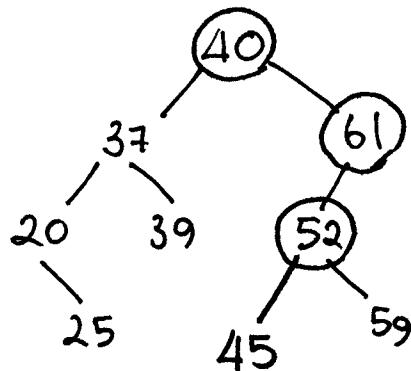
# Trees

## Binary Search Trees



- Find :  $O(h)$

- Insert : 69 45



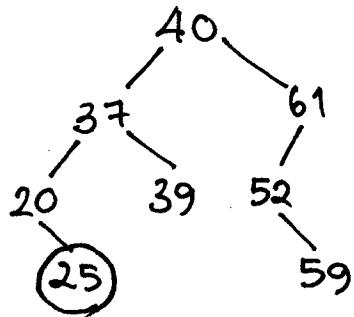
$O(h)$

- Delete

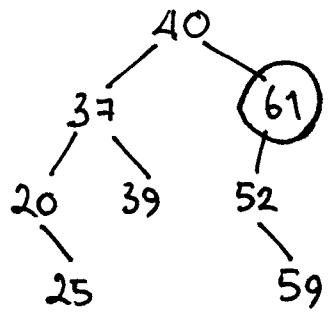
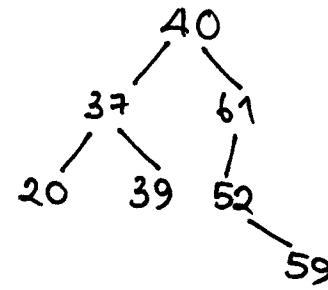
- លប់ឱ្យ
- លប់ node តូច 1 នូក
- លប់ node តូច 2 នូក

Delete

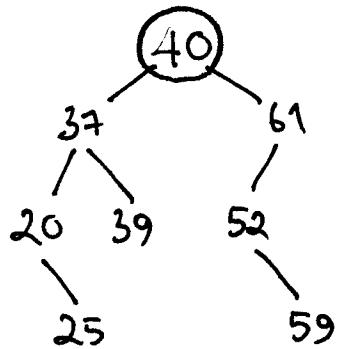
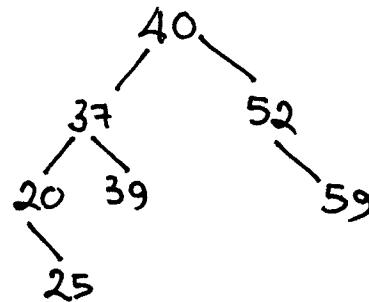
$O(h)$



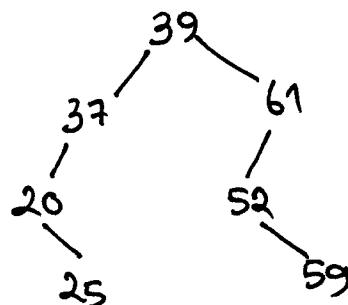
刪除 25



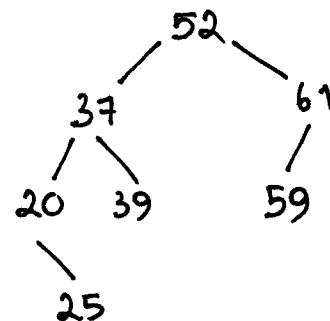
刪除 61



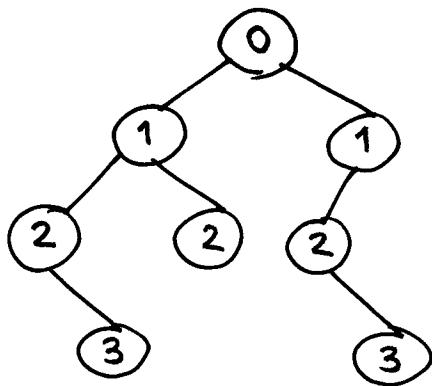
刪除 40



完成



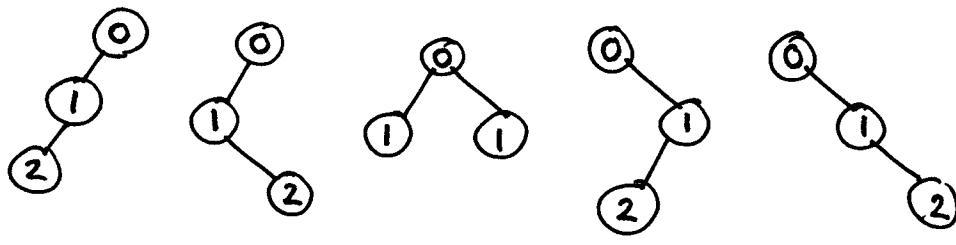
ទរាយត្រក់លីន node 9 នូវ BST



$$\begin{aligned}
 \text{ទរាយត្រក់លីនលើ node 8 នូវ សំណង់បន្ថែម} &= \frac{0+1+1+2+2+2+3+3}{8} \\
 &= \frac{14}{8} = 1\frac{6}{8}
 \end{aligned}$$


---

BST ត្រូវ 3 nodes



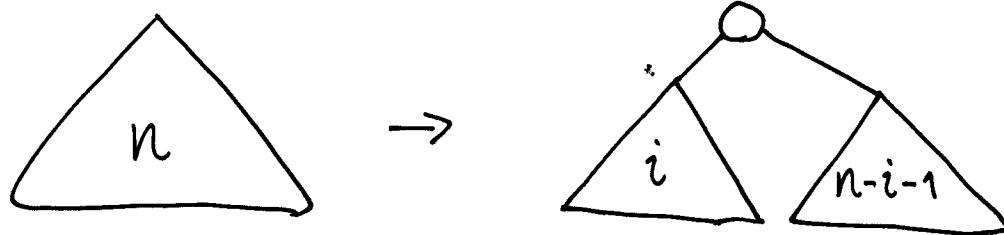
$$\frac{0+1+2}{3} \quad \frac{0+1+2}{3} \quad \frac{0+1+1}{3} \quad \frac{0+1+2}{3} \quad \frac{0+1+2}{3}$$

$$\begin{aligned}
 \text{ទរាយត្រក់លីនលើ node 9 នូវ BST ត្រូវ 3 nodes សាន្តិចាំ} &= \frac{1+1+\frac{2}{3}+1+1}{5} \\
 &= \frac{14}{15}
 \end{aligned}$$


---

## Expected Depth of node in BST of n nodes

Qū D(n) = ເກມາດວິທະນາຄານຂອງ node ອີ່ n nodes



$$D(n) = D(i) + D(n-i-1) + n-1$$

ການພົບໃນ ຂອງການ ດັວກເນັດ ຕະຫຼາມປົບປະຈຸບັນ ຈະ ປັດຕາ ແກ້ໄຂ ຂອງ ນັກ ຖໍ່ມີການ

$$\begin{aligned}
 D(n) &= \frac{1}{n} \sum_{i=0}^{n-1} [D(i) + D(n-i-1) + n-1] \\
 &= \left( \frac{1}{n} \sum_{i=0}^{n-1} D(i) \right) + \left( \frac{1}{n} \sum_{i=0}^{n-1} D(n-i-1) \right) + n-1 \\
 &= \left( \frac{2}{n} \sum_{i=0}^{n-1} D(i) \right) + n-1 \\
 &= O(n \log n)
 \end{aligned}$$

$$\therefore \text{Expected depth} = O(\log n)$$

# Hash Tables

List:

0	1	2	3	4	
7	11	5			3

Table :①

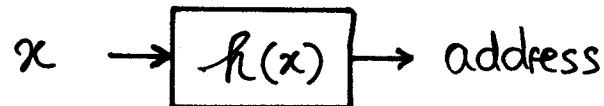
0	1	2	3	4	5	6	7	8	9	10	11
						✓		✓			✓

:②

0	1	2	3	4	5	6
			5	7		11

:③

0	1	2	3
7		11	5



$$\textcircled{1} \quad h(x) = x$$

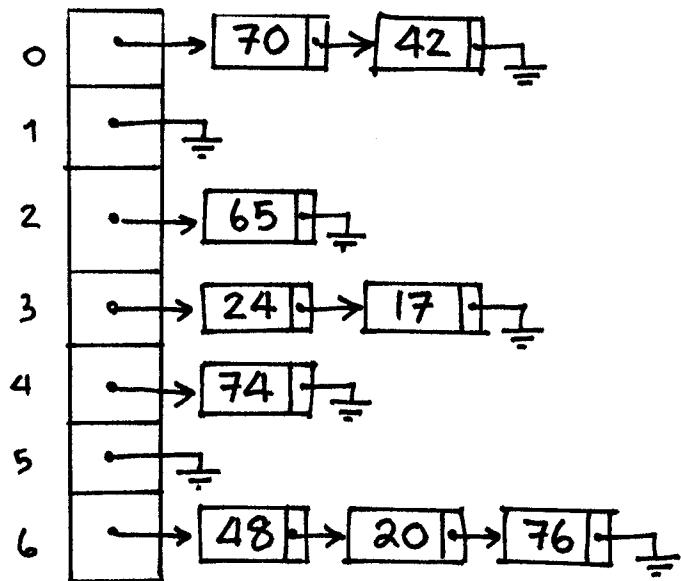
$$\textcircled{2} \quad h(x) = \lceil \frac{x}{2} \rceil$$

$$\textcircled{3} \quad h(x) = \lceil \frac{x}{2} \rceil \bmod 4$$

- hashing function
- Collision resolution

## Separate Chaining

Q. E.  $h(x) = x \bmod 7$



T : Table size

N : # data

$$\lambda = \frac{N}{T} \quad \text{load factor.}$$

ກົດ hash function ຕໍ່  $\rightarrow$  ຂອບພະນຈິກ: ການລັງ  
ເຄືອດຳວັດ ພອງກົດ

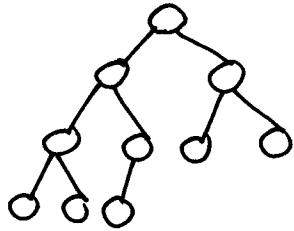
ການລັງພາກຕົວໄລ້: list = λ

Find, Delete  $\rightarrow O(\lambda)$

Insert  $\rightarrow O(1)$

# Heapsort

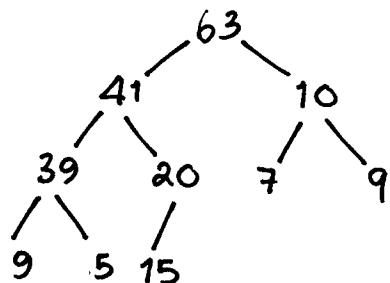
- Binary heap
  - Structure property



$n$  nodes

$$h = \lfloor \lg n \rfloor$$

- Heap order property

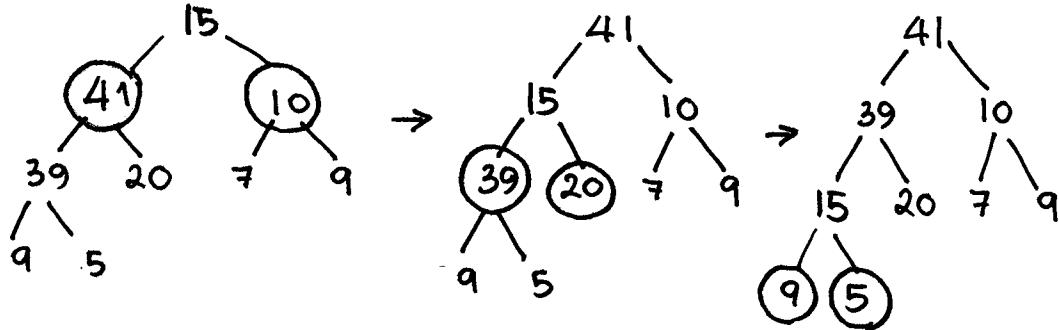
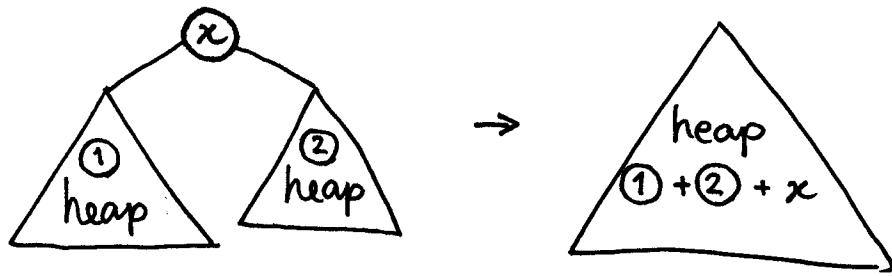


- Array implementation

10	1	2	3	4	5	6	7	8	9	10	11	12
	63	41	10	39	20	7	9	9	5	15		

- root の index 1
- 父ノードの node の index k の index 2k
- " → 727 " ————— k " ————— 2k+1
- " / 112 " " ————— k " —————  $\lfloor k/2 \rfloor$

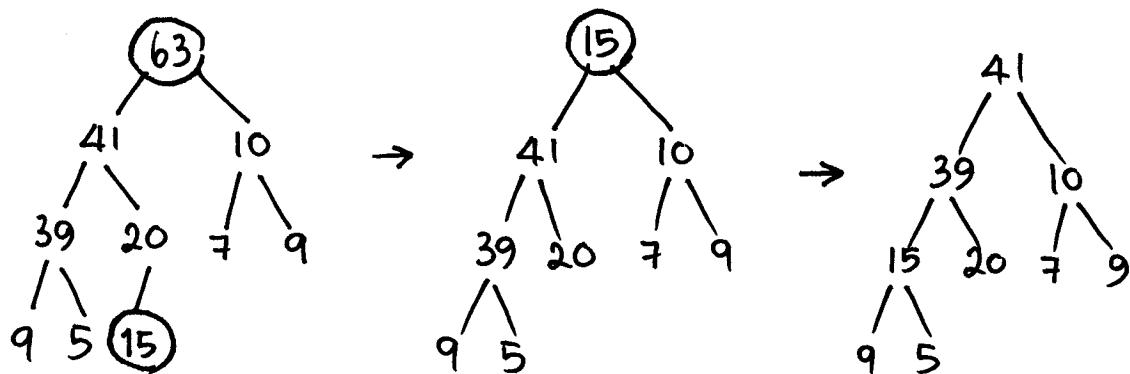
## Heapify



$O(h) \rightarrow O(\lg n)$

---

## Delete Max



$O(\lg n)$

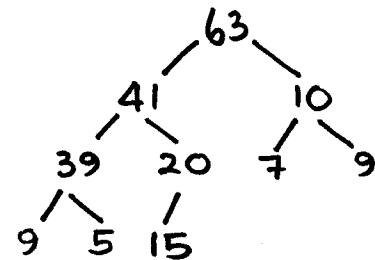
---

## Heapsort

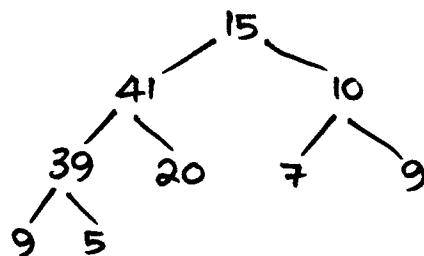
1	2	3	4	5	6	7	8	9	10	
10	9	15	63	5	20	7	10	9	39	41

"Build Heap"

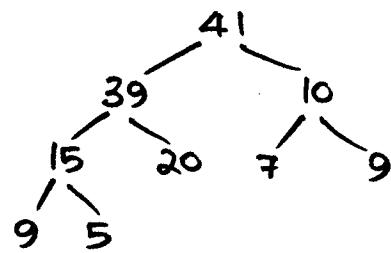
10	63	41	10	39	20	7	9	9	5	15
----	----	----	----	----	----	---	---	---	---	----



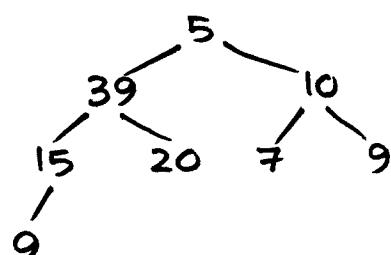
9	15	41	10	39	20	7	9	9	5	63
---	----	----	----	----	----	---	---	---	---	----



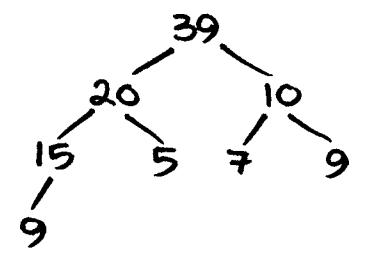
9	41	39	10	15	20	7	9	9	5	63
---	----	----	----	----	----	---	---	---	---	----



8	5	39	10	15	20	7	9	9	41	63
---	---	----	----	----	----	---	---	---	----	----

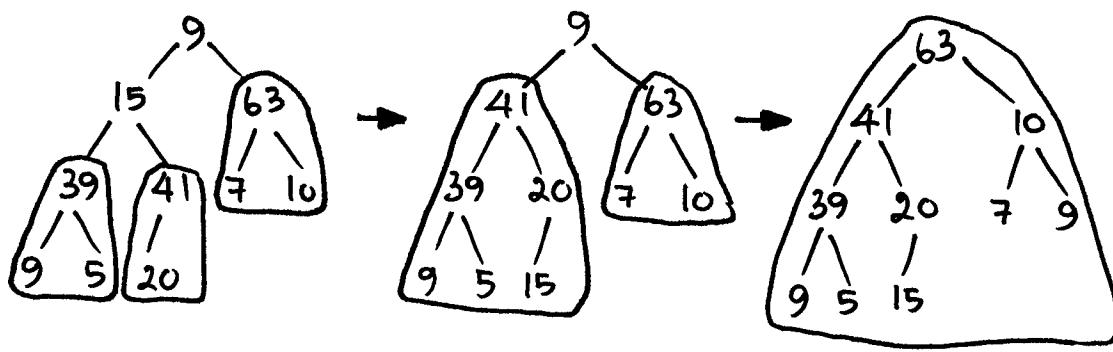
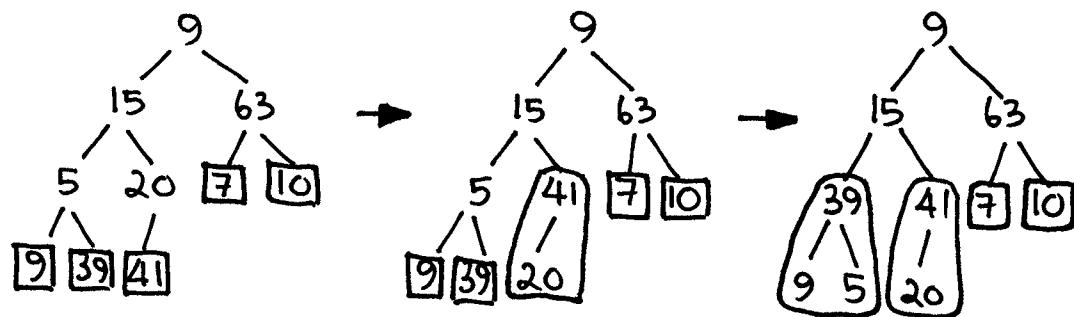
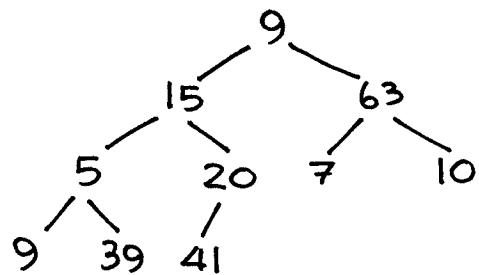


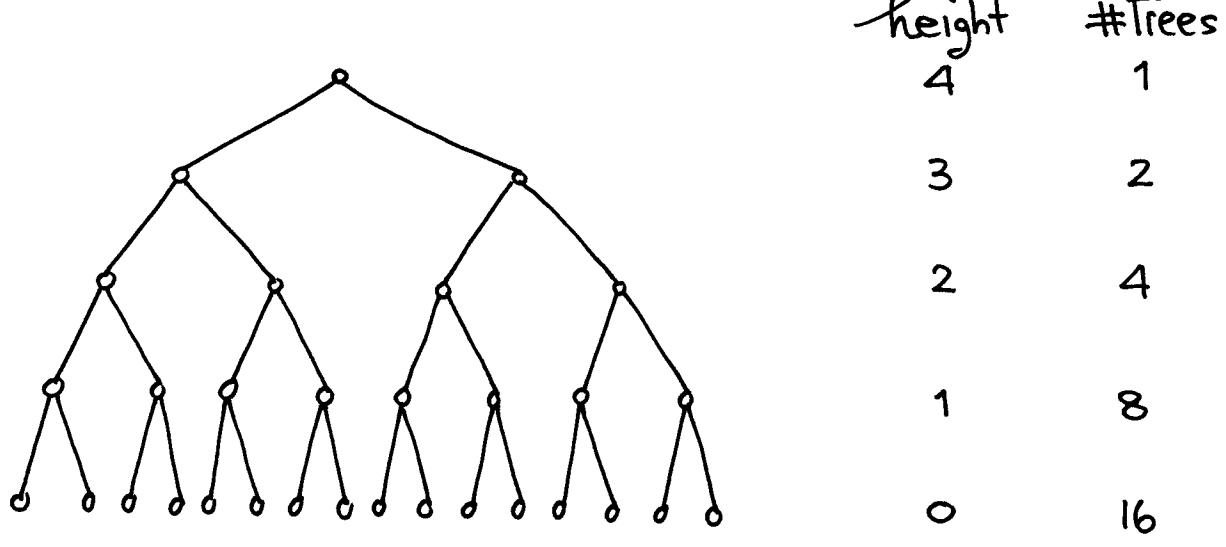
8	39	20	10	15	5	7	9	9	41	63
---	----	----	----	----	---	---	---	---	----	----



## Build Heap

1	2	3	4	5	6	7	8	9	10	
10	9	15	63	5	20	7	10	9	39	41





எனவே  $n$  nodes கிடைக்கும்  $\lfloor \lg n \rfloor$

$$\therefore \text{ஏனைய கீழ்க்கண்ட } \lfloor \lg n \rfloor \text{ காரணத்தினால்} \quad 1 \quad \text{என}$$

$$\dots \quad \lfloor \lg n \rfloor - 1 \quad \dots \quad 2 \quad "$$

$$\dots \quad \lfloor \lg n \rfloor - 2 \quad \dots \quad 4 \quad "$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$\dots \quad 2 \quad \dots \quad 2^{\lfloor \lg n \rfloor - 2}$$

$$\dots \quad 1 \quad \dots \quad 2^{\lfloor \lg n \rfloor - 1}$$

$$\dots \quad 0 \quad \dots \quad 2^{\lfloor \lg n \rfloor}$$

$$\begin{aligned} \therefore \text{தொகை} &\Rightarrow \sum_{h=0}^{\lfloor \lg n \rfloor} \left[ (2^{\lfloor \lg n \rfloor - h}) \cdot O(h) \right] \\ &= \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{n}{2^h} \cdot O(h) \\ &= O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) \\ &= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) = O(2n) = O(n) \end{aligned}$$

$$\sum_{h=0}^{\infty} h x^h = \frac{x}{(1-x)^2}$$

Heapsort( A[1..n] )

BuildHeap( A[1..n] ) ..... O(n)

for i < n down to 2 .....

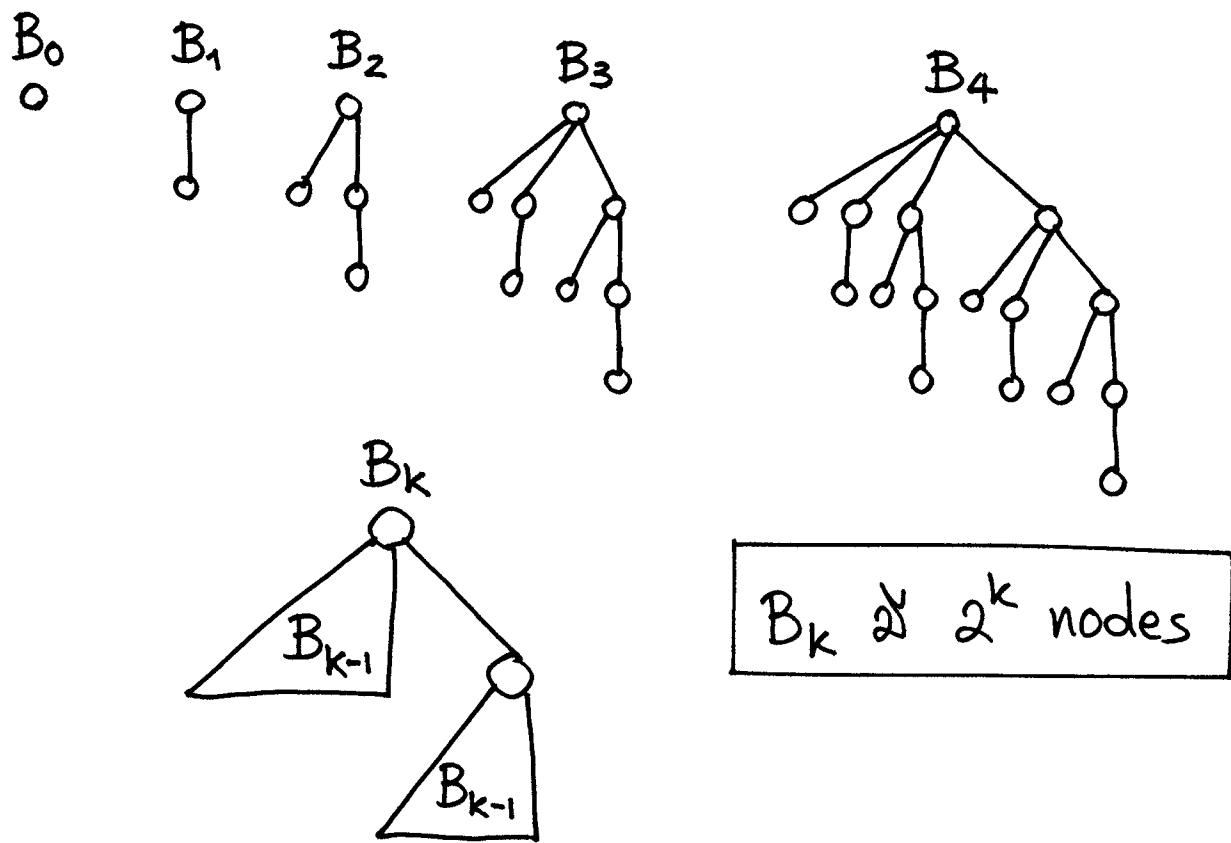
do Swap( A, 1, i ) ..... O(1) O(nlgn)

Heapify( A[1..n-1], 1 ) ..... O(lgn)

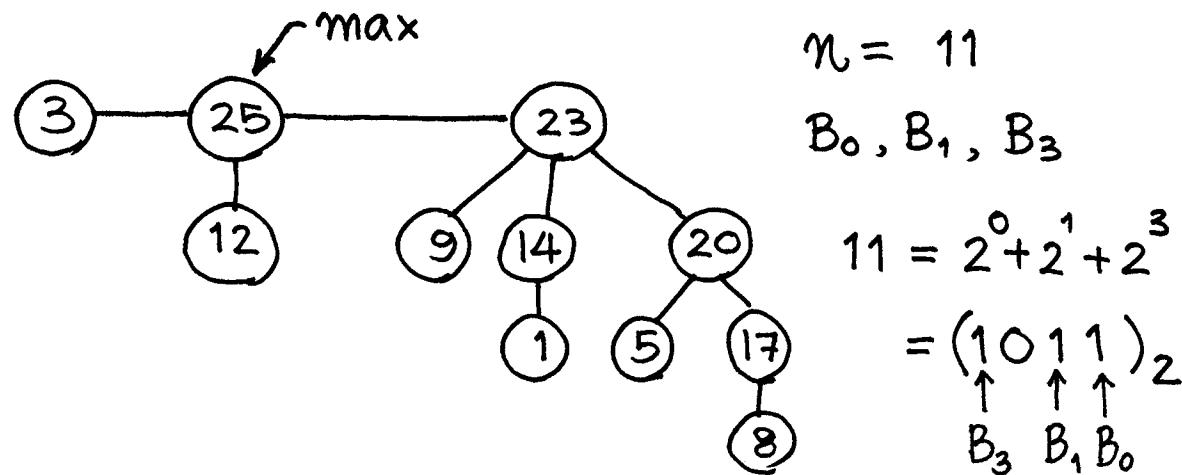
O(n log n)

# Binomial Heaps

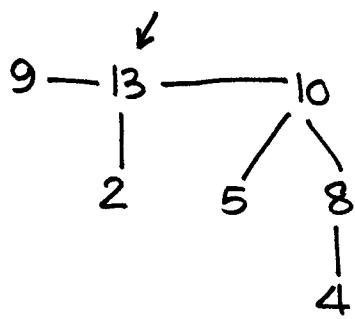
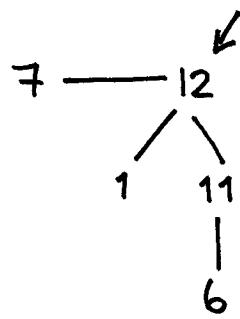
- a collection of binomial trees of different size with heap order property
- Binomial trees



- Binomial heaps



- merge

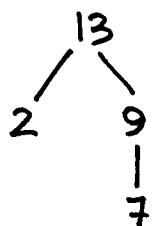


$$7 + 9 \rightarrow$$

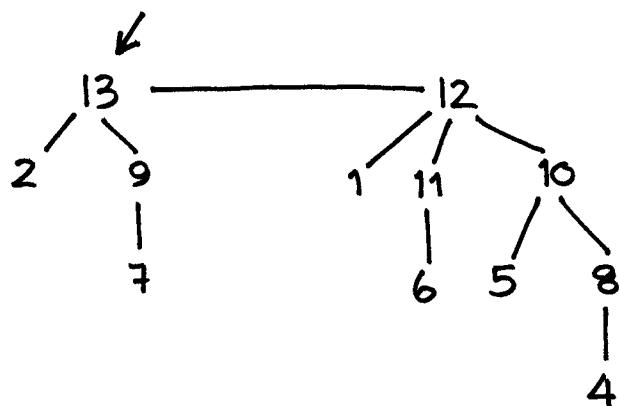
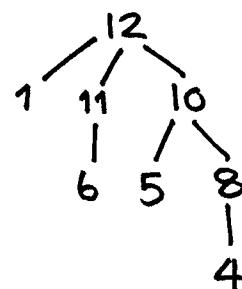


$$\begin{array}{r}
 B_3 \quad B_2 \quad B_1 \quad B_0 \\
 \hline
 0 & 1 & 0 & 1 \\
 0 & 1 & 1 & 1 \\
 \hline
 1 & 1 & 0 & 0
 \end{array}$$

$$9 + 13 \rightarrow$$



$$12 + 10 \rightarrow$$



- Binomial heap 2<sup>y</sup> n nodes  
2<sup>y</sup> binomial trees occur  $\lfloor \lg n \rfloor + 1$  times.
- 

E.g.  $n = 25$        $25_{10} = (11001)_2$   
 $= 2^4 + 2^3 + 2^0$

$\therefore$  2<sup>y</sup> 25 nodes heap:  $B_4, B_3, \text{ and } B_0$

$\therefore$  for  $25_{10}$  trees  $5$  bins  $= \lfloor \lg 25 \rfloor + 1$   
 $= 4 + 1$

---

- merge binomial heaps

- merging  $O(n)$
  - Find Max  $O(\log n) \rightarrow O(\log n)$
- 

- Find Max

- $q \approx n$  node  $\approx \max$   $\approx$

$$O(1)$$


---

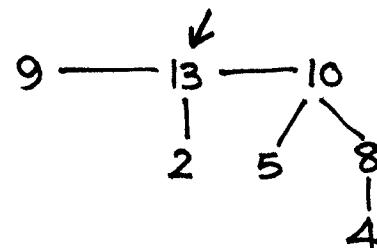
## • Insert

- เป็นต้นที่นิรดิษที่สุด เป็น binomial heap  
ตัวแรกๆ ก็จะเป็น  $B_0$  (ชีรดูง่ายกว่าต่อไป)

- merge heap ใหม่ๆ กับ heap 旧的ที่มีอยู่.

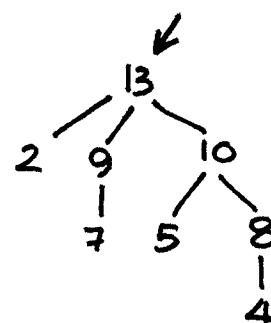
- 例.  $7 + 9 \rightarrow$

$$7 + 9 \rightarrow \begin{array}{c} 9 \\ | \\ 7 \end{array}$$



$$\begin{array}{c} 9 \\ | \\ 7 \end{array} + \begin{array}{c} 13 \\ | \\ 2 \end{array} \rightarrow \begin{array}{c} 13 \\ | \\ 2 \quad 9 \\ | \\ 7 \end{array}$$

$$\begin{array}{c} 13 \\ | \\ 2 \quad 9 \\ | \\ 7 \end{array} + \begin{array}{c} 10 \\ | \\ 5 \quad 8 \\ | \\ 4 \end{array} \rightarrow$$



- Insert 9 ห้าม 1 แล้วต่อไป 斐波那契数列 ลักษณะ 1

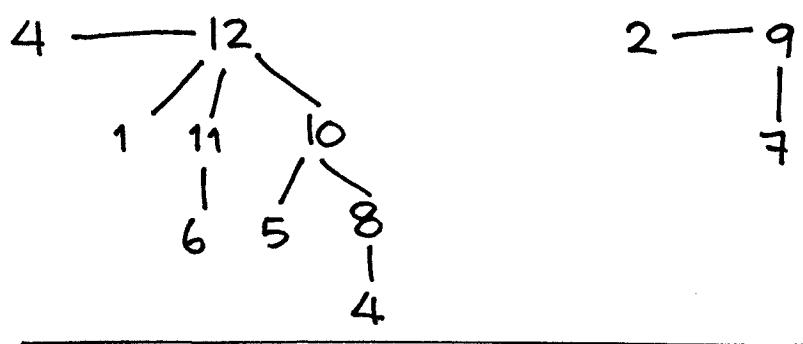
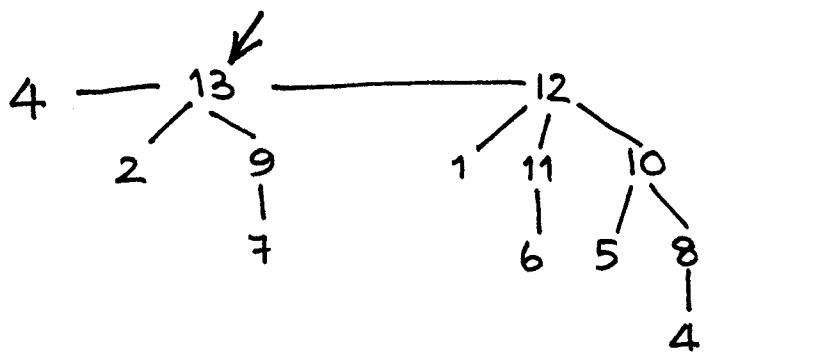
- แล้วต่อไป บวก 1  $\rightarrow O(\log n)$

$$111111 + 1 \rightarrow$$

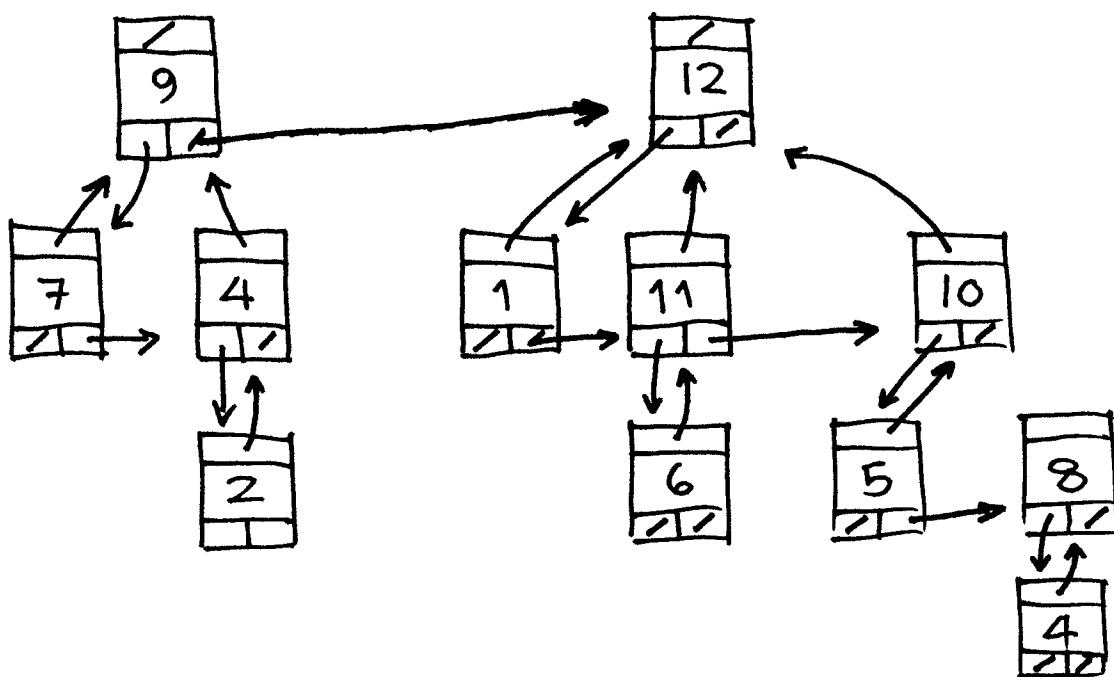
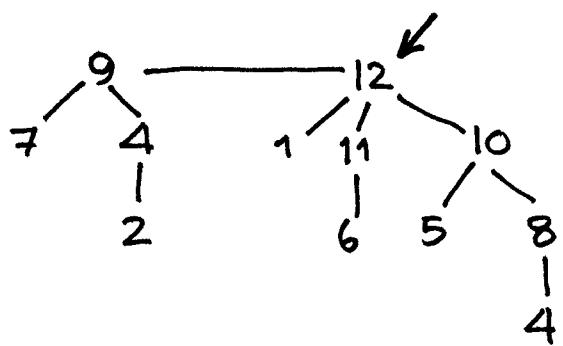
$$1111100 + 1 \rightarrow$$

- แล้วต่อไป increment  $Q_n$  binary counter  
amortized cost  $O(1)$

## • DeleteMax



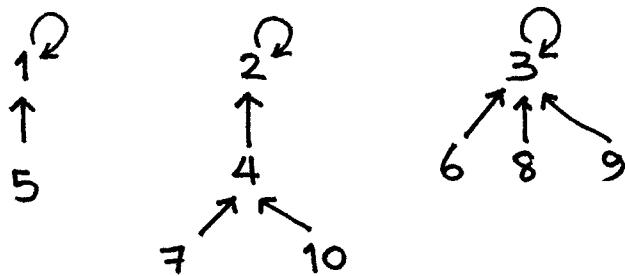
$O(\log n)$



# Disjoint Set

- ຂໍສ່ວນຂອງປົກຕົວໄວ້ 1 ລົ້ມ N
  - ຕ່າງການຮັບຮ່ວມຂອງແກ່ນີ້ປົກ sets
  - Operations
    - find : ແກ້ໄຂລາຍລະອຽດ set ກໍ່ຂອງຂອງປົກສົກສົກ
    - union : ຢັບ set 2 sets
- 

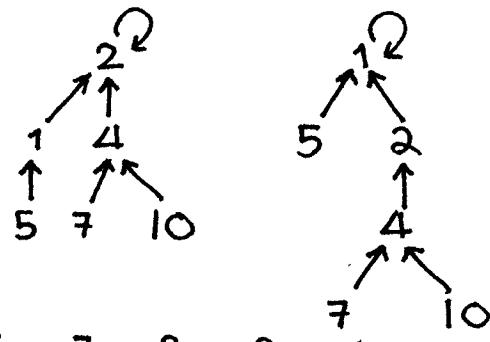
## Implementation



{1, 5}      {2, 4, 7, 10}      {3, 6, 8, 9}

S	1	2	3	4	5	6	7	8	9	10
	1	2	3	2	1	3	4	3	3	4

{1, 5}  $\cup$  {2, 4, 7, 10}



S	1	2	3	4	5	6	7	8	9	10
	-1	-2	-1	2	1	3	4	3	3	4

## o Union-by-height

- සිංහල තේව k nodes මෙහෙයුම්පාටුවන්  $\lfloor \lg k \rfloor$

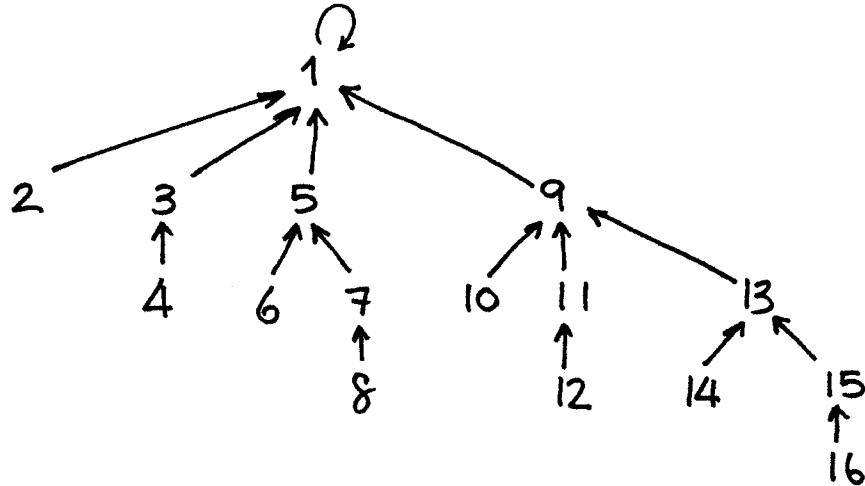
```
function find(x)
    if ( s[x] <= 0 )
        return x;
    else
        return ( find(s[x]) );
```

$O(\log n)$

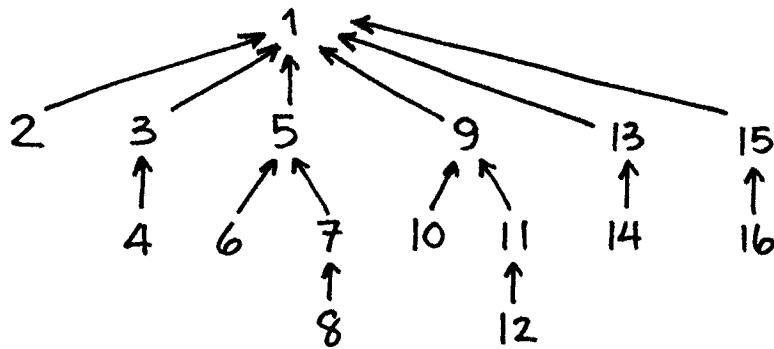
```
Union(a, b)
    if ( |s[a]| < |s[b]| )
        s[a] = b;
    else {
        if ( s[a] == s[b] )
            s[a] -- ;
        s[b] = a;
    }
```

$O(1)$

## Path Compression



Find (15)



```
function find( x )  
    if ( S[x] <= 0 )  
        return x;  
    else  
        return ( S[x]=find( S[x] ));
```

## Union-by-Rank & Path Compression

- Any sequence of  $m = \Omega(n)$  operations (Union/Find) takes a total of  $\underline{\mathcal{O}(m \log^* n)}$  running time

- $\log_2^* 2 = 1$ ,  $\log_2^* 2^2 = 2$ ,  $\log_2^* 2^{2^2} = 3$

$$\log_2^* 2^{2^k} = k$$

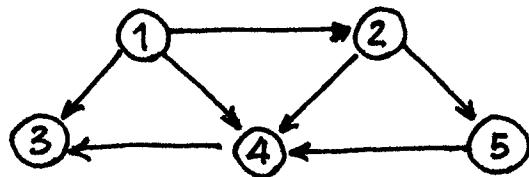
$$\log_2^* 2^{2^{2^{2^2}}} = 5 \quad ; \quad 2^{2^{2^{2^2}}} \text{ is a 20,000 digit number!}$$

$\mathcal{O}(m \log^* n)$  ~~is at most~~  $\mathcal{O}(m)$

$$\because \log^* n \leq 5$$

$\therefore$  Union / find operation takes  $\mathcal{O}(1)$  time!

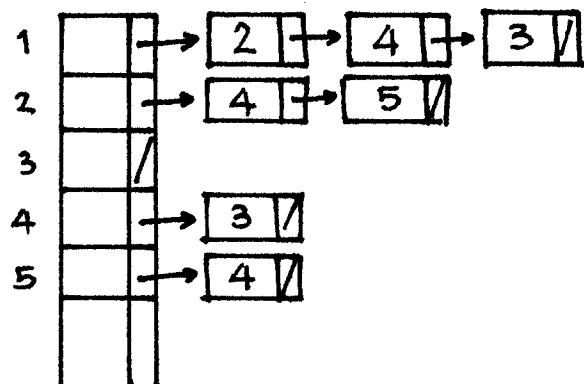
# Graphs



## Adjacency Matrix

	1	2	3	4	5
1	0	1	1	1	0
2	0	0	0	1	1
3	0	0	0	0	0
4	0	0	1	0	0
5	0	0	0	1	0

## Adjacency List

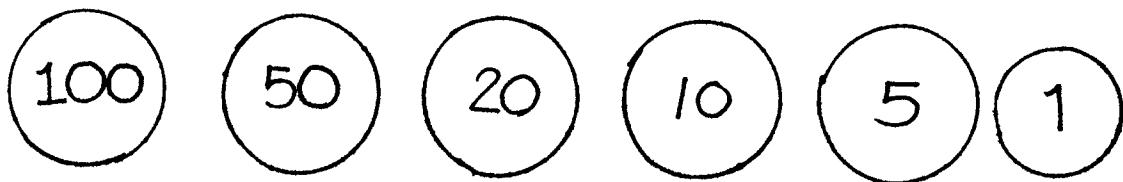


## Greedy Algorithms

- simple
- easy to invent
- easy to implement
- efficient
- typically used to solve optimization problems

## Making Changes

- ទទួលយករាយបានចាប់ពីតម្លៃធំបន្ថែមដល់ការបង្ហាញ។



$$247 = 100 \times 2 + 20 \times 2 + 5 \times 1 + 2 \times 1$$

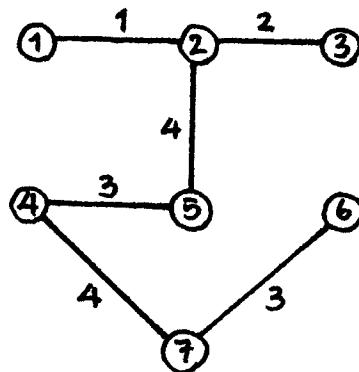
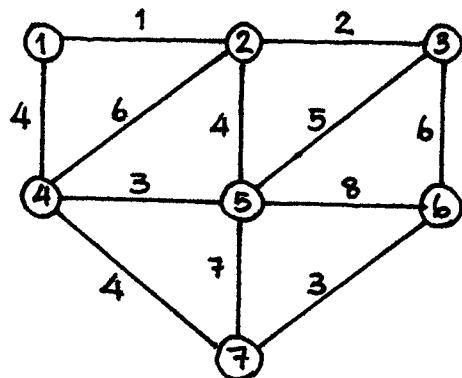
បានចាប់ពីតម្លៃធំបន្ថែម

```
function Greedy ( C : set ) : set
    S  $\leftarrow \emptyset$ 
    while  $C \neq \emptyset$  and not solution( S ) do
         $x \leftarrow \text{Select}(C)$ 
         $C \leftarrow C - \{x\}$ 
        if feasible(  $S \cup \{x\}$  ) then  $S \leftarrow S \cup \{x\}$ 
    if solution( S ) then return S
    else return "no solution"
```

C - candidate set

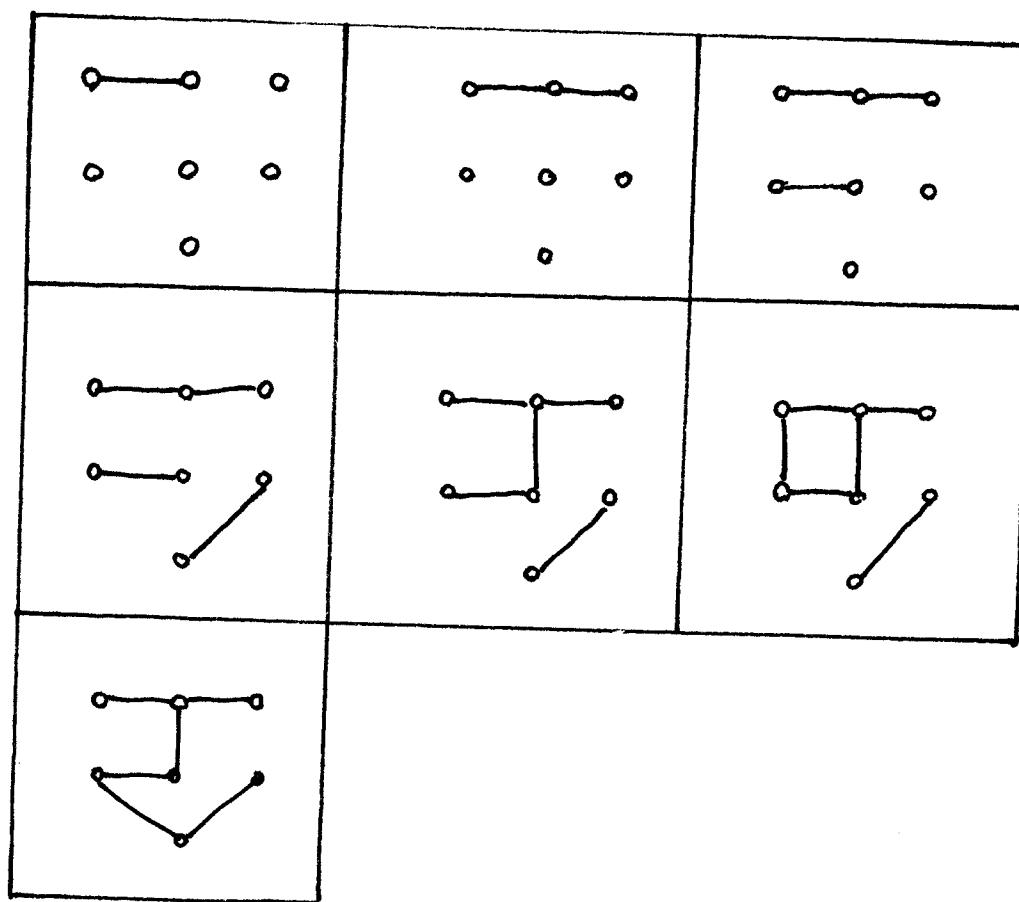
S - Solution set

## Minimum Spanning Trees



## Kruskal's algorithm

- เลือก edge สัมเมซู
  - รวม edge ที่ ถ้าไม่เกิดวงจรอ
- ←  
เลือกไป  
 $n-1$  edges



Kruskal( $G, \omega$ )

$T \leftarrow \emptyset$

for each vertex  $v \in V(G)$  do

    MAKE-SET( $v$ )

Sort the edges of  $E(G)$  by nondecreasing weight  $\omega$

for each edge  $(u, v) \in E(G)$

    in order by non decreasing weight do

$x \leftarrow \text{find}(u)$

$y \leftarrow \text{find}(v)$

        if  $x \neq y$  then

            UNION( $x, y$ )

$T \leftarrow T \cup \{(u, v)\}$

return  $T$

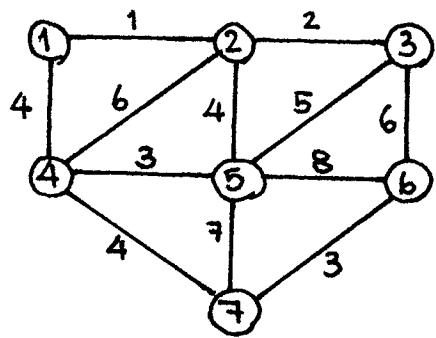
Sort :  $O(|E| \log |E|) \rightarrow O(|E| \log |V|)$

$$\because |V|-1 \leq |E| \leq |V|(|V|-1)/2.$$

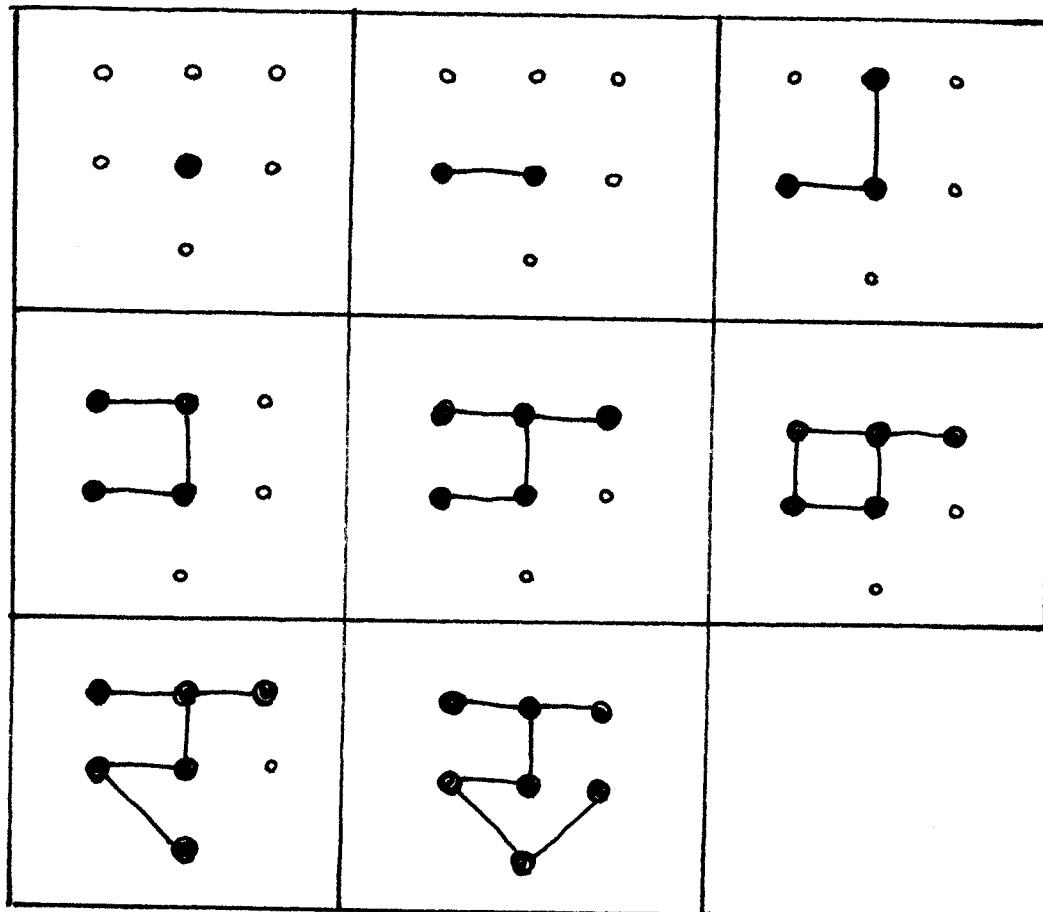
At most  $2|E|$  find operations }  $O(|E| \log^* |V|)$   
 $|V|-1$  Union    ... }

$\therefore O(|E| \log |V|)$

## Prim's algorithm



- เลือก 1 စု
- เลือก edge သို့နှင့်ကို စောက်ပါ၍ ၅။ graph
- သော် edge နှင့် ပါများမြှိုက်လောက်

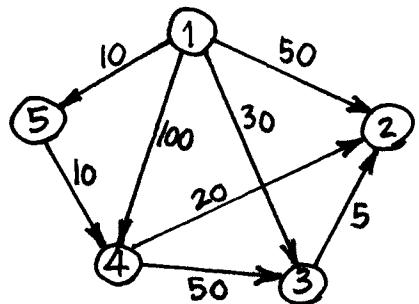


Prim(  $G, \omega$  )

```
 $Q \leftarrow V(G)$  /* priority queue */  
for each  $u \in Q$  do  $\text{key}[u] \leftarrow \infty$   
 $r \leftarrow$  an arbitrary node in  $V(G)$   
 $\text{key}[r] \leftarrow 0$   
 $\pi[r] \leftarrow \text{NIL}$   
while  $Q \neq \emptyset$  do  
   $u \leftarrow \text{DeleteMin}(Q)$   
  for each  $v \in \text{Adj}[u]$  do  
    if  $v \in Q$  and  $\omega(u, v) < \text{key}[v]$   
      then  $\pi[v] \leftarrow u$   
           $\text{key}[v] \leftarrow \omega(u, v)$ 
```

$Q$  : binary heap - Build heap  $O(|V|)$   
- DeleteMin  $O(|V| \log |V|)$   
- DecreaseKey  $O(|E| \log |V|)$   
 $\therefore O(|E| \log |V|)$

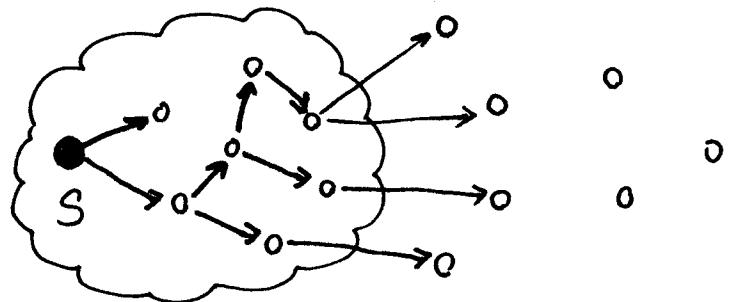
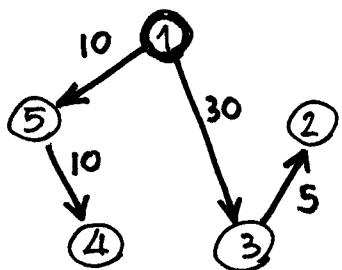
## Shortest Paths (Single Source)



S:U:100

- Այս անում այս source node

Վյէյ node Ծնյ գիշտ: Ա: Հնդի.



## Dijkstra's Algorithm

Dijkstra(  $G, w, s$  )

for each vertex  $v \in V(G)$  do

$d[v] \leftarrow \infty$

$\pi[v] \leftarrow \text{NIL}$

$d[s] \leftarrow 0;$

$ST \leftarrow \emptyset$

$Q \leftarrow V(G)$

while  $Q \neq \emptyset$  do

$u \leftarrow \text{DeleteMin}(Q)$

$ST \leftarrow ST \cup \{u\}$

for each vertex  $v \in \text{Adj}(u)$  do

if  $d[v] > d[u] + w(u, v)$  then

$d[v] \leftarrow d[u] + w(u, v)$

$\pi[v] \leftarrow u$

## Huffman codes

	a	b	c	d	e	f
frequency ( $\times 10^3$ )	45	13	12	16	9	5
① fix-length	000	001	010	011	100	101
② variable-length	0	101	100	111	1101	1100

① If we use 300,000 bits

② ----- 224,000 "

### Prefix code :

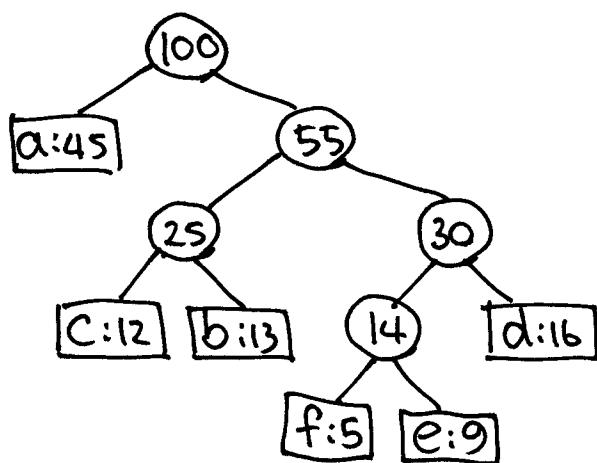
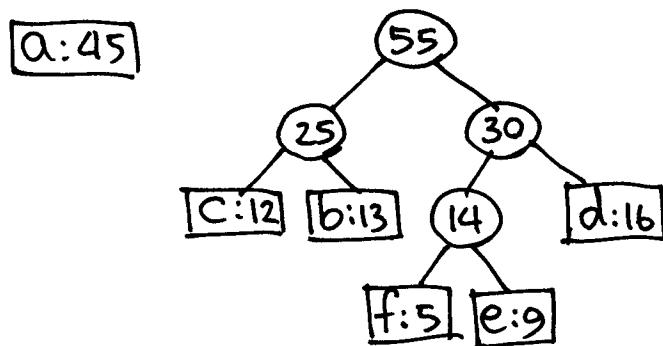
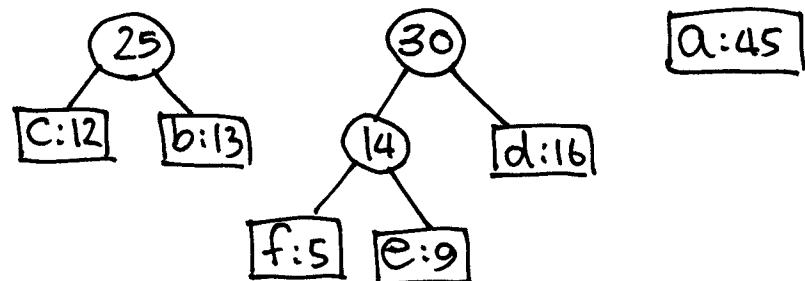
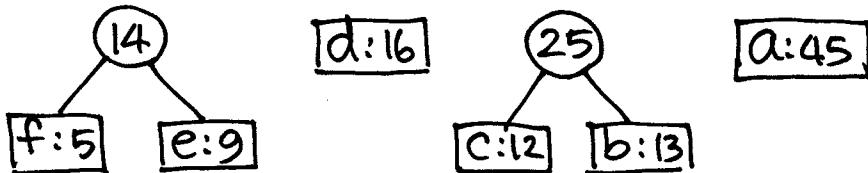
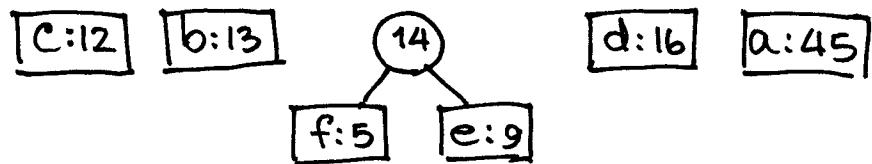
no codeword is a prefix of  
some other codeword

Huffman code  $\Rightarrow$  optimal prefix code

---

f:5 e:9 c:12 b:13 d:16 a:45

---



Huffman( c )

$n \leftarrow |c|$

$Q \leftarrow c$

for  $i \leftarrow 1$  to  $n-1$  do

$z \leftarrow \text{Allocate\_Node}()$

$x \leftarrow \text{left}[z] \leftarrow \text{DeleteMin}(Q)$

$y \leftarrow \text{right}[z] \leftarrow \text{DeleteMin}(Q)$

$f[z] \leftarrow f[x] + f[y]$

$\text{Insert}(Q, z)$

return  $\text{DeleteMin}(Q)$

Build Heap :  $O(n)$

for loop :  $O(n \log n)$

$\therefore O(n \log n)$

## Knapsack

- $n$  objects & a knapsack
- weight  $w_1, w_2, \dots, w_n$
- value  $v_1, v_2, \dots, v_n$
- the knapsack can carry weight  $\leq W$

$$\text{maximize } \sum_{i=1}^n x_i v_i$$

subject to

$$\sum_{i=1}^n x_i w_i \leq W$$

$$\begin{aligned} 0 \leq x_i \leq 1 \\ v_i > 0 \\ w_i > 0 \end{aligned} \quad \left. \right\} 1 \leq i \leq n$$

- Solution : a vector  $(x_1, x_2, \dots, x_n)$

M.E.

$$n = 5, W = 100$$

i	1	2	3	4	5
$w_i$	10	20	30	40	50
$v_i$	20	30	66	40	60
$v_i/w_i$	2.0	1.5	2.2	1.0	1.2

Select	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Total value
$\max v_i$	0	0	1	0.5	1	146
$\min w_i$	1	1	1	1	0	156
$\max v_i/w_i$	1	1	1	0	0.8	164

~~Knapsack ( $w[1..n]$ ,  $v[1..n]$ ,  $W$ )~~

~~for  $i \leftarrow 1$  to  $n$  do  $x[i] \leftarrow 0$~~

~~for  $i \leftarrow 1$  to  $n$  do  $vw[i] \leftarrow v[i]/w[i]$~~

~~Sort  $vw$  in nondecreasing order~~

~~wght  $\leftarrow 0$ ;  $i \leftarrow 1$~~

~~while  $wght < W$  do~~

~~if  $wght + w[i] \leq W$  then  $x[i] \leftarrow 1$~~

~~wght  $\leftarrow w[i]$~~

~~else  $x[i] \leftarrow \frac{(W - wght)}{w[i]}$~~

~~wght  $\leftarrow W$~~

Knapsack (  $w[1..n]$ ,  $v[1..n]$ ,  $W$  )

for  $i \leftarrow 1$  to  $n$  do

$x[i] \leftarrow 0$

$p[i] \leftarrow v[i]/w[i]$

sort  $p$  in nondecreasing order

$wght \leftarrow 0$

$i \leftarrow 1$

    while ( $wght < W$ ) do

        if  $wght + w[i] \leq W$  then

$x[i] \leftarrow 1$

$wght += w[i]$

        else

$x[i] \leftarrow (W - wght) / w[i]$

$wght = W$

$i \leftarrow i + 1$

    return  $x$

$O(n \log n)$

## Scheduling

- Jobs  $j_1, j_2, \dots, j_n$
- Known running times  $t_1, t_2, \dots, t_n$
- A single processor
- Find the best way to schedule these jobs in order to minimize the average completion time

M.V.  $n = 3, t_1 = 5, t_2 = 10, t_3 = 3$

Order	T
1 2 3 :	$5 + (5+10) + (5+10+3) = 38$
1 3 2 :	$5 + (5+3) + (5+3+10) = 31$
:	:
3 1 2 :	$3 + (3+5) + (3+5+10) = 29*$
3 2 1 :	$3 + (3+10) + (3+10+5) = 34$

Greedy choice : "shortest job first"

## Greedy Algorithms

- Greedy-choice property

a globally optimal solution can be arrived at by making a locally optimal (greedy) choice

---

- Optimal substructure

Optimal solution to the problem contains within it optimal solutions to subproblems .

---

## Divide-and-Conquer

- បើយំ instance នេះជាបញ្ហា  
តាម subinstances ចំណាំយ៉ាង  
នៅក្នុងហាន់
  - ខាតំសមតុល្យ នៃ subinstances
  - គួរតាមតុល្យទៀតកំណត់  
តាមពាណិជ្ជកម្ម instance ទី២.
- 

## Multiplying large integers

$$\begin{aligned} \textcircled{1} \quad 0981 \times 1234 &= (10^2 \times 09 + 81) \times (10^2 \times 12 + 34) \\ &= 10^4 \times 09 \times 12 + 10^2 \times (09 \times 34 + 81 \times 12) + 81 \times 34 \\ t(n) &= 4t(n/2) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$


---

$$\begin{aligned} \textcircled{2} \quad p &= 09 \times 12, \quad q = 81 \times 34 \\ r &= (09 + 81) \times (12 + 34) \end{aligned}$$

$$0981 \times 1234 = 10^4 p + 10^2(r - p - q) + q$$

$$\begin{aligned} t(n) &= 3t(n/2) + \Theta(n) \\ &= \Theta(n^{\lg 3}) = \Theta(n^{1.585}) \end{aligned}$$

# Merge Sort

MERGE-SORT( A, p, r )

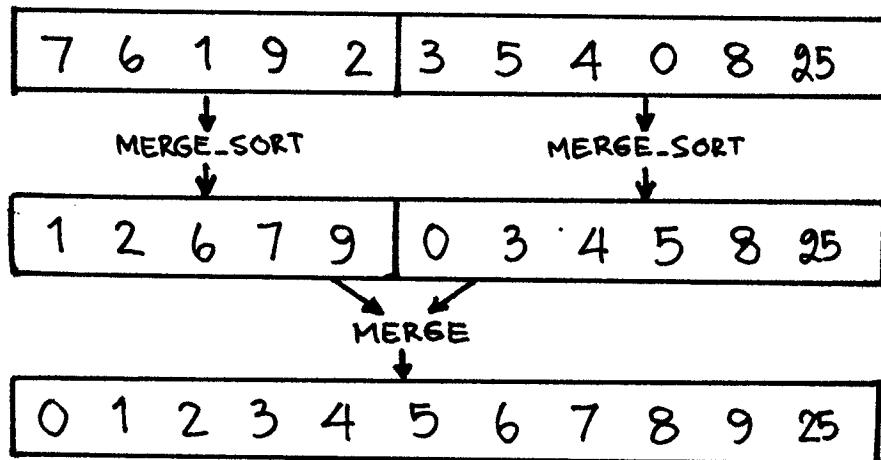
if  $p < r$  then

$$q \leftarrow \lfloor (p+r)/2 \rfloor$$

MERGE-SORT( A, p, q )

MERGE-SORT( A, q+1, r )

MERGE( A, p, q, r )



$$t(n) = 2t(n/2) + \Theta(n)$$

$$= \Theta(n \log n)$$

# Quick Sort

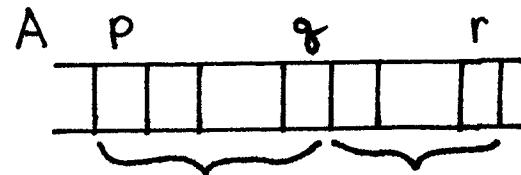
```
QUICK-SORT( A, p, r )
```

```
if p < r then
```

```
    q ← PARTITION( A, p, r )
```

```
    QUICK-SORT( A, p, q - 1 )
```

```
    QUICK-SORT( A, q + 1, r )
```



```
PARTITION( A, p, r )
```

```
x ← A[p]
```

```
i ← p - 1
```

```
j ← r + 1
```

```
while TRUE do
```

```
    repeat
```

```
        j ← j - 1
```

```
    until A[j] ≤ x
```

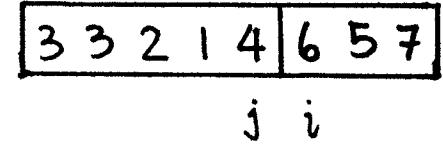
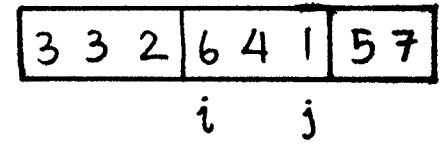
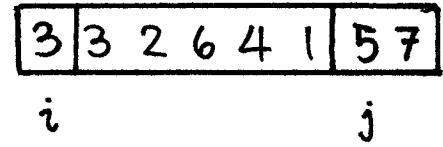
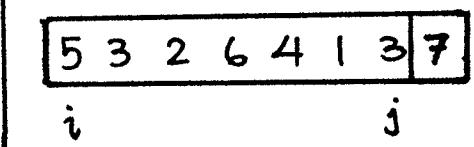
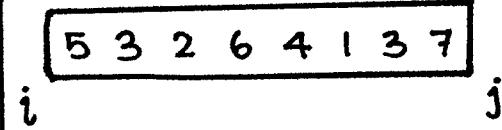
```
    repeat
```

```
        i ← i + 1
```

```
    until A[i] ≥ x
```

```
    if i < j then A[i] ↔ A[j]
```

```
    else return j
```



## Performance of Quicksort

$$t(n) = t(q) + t(n-q) + \Theta(n)$$

Worst-case :

$$\begin{aligned} t(n) &= t(1) + t(n-1) + \Theta(n) \\ &= t(n-1) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

Best-case :

$$\begin{aligned} t(n) &= t(n/2) + t(n-n/2) + \Theta(n) \\ &= 2t(n/2) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

Average-case :

assumption: all permutations of the input numbers are equally likely.

$$\begin{aligned} t(n) &= \frac{1}{n} \left[ t(1) + t(n-1) + \sum_{q=1}^{n-1} (t(q) + t(n-q)) \right] + \Theta(n) \\ &= \frac{1}{n} \sum_{q=1}^{n-1} (t(q) + t(n-q)) + \Theta(n) \\ &= O(n \log n) \end{aligned}$$

## Randomized Quick sort

```
RAND_PARTITION ( A, p, r )  
    i ← RANDOM( p, r )  
    A[p] ↔ A[i]  
    return PARTITION( A, p, r )
```

## Performance of QuickSort

### Worst-case partitioning

$$\begin{aligned}
 t(n) &= t(n-1) + \Theta(n) \\
 &= \sum_{k=1}^n \Theta(k) \\
 &= \Theta\left(\sum_{k=1}^n k\right) \\
 &= \Theta(n^2)
 \end{aligned}$$

### Best-case partitioning

$$\begin{aligned}
 t(n) &= 2t(n/2) + \Theta(n) \\
 &= \Theta(n \log n)
 \end{aligned}$$

### Average-case

assumption: all permutations of the input numbers are equally likely.

$$\begin{aligned}
 t(n) &= t(q) + t(n-q) + \Theta(n) \\
 &= \frac{1}{n} \left( T(1) + T(n-1) + \sum_{q=1}^{n-1} (t(q) + t(n-q)) \right) + \Theta(n) \\
 &= \frac{1}{n} \sum_{q=1}^{n-1} (t(q) + t(n-q)) + \Theta(n) \\
 &= \frac{2}{n} \sum_{k=1}^{n-1} t(k) + \Theta(n)
 \end{aligned}$$

Solving the recurrence using substitution method .

$$\text{base case} \rightarrow t(n) \leq \alpha n \lg n + b \rightarrow t(1) \leq \alpha n \lg n + b$$

$$\begin{aligned} t(n) &= \frac{2}{n} \sum_{k=1}^{n-1} t(k) + \Theta(n) \\ &\leq \frac{2}{n} \sum_{k=1}^{n-1} (\alpha k \lg k + b) + \Theta(n) \\ &= \frac{2\alpha}{n} \sum_{k=1}^{n-1} k \lg k + \frac{2b}{n}(n-1) + \Theta(n) \end{aligned}$$

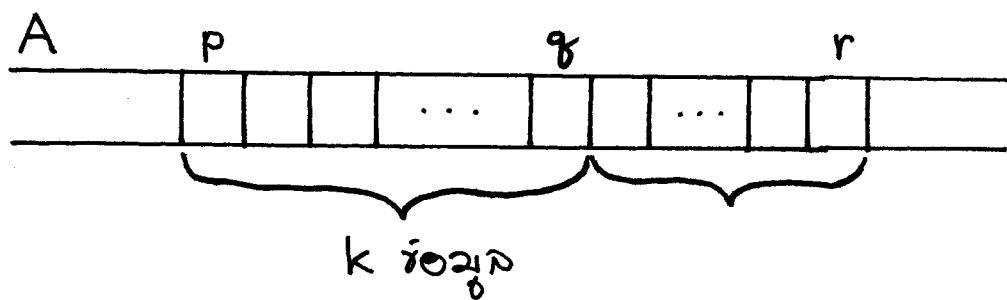
$$\therefore \sum_{k=1}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$$

$$\begin{aligned} t(n) &\leq \frac{2\alpha}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \frac{2b}{n}(n-1) + \Theta(n) \\ &\leq \alpha n \lg n - \frac{\alpha}{4} n + 2b + \Theta(n) \\ &= \alpha n \lg n + b + \left( \Theta(n) + b - \frac{\alpha}{4} n \right) \\ &\leq \alpha n \lg n + b \\ &= O(n \lg n) \end{aligned}$$

From a  $\frac{\alpha}{4} n$  into  $\Theta(n)$

## Selection ( find the $i^{\text{th}}$ smallest element )

```
SELECT( A, p, r, i )
    if p = r then return A[p]
    q ← PARTITION( A, p, r )
    k ← q - p + 1
    if i ≤ k then return SELECT( A, p, q, i )
    else return SELECT( A, q+1, r, i-k )
```



using RAND-PARTITION → expected linear time

$$\begin{aligned} t(n) &= \frac{1}{n} \left[ t(\max(1, n-1)) + \sum_{k=1}^{n-1} t(\max(k, n-k)) \right] + O(n) \\ &\leq \frac{1}{n} \left[ t(n-1) + 2 \sum_{k=\lceil n/2 \rceil}^{n-1} t(k) \right] + O(n) \\ &= O(n) \end{aligned}$$

## ଟାର୍କିଂସ worst-case linear time

- guarantee a good partition
- median of median of five

ସେସି

x	x	x	x	x	x	x
x	x	x	x	x	x	x
⊗	⊗	⊗	⊗	⊗	⊗	⊗
x	x	x	x	x	x	x
x	x	x	x	x	x	x

$n=35$  • ପରେମନଗ୍ରୁହୀତିରେ 5 ଟଙ୍କା  
ଏହି ନାହିଁ.

- ଏହି median ରୂପାଳୋଧି ଗ୍ରୁହୀତି ଦିଲ୍ଲିଯାଇଲା
- ଏହି median ରୂପାଳୋଧି ଗ୍ରୁହୀତି ଦିଲ୍ଲିଯାଇଲା



$O(n)$

$O(n)$

$t(n/5)$

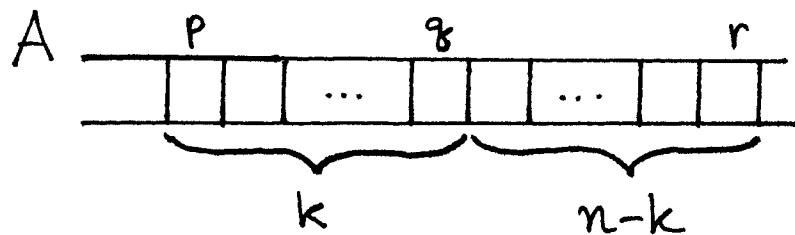
• କେବଳ  $\lfloor n/5 \rfloor$  ଗ୍ରୁହୀତି

• କେବଳ  $\frac{3}{2} \lfloor n/5 \rfloor$  ଶତାବ୍ଦୀ କେବଳ ଅନେକାଂଶରେ ପରେମନଗ୍ରୁହୀତି ଦିଲ୍ଲିଯାଇଲା

$$\bullet \because \left\lfloor \frac{n}{5} \right\rfloor \geq \frac{(n-4)}{5}, \therefore \frac{3n-12}{10}$$

$$\bullet \text{କେବଳ } n - \left( \frac{3n-12}{10} \right) = \frac{7n+12}{10} \text{ ଶତାବ୍ଦୀ}$$

କେବଳ ୨୦୦ଟି ଶତାବ୍ଦୀ ଦିଲ୍ଲିଯାଇଲା



$$t(n) = t(\max(k, n-k)) + O(n)$$

Since  $\max(k, n-k) \leq \frac{7n+12}{10}$

$$\begin{aligned} t(n) &\leq t\left(\frac{7n+12}{10}\right) + t(\text{unmm5}) + O(n) \\ &\leq t\left(\frac{7n+12}{10}\right) + t\left(\lfloor n/5 \rfloor\right) + O(n) \\ &= O(n) \end{aligned}$$

If  $\sum_{i=1}^k \alpha_i < 1$ , then the solution of

$$t(n) = \sum_{i=1}^k t(\alpha_i n) + O(n)$$

is  $O(n)$

Ex  $t(n) = t(0.1n) + t(0.3n) + t(0.59n) + O(n)$

Sol  $\frac{7n+12}{10} + \frac{n}{5} = \frac{7n+12+2n}{10} = 0.9n + 1.2$

## Matrix Multiplication

Given A and B are  $n \times n$  matrices

$$C = A \times B$$

$$C_{ij} = \sum_{k=1}^n A_{ik} B_{kj} \leftarrow \Theta(n)$$

$$\therefore C_{ij} \in \Omega(n^2)$$

$\therefore$  matrix multiplication  $\rightarrow \Theta(n^3)$

---

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$C_{11} = a_{11}b_{11} + a_{12}b_{21}$$

$$C_{12} = a_{11}b_{12} + a_{12}b_{22}$$

$$C_{21} = a_{21}b_{11} + a_{22}b_{21}$$

$$C_{22} = a_{21}b_{12} + a_{22}b_{22}$$

$$t(n) = 8t(n/2) + \Theta(n^2)$$

$$= \Theta(n^3)$$

---

## Strassen (1969)

$$m_1 = (a_{21} + a_{22} - a_{11})(b_{22} - b_{12} + b_{11})$$

$$m_2 = a_{11} b_{11}$$

$$m_3 = a_{12} b_{21}$$

$$m_4 = (a_{11} - a_{21})(b_{22} - b_{12})$$

$$m_5 = (a_{21} + a_{22})(b_{12} - b_{11})$$

$$m_6 = (a_{12} - a_{21} + a_{11} - a_{22})b_{22}$$

$$m_7 = \cancel{a_{22}}(b_{11} + b_{22} - b_{12} - b_{21})$$

$$C = A \times B$$

$$= \begin{pmatrix} m_2 + m_3 & m_1 + m_2 + m_5 + m_6 \\ m_1 + m_2 + m_4 - m_7 & m_1 + m_2 + m_4 + m_5 \end{pmatrix}$$

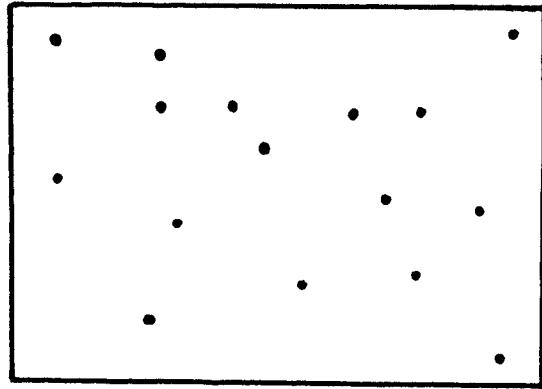
$$t(n) = 7t(n/2) + \Theta(n^2)$$

$$= \Theta(n^{\lg 7})$$

$$= O(n^{2.81})$$

---

## Closest Pair



ក្នុងរៀបចំណីកល់នេះ មានកូនប៉ុសតិចប៉ុណ្ណោះទេ?

Brute force : គុណភាពចំណុច =  $\binom{n}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$

D & Q : ① រៀបចំក្នុងជំនួយ 2 នៅ ចំណាំ, ចំណាំ

② ស្វែងរក closest pair ចំណាំ

③ " \_\_\_\_\_ . ចំណាំ

④ " \_\_\_\_\_ . កំណត់

$$t(n) = 2t(n/2) + t(4)$$

$$\text{ពី } t(4) = \Theta(n),$$

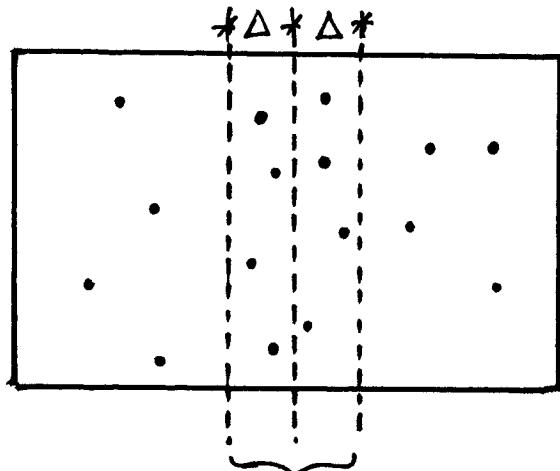
$$t(n) = \Theta(n \log n)$$

① ໄຟ ດີເລ ດັ່ງນີ້ແມ່ນກວດ closest pair ລັກຈິຕ

dr ດັ່ງ " — . — . ລາ "

$$\Delta = \min(d_e, dr)$$

ໃຫ້ ④ ພິບ



ລະຫວີງເພື່ອ: ກຸດິນ "ແຮບ" ນີ້

```
for(i < 1 to s)
    for(j < i+1 to s)
        if (distance(Pi, Pj) < Δ)
            Δ = distance(Pi, Pj)
```

worst case :  $\Theta(n^2)$

II

សំណង់លេខ: រូបភាព

រូបភាសាក់សមារណក់ ជំនួយ <  $\Delta$

រចនាប្រព័ន្ធភាសាអ៊ីនិក និងអ៊ីនិក <  $\Delta$

for ( $i \leftarrow 1$  to  $S$ )

    for ( $j \leftarrow i+1$  to  $S$ )

        if ( $|y(P_i) - y(P_j)| > \Delta$ )

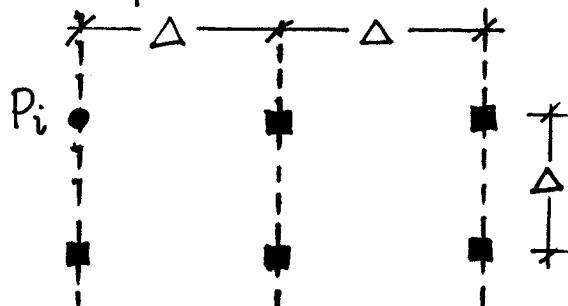
            break // inner loop

    else

        if ( $\text{distance}(P_i, P_j) < \Delta$ )

$\Delta = \text{distance}(P_i, P_j)$

"រូប  $P_i$  នឹងរូប  $P_j$  ជាប្រើប្រាស់ក្នុង  $\Delta$  ទៅលើ"  $\Delta$



$\therefore$  ④ នំនាំ  $\Theta(n)$

និងពីរ sort រូបីនុ នៃបញ្ហាយ  $\Theta(n \log n)$

$$t(n) = 2t(n/2) + \Theta(n \log n) = \Theta(n \log^2 n)$$

# Dynamic Programming

- typically applied to optimization problems
  - solves problems by combining the solutions to subproblems
  - solves every subproblem just once.
- 

## Q1.v. Fibonacci number

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 2$$

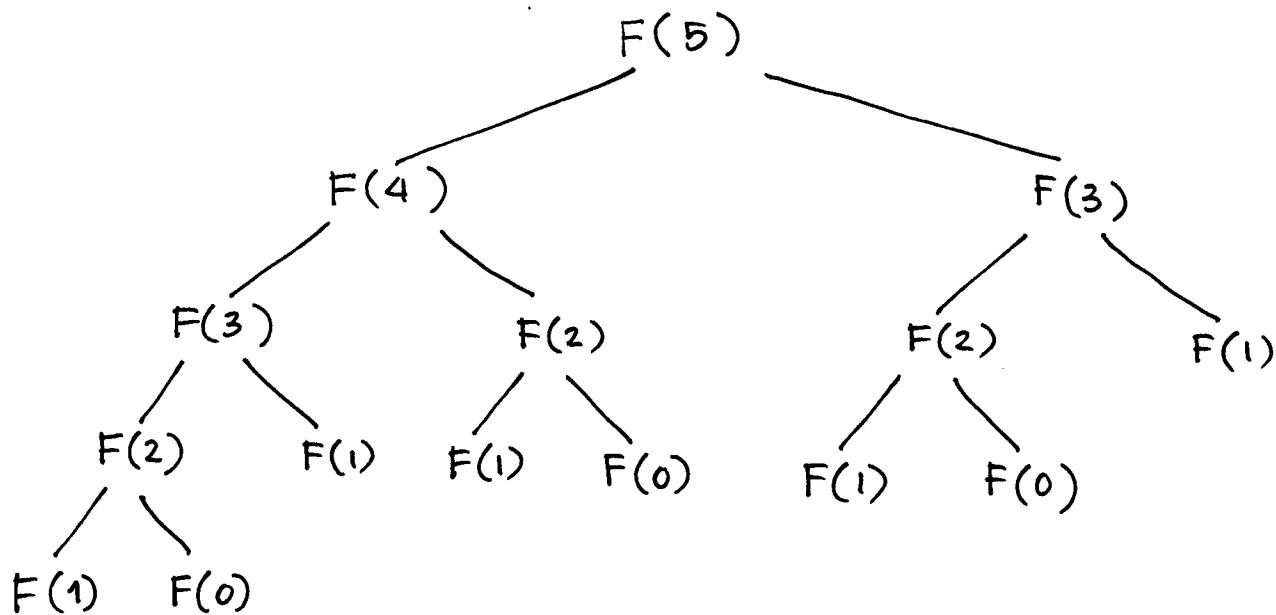
$$f_0 = 0, \quad f_1 = 1$$

①

```
int Fib1( int n )
    if n < 2 then return n
    return( Fib1(n-1) + Fib1(n-2) )
```

$$t_n = t_{n-1} + t_{n-2} + O(1)$$

$$= O(\phi^n) \quad \phi = \frac{1+\sqrt{5}}{2}$$



→ 15 नं. Fib1() का अर्थ है।

2

Fib2( int n )

$F[0] \leftarrow 0$

$F[1] \leftarrow 1$

for  $i=2$  to  $n$

$F[i] = F[i-1] + F[i-2]$

return  $n$

$$t_n = \Theta(n)$$

## Making Change

- វិធីលក្ខណៈនៃប្រាក់ការងារ  
 $d_i$  ជាន់តែចាប់បូន្ត់ប្រាក់ការងារទី  $i$
  - ទីតាំងរាយការងារនៃការងារទី  $N$  នឹង  
ត្រូវបានដោឡើងនៅលម្អិត
- 

ស្រុងពារាំង  $C[1..n, 0..N]$

$C[i, j]$  គឺជាដារាងបែងចែកនៃការងារទី  $i$  ដែលត្រូវបានដោឡើងនៅលម្អិត  $j$  ដល់ទី  $i$ .  
ស្រុងពារាំង  $C[1..n, 0..N]$

①  $c[i, 0] = 0 \quad 1 \leq i \leq n$

②  $c[i, j]$   $c[i-1, j]$   $\leftarrow$  មានវិធីបែងចែកនៃការងារទី  $i$   
 $1 + c[i, j - di]$   $\leftarrow$  មានវិធីបែងចែកនៃការងារទី  $i$   
និង  $1$  បែងចែក.

$$c[i, j] = \min \{ c[i-1, j], (1 + c[i, j - di]) \}$$

Q.V.

$$N = 8, n = 3$$

$$d_1 = 1, d_2 = 4, d_3 = 6$$

	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0	1	2	3	4	5	6	7	8
$d_2 = 4$	0	1	2	3	1	2	3	4	2
$d_3 = 6$	0	1	2	3	1	2	1	2	<span style="border: 1px solid black; padding: 2px;">2</span>

$\Theta(nN)$  time

# Knapsack Problem (0/1)

- $n$  objects
- weight  $w_i$
- value  $v_i$
- knapsack  $W$

$$\text{maximize } \sum_{i=1}^n x_i v_i$$

$$\text{subject to } \sum_{i=1}^n x_i w_i \leq W$$

$$x_i = 0, 1$$


---

សរុបតារាង  $V[1..n, 0..W]$

$V[i, j]$  កែង value នៃវគ្គភាពកីឡូន ដែល

ជាន់នៅ object  $i$  តិច  $j$  និង

knapsack មួយតិច  $j$

---

$$V[i, j] = \max \left\{ V[i-1, j], (v_i + V[i-1, j-w_i]) \right\}$$

## Matrix-chain Multiplication

$A_1 \ A_2 \ A_3$	$[10 \times 100] [100 \times 5] [5 \times 50]$
① $((A_1 A_2) A_3)$	$10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7,500$
② $(A_1 (A_2 A_3))$	$100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50 = 75,000$

$A_1 A_2 A_3 A_4$

- $(A_1 (A_2 (A_3 A_4)))$
  - $(A_1 ((A_2 A_3) A_4))$
  - $((A_1 A_2) (A_3 A_4))$
  - $((A_1 (A_2 A_3)) A_4)$
  - $((((A_1 A_2) A_3) A_4))$
- 

จำนวนการนับวิธี

จำนวนวิธี  $P(n)$  คือ จำนวนการนับวิธีที่มีวิธี

แมตทริกซ์  $n$  ตัว

$$( \boxed{n} )$$

$$( \boxed{k} ) ( \boxed{n-k} )$$

$$P(n) = \begin{cases} 1 & \text{if } n=1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2. \end{cases}$$

$\Omega\left(\frac{4^n}{n^{3/2}}\right)$

- $A_1 A_2 \cdots A_n$
- matrix  $A_i \rightarrow [p_{i-1} \times p_i]$

Ex.  $A_1 \cdot A_2 \cdot A_3$

10 100 5 50

$P_0 P_1 P_2 P_3$

---

Let  $m[i, j]$  តើចំណាំទម្រង់របស់ scalar  
កំណើលដ្ឋាននៃការចូល  
matrix chain  $A_i A_{i+1} \cdots A_j$

$$(A_i A_{i+1} \cdots A_k)(A_{k+1} \cdots A_j)$$

$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} \cdot p_k \cdot p_j$$


---

$$m[i, j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \leq k < j} \{ m[i, k] + m[k+1, j] + p_{i-1} \cdot p_k \cdot p_j \} & \text{if } i < j \end{cases}$$


---

ឬ  $s[i, j]$  តើថា  $k$  បានចូលរួច  $A_i A_{i+1} \cdots A_j$

និងវាបានចូលរួច scalar កំណើលដ្ឋាន

Q1. U. A<sub>1</sub> A<sub>2</sub> A<sub>3</sub> A<sub>4</sub>

13 5 89 3 34

P<sub>0</sub> P<sub>1</sub> P<sub>2</sub> P<sub>3</sub> P<sub>4</sub>

	j = 1	2	3	4
i = 1	0			
2		0		
3			0	
4				0

① u7 m[1,2], m[2,3], m[3,4]

$$m[1,2] = m[1,1] + m[2,2] + 13 \times 5 \times 89 = 5785$$

$$m[2,3] = m[2,2] + m[3,3] + 5 \times 89 \times 3 = 1335$$

$$m[3,4] = m[3,3] + m[4,4] + 89 \times 3 \times 34 = 9078$$

② u7 m[1,3], m[2,4]

$$\begin{aligned} m[1,3] &= \min \left\{ (m[1,1] + m[2,3] + 13 \times 5 \times 3), \right. \\ &\quad \left. (m[1,2] + m[3,3] + 13 \times 89 \times 3) \right\} \\ &= \min \{ 1530, 9256 \} = 1530 \end{aligned}$$

$$\begin{aligned} m[2,4] &= \min \left\{ (m[2,2] + m[3,4] + 5 \times 89 \times 34), \right. \\ &\quad \left. (m[2,3] + m[4,4] + 5 \times 3 \times 34) \right\} \\ &= \min \{ 24208, 1845 \} = 1845 \end{aligned}$$

③  $m[1,4]$

$$\begin{aligned}m[1,4] &= \min \{ (m[1,1]+m[2,4]+13 \times 5 \times 34), \\&\quad (m[1,2]+m[3,4]+13 \times 89 \times 34), \\&\quad (m[1,3]+m[4,4]+13 \times 3 \times 34) \} \\&= \min \{ 4055, 54201, 2856 \} = 2856\end{aligned}$$

---

MATRIX-CHAIN( $p, n$ )

for  $i \leftarrow 1$  to  $n$  do  $m[i,i] \leftarrow 0$

for  $l \leftarrow 2$  to  $n$  do

for  $i \leftarrow 1$  to  $n-l+1$  do

$j \leftarrow i+l-1$

$m[i,j] \leftarrow \infty$

for  $k \leftarrow i$  to  $j-1$  do

$g \leftarrow m[i,k]+m[k+1,j]+$

$P_{i-1} \cdot P_k \cdot P_j$

if  $g < m[i,j]$  then

$m[i,j] \leftarrow g$

$S[i,j] \leftarrow k$

$\Theta(n^3)$  time

$\Theta(n^2)$  space

return  $m$  and  $s$

## Memoization

- memory function

```
MEMOIZED-MATRIX-CHAIN( p, n )
```

```
    for i ← 1 to n
```

```
        for j ← i to n
```

```
            m[i, j] ← ∞
```

```
    return LOOKUP-CHAIN( p, 1, n )
```

```
LOOKUP-CHAIN( p, i, j )
```

```
    if m[i, j] < ∞ then return m[i, j]
```

```
    if i = j then m[i, j] ← 0
```

```
    else for k ← i to j - 1
```

```
        g ← LOOKUP-CHAIN( p, i, k ) +
```

```
        LOOKUP-CHAIN( p, k + 1, j ) +
```

```
        Pi-i · Pk · Pj
```

```
        if g < m[i, j] then m[i, j] ← g
```

```
    return m[i, j]
```

# Virtual Initialization

- $T[1..n]$
- $a[1..n], b[1..n], c$

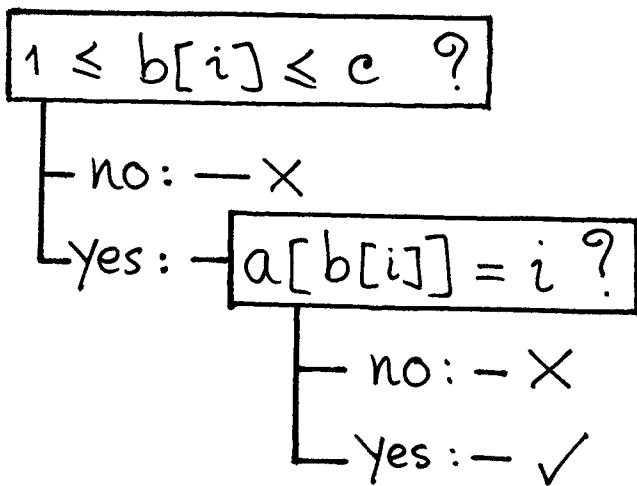
	1	2	3	4	5	6	7	8
T		-6		17			24	

a	4	7	2					
---	---	---	---	--	--	--	--	--

b		3		1			2	
---	--	---	--	---	--	--	---	--

c	3
---	---

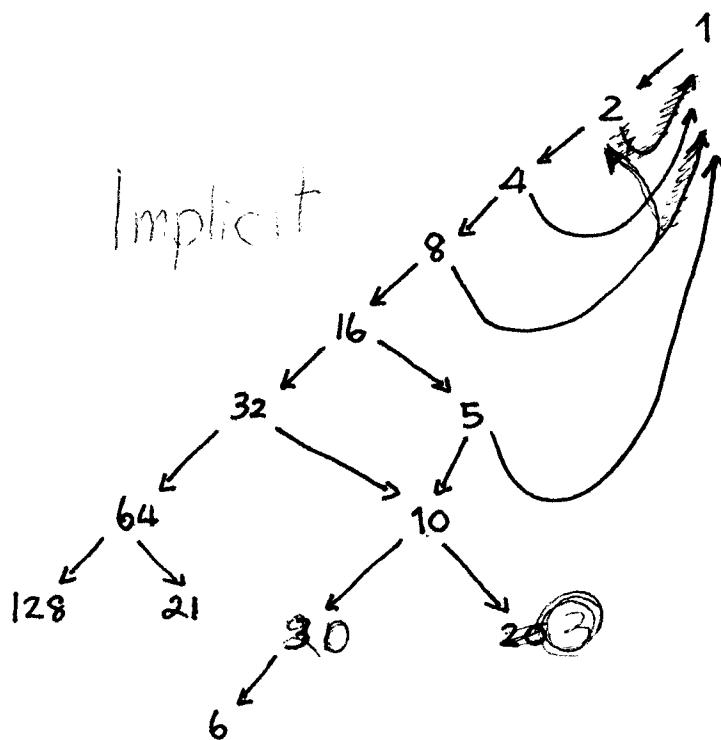
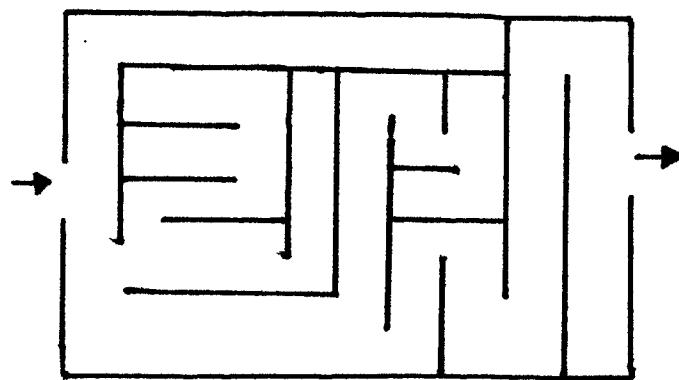
అగ్గకింది  $T[i]$  వఱాగెబంచు అస్తిత్వాన్ని



# Dynamic Programming

- characterize the structure of an optimal sol<sup>n</sup>.  
" optimal substructures"
- recursively define the value of an optimal sol<sup>n</sup>.  
" overlapping subproblems"
- compute the value in a bottom-up fashion .
- construct an optimal sol<sup>n</sup>. from computed info.

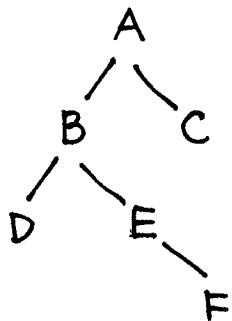
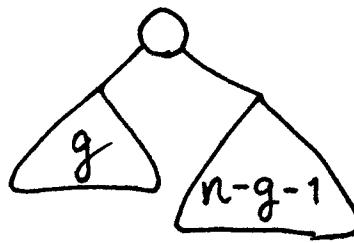
# Search and Traversal



$$3 = 1 \times 2 \times 2 \times 2 \times 2 \div 3 \times 2$$

## Tree Traversal

- pre order
- in order
- post order



preorder : A, B, D, E, F, C

inorder : D, B, E, F, A, C

postorder : D, F, E, B, C, A

$$T(n) \leq \max_{0 \leq g \leq n-1} \{ T(g) + T(n-g-1) + c \}$$

$$\text{Let } T(n) \leq an + b$$

Basis :  $n=0$  choose  $b \geq c$

$$\begin{aligned} \text{Induction: } T(n) &\leq \max_{0 \leq g \leq n-1} \{ (ag+b) + (a(n-g-1)+b) + c \} \\ &\leq an + 3b - a \end{aligned}$$

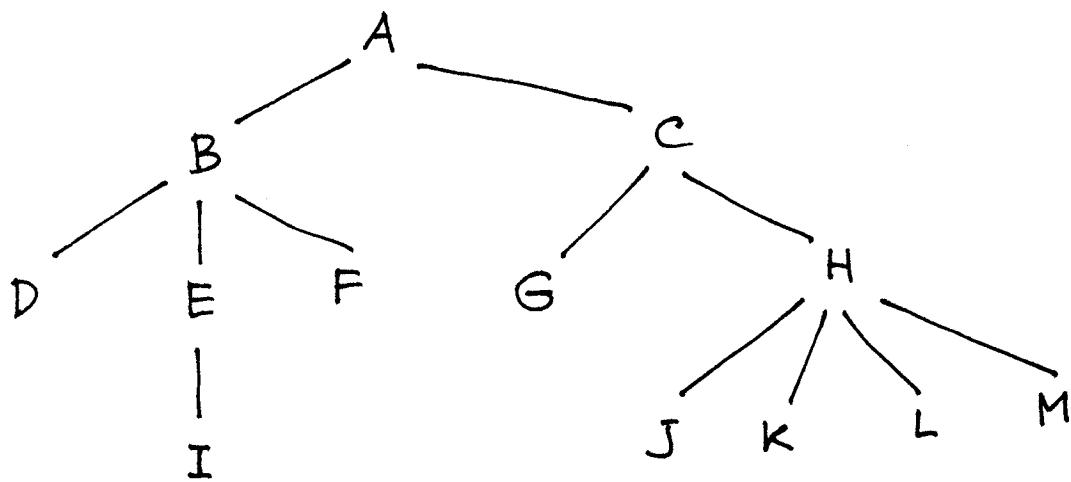
choose  $a \geq 2b$

$$\therefore T(n) \leq an + b \rightarrow O(n)$$

$$T(n) = \Omega(n)$$

$$\left. \right\} \Theta(n)$$

## Ancestry in a rooted tree



Determine whether  $v$  is an ancestor of  $w$

- $\Omega(n)$  alg.
- preprocessing in  $\Theta(n)$   
subsequent query in  $O(1)$

---

$$\text{prenum}[v] \leq \text{prenum}[w] \rightarrow \textcircled{v}$$

- $v$  is an ancestor of  $w$  or
  - $v$  is to the left of  $w$
- 

$$\text{postnum}[v] \geq \text{postnum}[w]$$

- $v$  is an ancestor of  $w$  or
  - $v$  is to the right of  $w$
-

# Graph Search & Traversal

- Depth-first
  - Breadth-first
- 

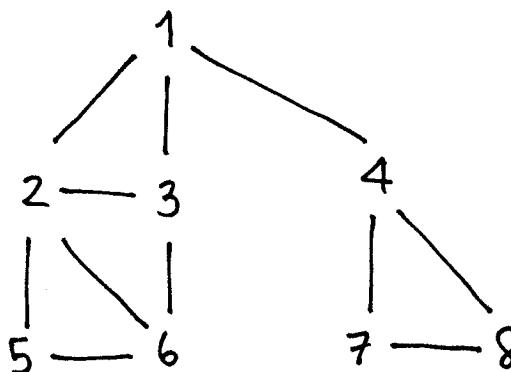
## Depth-first Search : Undirected graphs

DFS( $v$ )

mark[ $v$ ]  $\leftarrow$  visited

for each node  $w$  adjacent to  $v$  do

if mark[ $w$ ]  $\neq$  visited then  $DFS(w)$



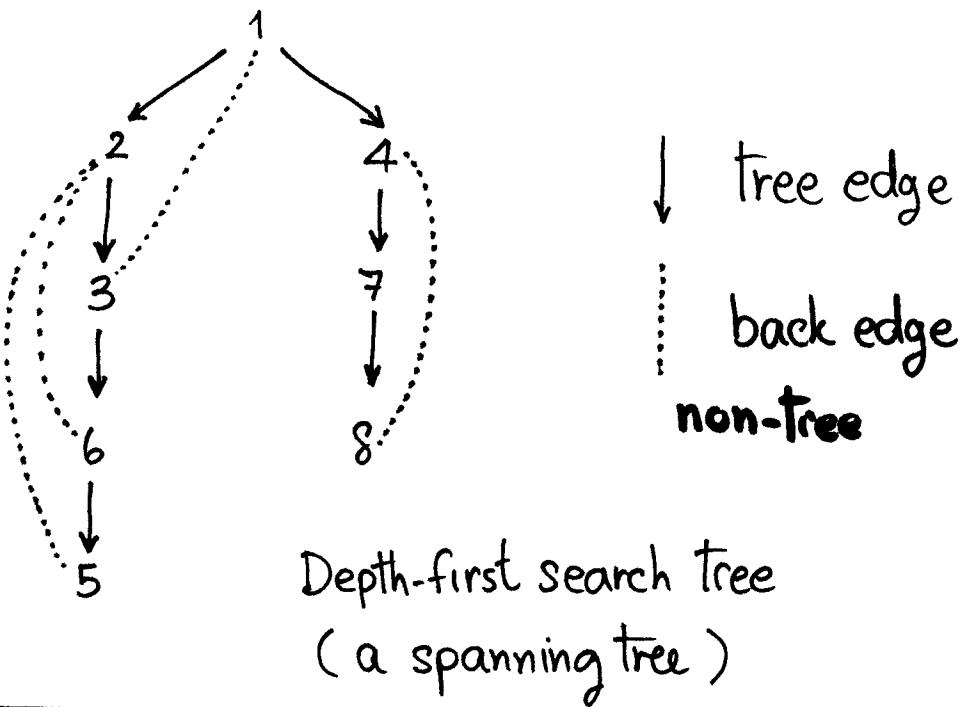
DFS(1)

DFSearch( $G$ )

for each  $v \in V$  do  $mark[v] \leftarrow$  not-visited

for each  $v \in V$  do

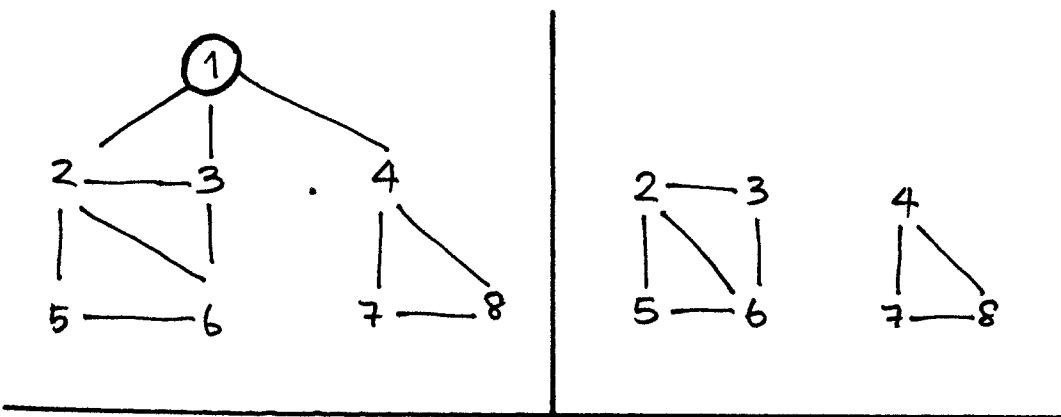
if  $mark[v] \neq$  visited then  $DFS(v)$



$G = (V, E)$  has  $v$  vertices,  $e$  edges.

- visit node  $n$ :  $\Theta(v)$   $\rightarrow \propto v$
  - of mark  $\Theta(v)$  neighbouring nodes  $\rightarrow \propto e$   
( $\Theta$  adjacency list)
- 
- $\Theta(\max(e, v))$

## Articulation Points

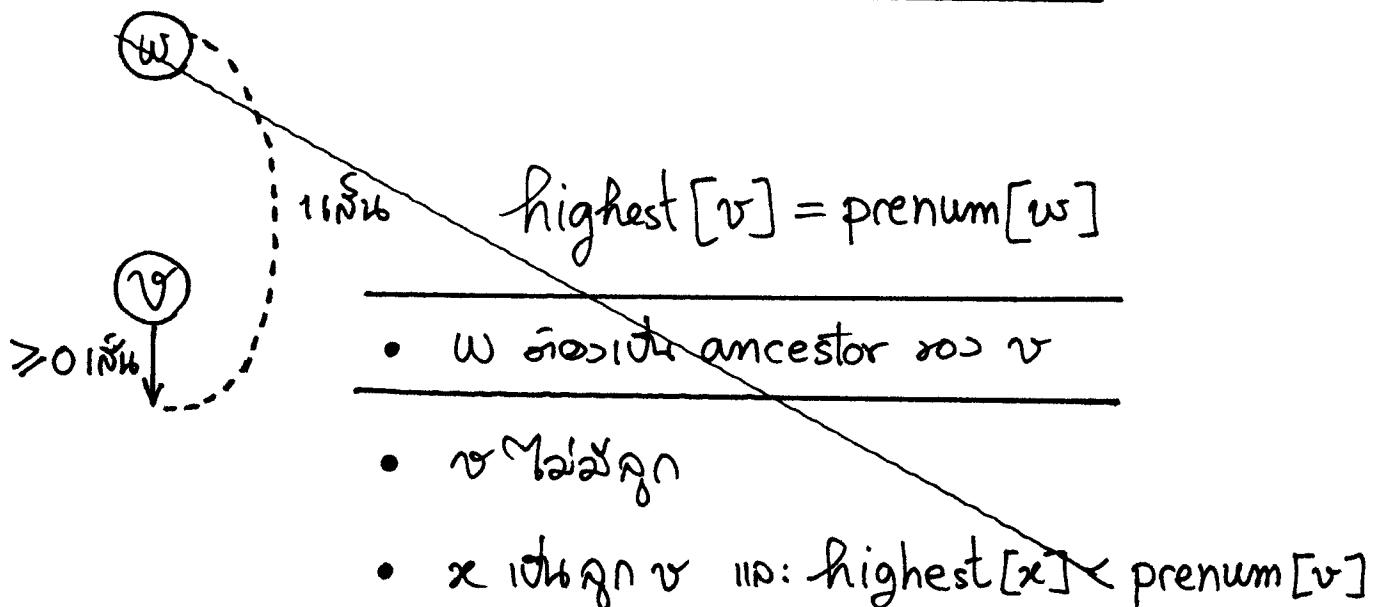


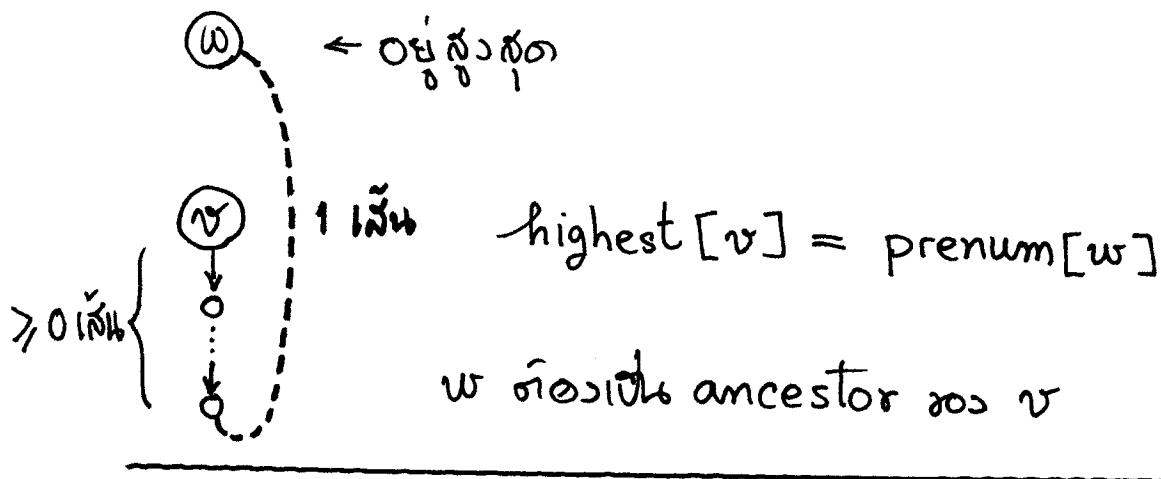
now G connected

at articulation point 200 G

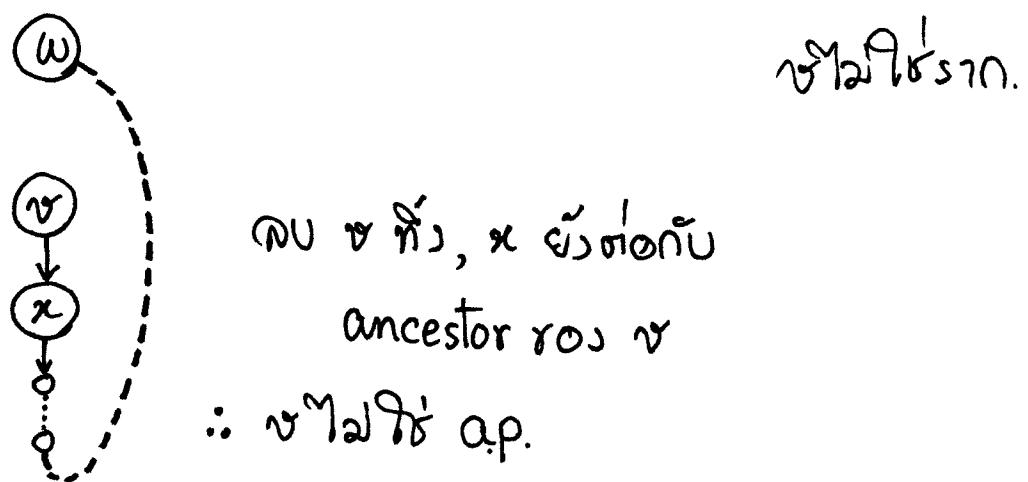
ກຳໄຟມີ subgraph ຖະແຫຼນທີ່ 1 component

Biconnected graph  $\Rightarrow$  no articulation points.

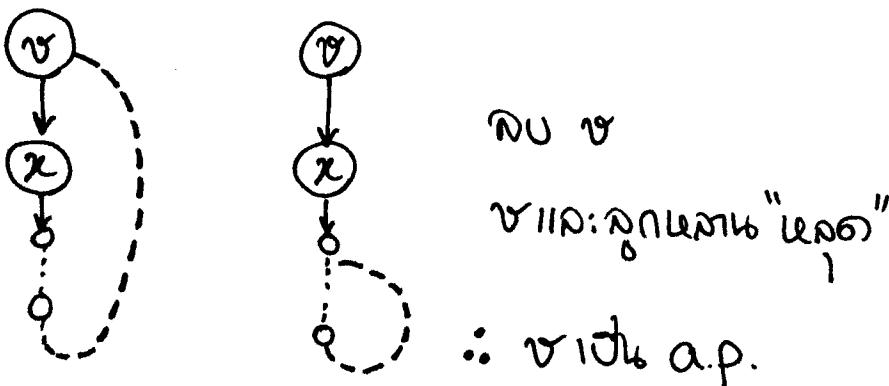


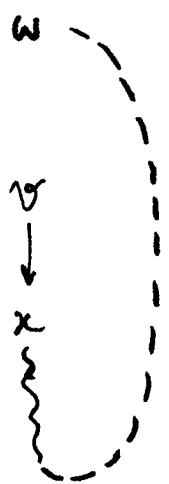
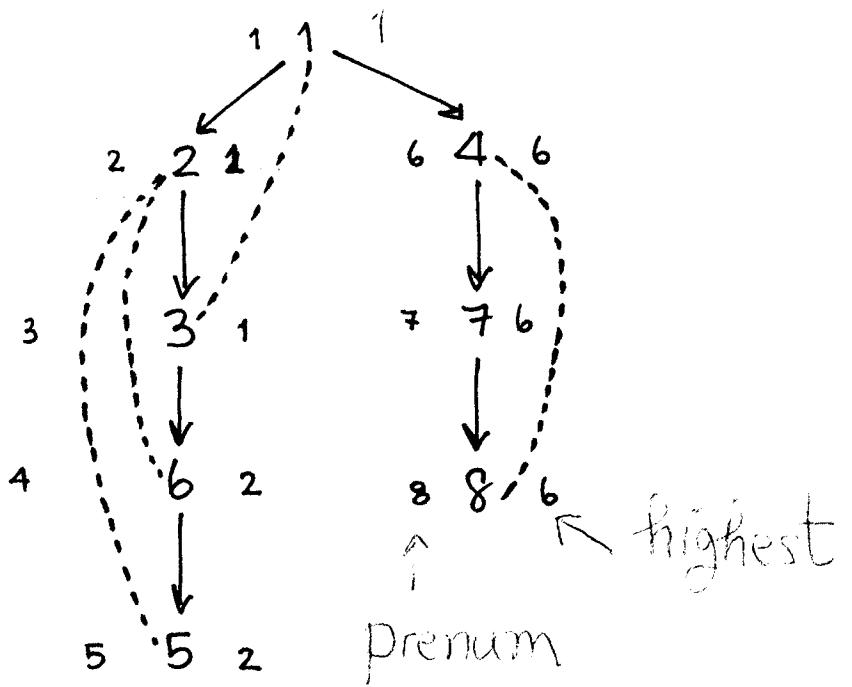


- ① ការ ចាយឲ្យក្នុង  $\rightarrow$  ចាយឲ្យទៅ articulation point.
- ② ការ  $x$  បើន្បាគរបស់  $v$  ឱ្យ:  $\text{highest}[x] < \text{pnum}[v]$



- ③ ការ  $x$  បើន្បាគរបស់  $v$  ឱ្យ:  $\text{highest}[x] \geq \text{pnum}[v]$





## Finding Articulation Points

- ① prenum ပါ၍ စာက DFS

pnum ++;

pnum[v]  $\leftarrow$  pnum

---

- ② traverse DFS tree in postorder

highest[v] = min. of

- prenum[v]
  - prenum[w] ( $v, w$ ) is a back edge
  - highest[x]  $x \in$  ပေးလွှားသူ
- 

- ③ • root သူ DFS tree စုံ a.p. ငါးခု နဲ့  $> 1$

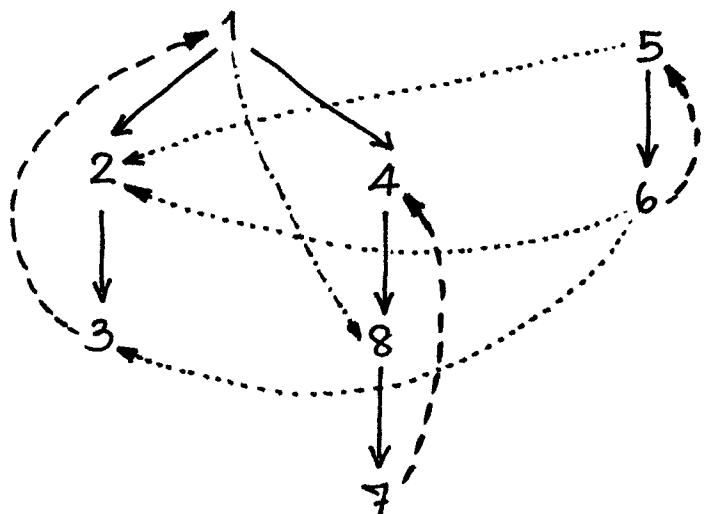
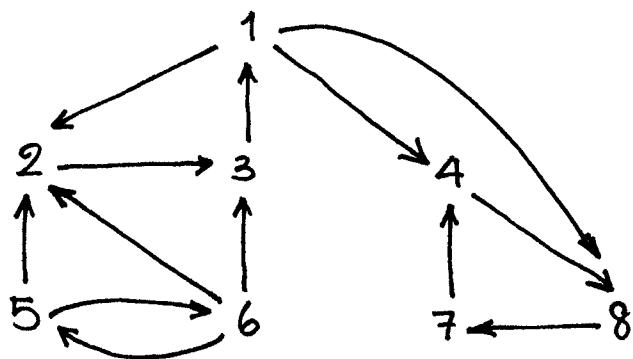
- node  $v$  စုံ a.p. ဒိုက်ခဲ့တဲ့ သမ္မတ  $v'$

highest[x]  $\geq$  prenum[v]

---

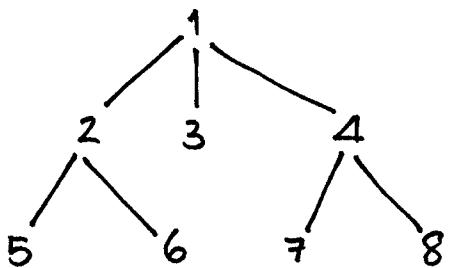
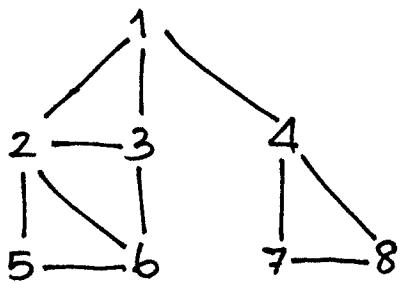
## Depth-first Search : Directed graphs

- คล้ายวิธีการทำ DFS บน undirected graph.
- "adjacent" นับทั้ง 2 ด้าน edge ตัว



- tree edge
- back edge
- forward edge
- cross edge

## Breadth-first Search



BFS tree

$\text{BFS}(v)$

$Q \leftarrow \emptyset$

$\text{mark}[v] \leftarrow \text{visited}$

$\text{enqueue}(Q, v)$

while  $Q \neq \emptyset$  do

$u \leftarrow \text{dequeue}(Q)$

for each node  $w$  adj. to  $u$  do

if  $\text{mark}[w] \neq \text{visited}$  then

$\text{mark}[w] \leftarrow \text{visited}$

$\text{enqueue}(Q, w)$

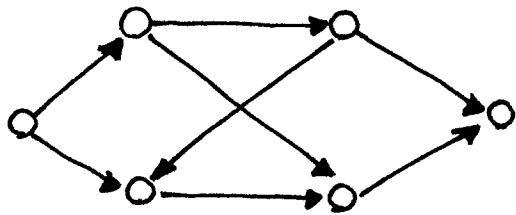
$\Theta(\max(e, v))$

• adjacency list

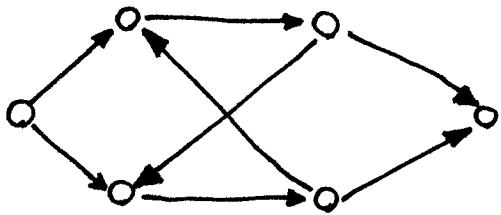
BFSearch 用於 DFS

DFS 用於 BFS

# Directed Acyclic Graphs



✓

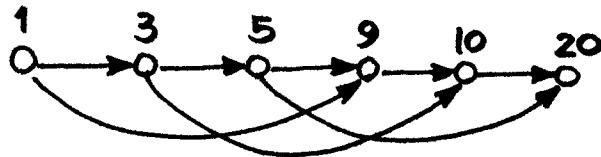
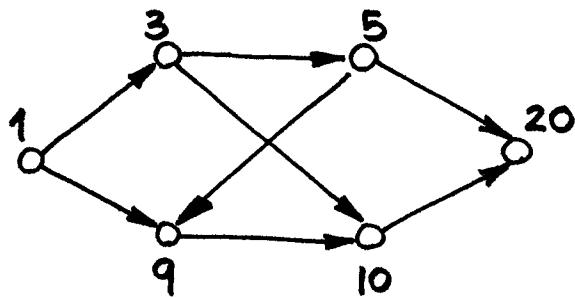


✗

Is  $G$  a DAG?

- DFSearch( $G$ )
- $G$  is acyclic iff no back edge

# Topological Sorting



- 
- ① ដើរ "push v" ក្នុងការកំណត់លទ្ធផល DFS
  - ② ធ្វើការ DFSearch ( G )
  - ③ ត្រូវបញ្ជូនក្នុងការកំណត់លទ្ធផល pop stack ឡើងទៅ
-

# Backtracking

- solution :  $(x_1, x_2, \dots, x_n)$

$x_i$  នៅលើនាមុខ finite set  $S_i$

- satisfy criterion function

$$P(x_1, x_2, \dots, x_n)$$

---

## Brute force :

ក្នុងសំណើ  $S_i$  មាន  $m_i$

$\therefore$   $m_1 m_2 \dots m_n$  possible candidates

## Backtracking :

- រាយការ candidates តាមដឹង  
ដែលនឹង brute force ឡាយ

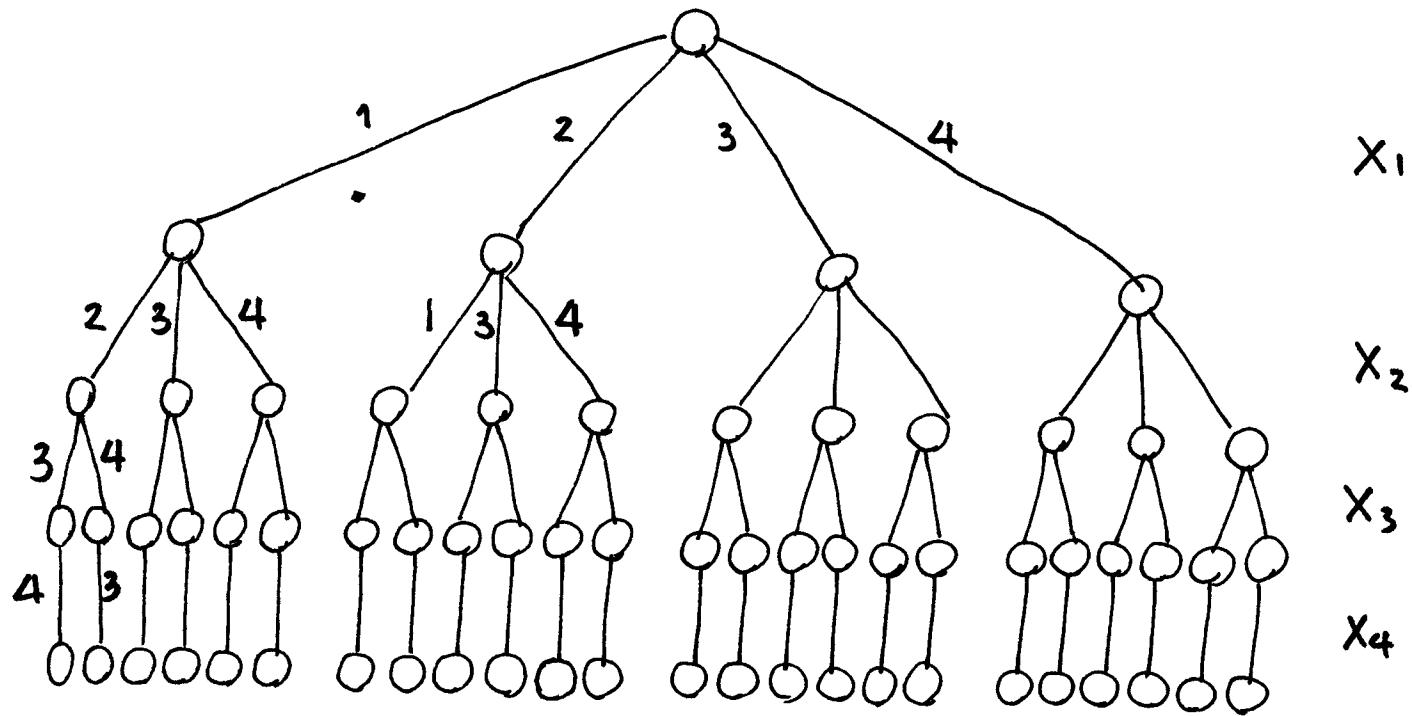
- រាយការ "partial" vector

$$(x_1, x_2, \dots, x_i)$$

និងត្រូវជាបញ្ជីក្នុងការបង្កើត

រាយការនៃ  $m_{i+1} \dots m_n$  vectors.

## 4 - Queen



## Sum of Subsets

Given  $\{w_1, w_2, w_3, \dots, w_n\}$  bbl: M

စောက် subset ၏ လုပ်သည့် နည်းစွမ်း  
သုတေသနများ M

---

E.U.  $\{2, 1, 3, 5, 10, 4\}$ , 8

ရှိခိုင်း  $\{1, 3, 4\}, \{3, 5\}, \{2, 1, 5\}$

---

### ပုဂ္ဂန်များ

①  $(x_1, x_2, \dots, x_k) \quad 1 \leq k \leq n$

- $\sum_{i=1}^k w_{x_i} = M$
- $x_i < x_{i+1}$

(2, 3, 6)  
(3, 4)  
(1, 2, 4)

②  $(x_1, x_2, \dots, x_n)$

- $\sum_{i=1}^n w_i \cdot x_i = M$
- $x_i = 0 \text{ or } 1$

(0, 1, 1, 0, 0, 1)  
(0, 0, 1, 1, 0, 0)  
(1, 1, 0, 1, 0, 0)

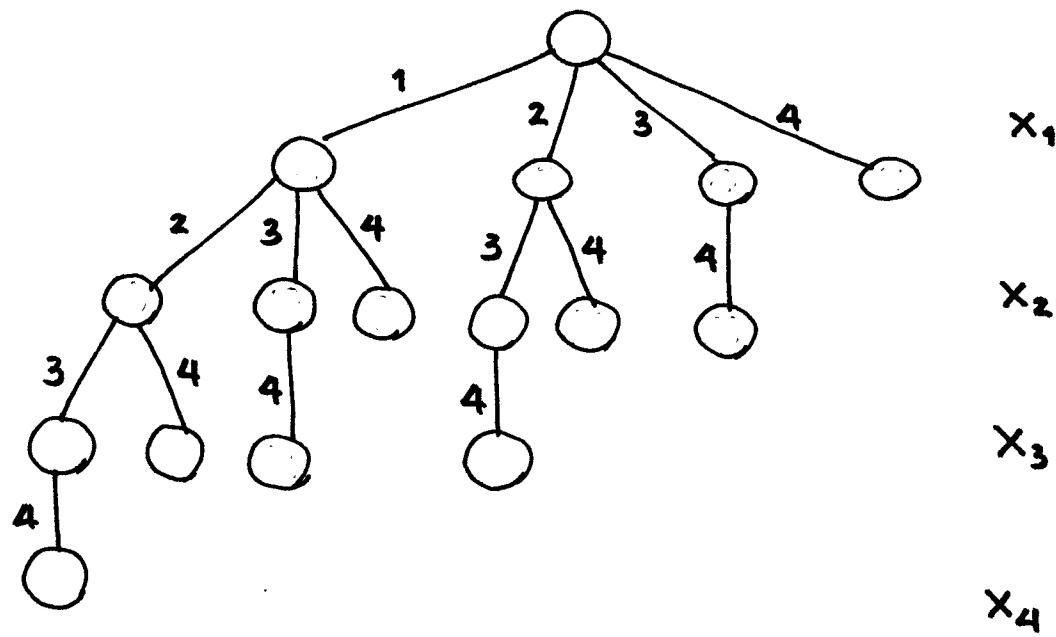
---

Solution space အမြတ်  $2^n$

## State Space Trees

### Sum of Subsets

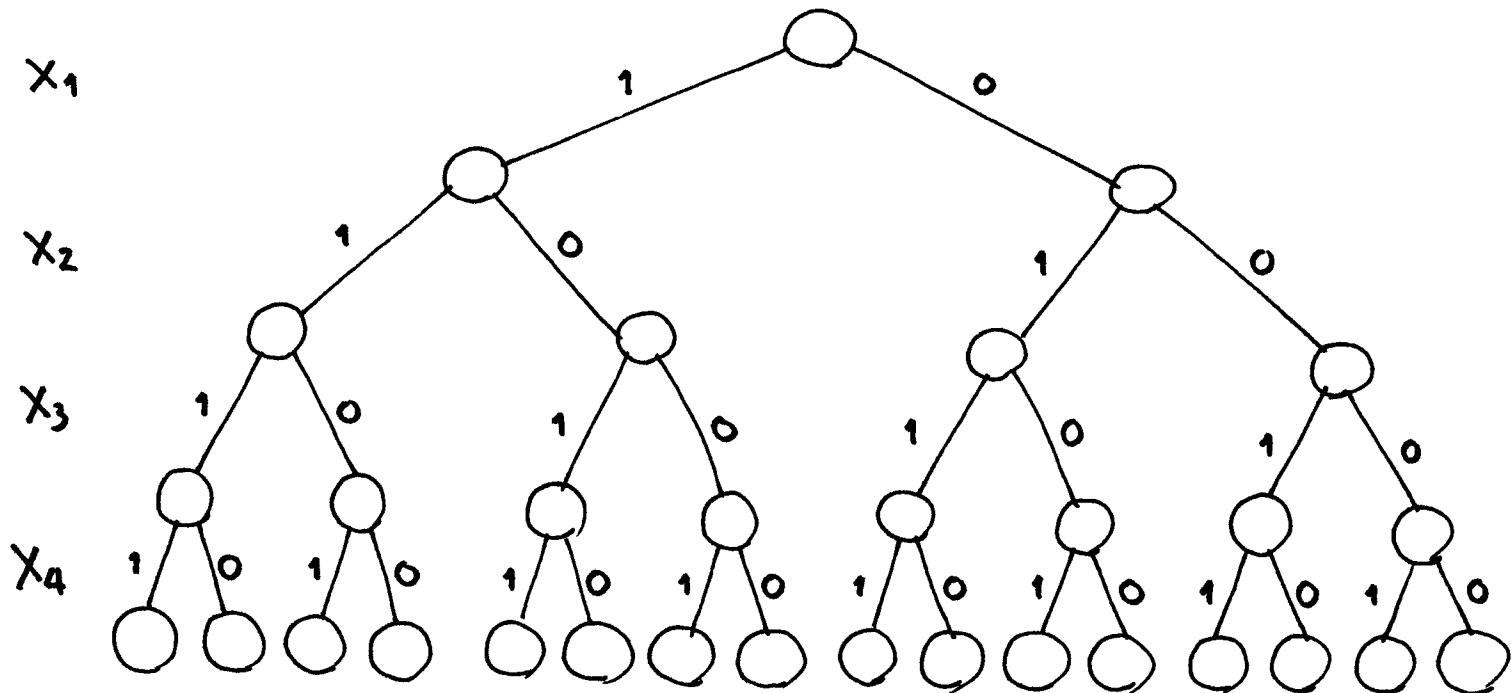
- $n = 4$
- $\sum_{i=0}^n \text{subset}_i$  ①



Q.U.  $\{10, 3, 5, 9\}$ , 12.

## Sum of Subsets

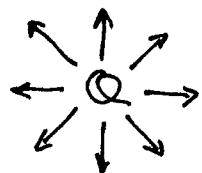
- $\sum_{i=1}^n x_i = 2^n$



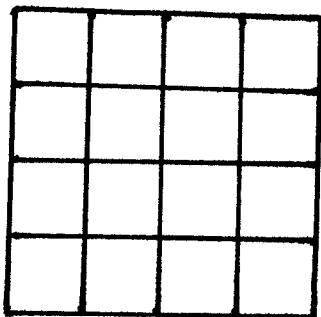
- problem states
- solution states
- answer states

## Eight-Queen Problem

- ဘາ Queen ចໍ່ນານ 8 ຕົວ  
ບໍລິສາງ 8x8 ໄຕີໄວ້ນີ້  
Queen ສອງຕັ້ງໃຫ້ "attack" ກັນ



- 4-Queen



① ເກືອກວາງ 4 ເຮັດຈາກ 16 ແທງ .  $\rightarrow \binom{16}{4} =$

② ແລກນະ: ຈົ້າ  $\rightarrow 4^4 =$

③ ແລກນະ: ຕັ້ງ ດອດລວມຜົນ: ຕັ້ງ  $\rightarrow 4! =$

"Solution space"

- 8-Queen
  - permutation ຮອງ  $(1, 2, 3, 4, 5, 6, 7, 8)$
  - ນິວ້າວ Queen "ເບີນ" ກົບ ອຸປະກາໂຄງ

# Backtracking

- Depth-first search ( state space tree ) + Bounding function ( promising ? )
- General template

Backtrack( $x[1..k]$ )

if  $x$  is an answer then write  $x$

else

for each  $(k+1)$ -promising vector  $w$

such that  $w[1..k] = x[1..k]$

do Backtrack( $w[1..k+1]$ )

4-queen problem

$x$	2	4		
-----	---	---	--	--

	Q		
			Q

Two Q's:  $(r_1, c_1), (r_2, c_2)$

same diagonal iff

$$|c_1 - c_2| = |r_1 - r_2|$$

## Bounding Function

n-queen problem:

Two Q's :  $(r_1, c_1), (r_2, c_2)$

are on the same diagonal iff

$$|c_1 - c_2| = |r_1 - r_2|$$

Sum of subsets : (solution ②)

assume  $w_i$  are initially  
in nondecreasing order.

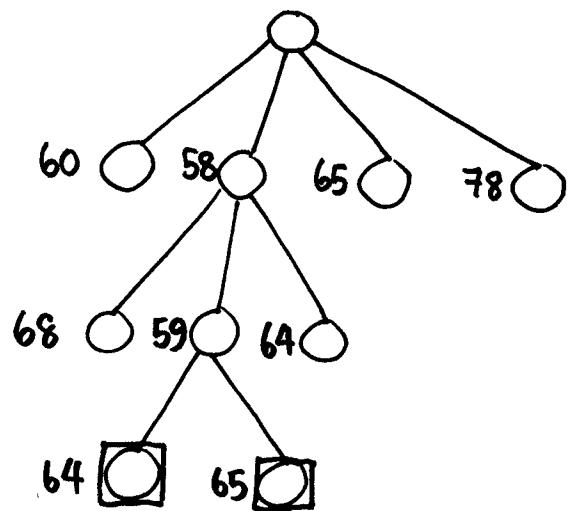
$x[1..k]$  maintains the following in

$$\textcircled{1} \quad \left( \sum_{i=1}^k w[i] \times x[i] \right) + w[k+1] > M$$

$$\textcircled{2} \quad \left( \sum_{i=1}^k w[i] \times x[i] \right) + \left( \sum_{i=k+1}^n w[i] \right) < M$$

## Branch-and-Bound

- Depth-first : stack
- Breadth-first : queue
- Least Cost : priority queue



LC-Search + Bounding function

ນີ້ໄດ້ 1 ຕໍາມລວກົບຕໍ່ 64

ສ່ວນຮັບຕໍ່ node ຕີ່ໃກ້  $> 64$  ເຄີຍ

# 15-puzzle

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

U

R

D

L

1	2		4
5	6	3	8
9	10	7	11
13	14	15	12

1	2	3	4
5	6	8	
9	10	7	11
13	14	15	12

1	2	3	4
5	6	7	8
9	10		11
13	14	15	12

1	2	3	4
5		6	8
9	10	7	11
13	14	15	12

4

4

2

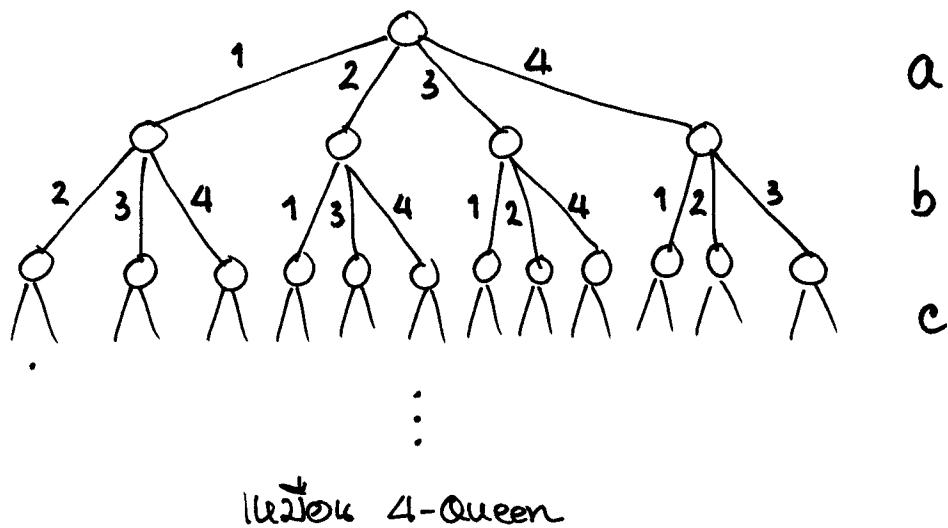
4

## Assignment Problem

- Այսինքն այսպիսում ուսումնական է առ
  - minimize total cost.

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

- state space tree



- Bounding function

- upper bound.

ເລື້ອກນັກ 1 feasible solution

$$a \rightarrow 1, b \rightarrow 2, c \rightarrow 3, d \rightarrow 4$$

$$11 + 15 + 19 + 28 = 73 \quad \text{"ອະທິງໄຕນົມບາງ"}$$

- lower bound

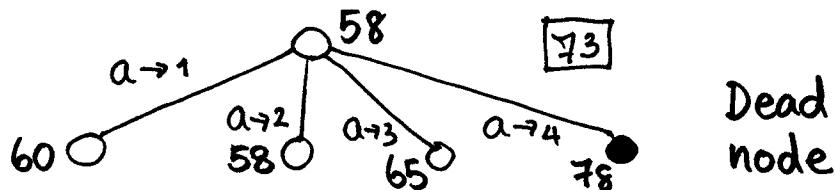
ຍອດວາມຂອງດໍານັກສຸດໃນແຕ່ລະ: column

$$11 + 12 + 13 + 22 = 58 \quad \text{"ອະທິງໄຕນົມບາງ"}$$

- answer node ຈົດກຳນົດຈົດ

$$[58 \dots 73]$$

- Branching

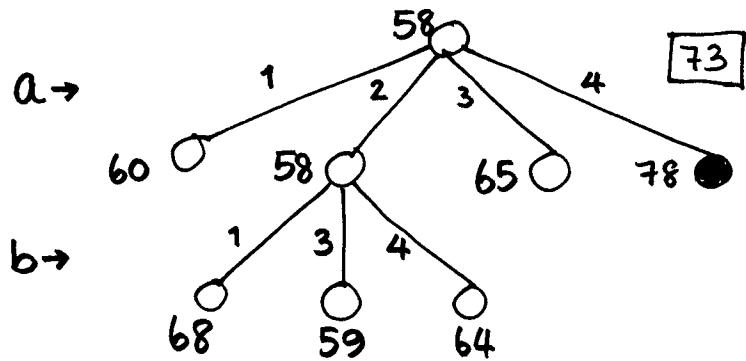


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

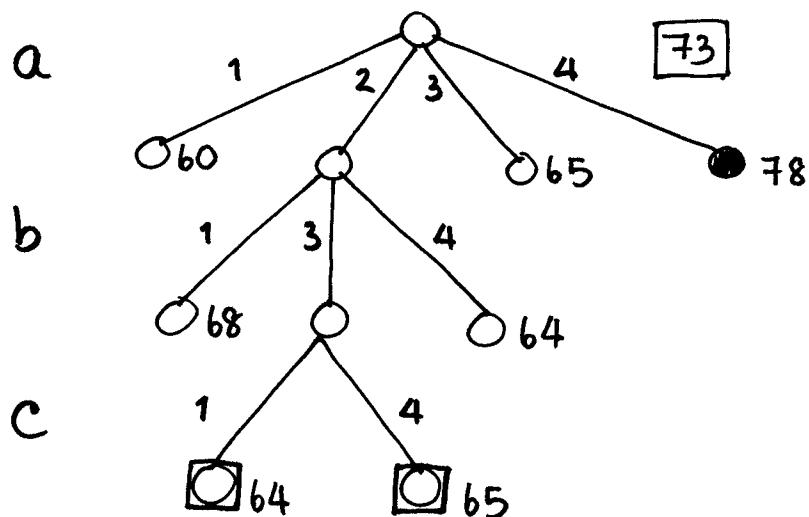
a->1	<u>11</u> + 14 + 13 + 22 = 60
a->2	<u>12</u> + 11 + 13 + 22 = 58
a->3	<u>18</u> + 11 + 14 + 22 = 65
a->4	<u>40</u> + 11 + 14 + 13 = 78 *

$$11 + 14 + 13 + 22 \\ a \rightarrow 1$$

$$12 + 11 + 13 + 22 \\ a \rightarrow 2$$



	1	2	3	4	
a	11	12	18	40	$a \rightarrow 2, b \rightarrow 1$
b	14	15	13	22	$a \rightarrow 2, b \rightarrow 3$
c	11	17	19	23	$a \rightarrow 2, b \rightarrow 4$
d	17	14	20	28	$12 + \underline{14} + 19 + 23 = 68$
					$12 + \underline{13} + 11 + 23 = 59$
					$12 + \underline{22} + 11 + 19 = 64$

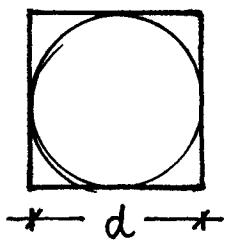


	1	2	3	4	
a	11	12	18	40	$a \rightarrow 2, b \rightarrow 3, c \rightarrow 1 (d \rightarrow 4)$
b	14	15	13	22	$12 + 13 + \underline{11} + 28 = 64$
c	11	17	19	23	$a \rightarrow 2, b \rightarrow 3, c \rightarrow 4 (d \rightarrow 1)$
d	17	14	20	28	$12 + 13 + \underline{23} + 17 = 65$

## Probabilistic Algorithms

- may behave differently when it is applied twice
  - may yield erroneous results
  - usually more efficient, less precise.
- 
- Numerical
    - provide an approx. to the correct answer
  - Monte Carlo
    - give the exact answer with high probability
  - Las Vegas
    - use probabilistic choices to quickly guide the search to a correct solution
    - never give an incorrect answer

# Numerical Probabilistic Alg.



$$\text{w.n. օղակ} = \frac{\pi d^2}{4}$$

$$\text{w.n. քառակույթ} = d^2$$

$A = \# \text{ զույգ պայմանավորություններ}$

$B = \# \text{ համապատասխան պայմանավորություններ}.$

$$\therefore \frac{A}{B} = \frac{\pi d^2}{4 d^2} = \frac{\pi}{4}$$

$$\pi = \frac{4A}{B}$$

---

# Monte Carlo Alg.

## Majority

1, 1, 2, 1, 1, 2, 2, 3, 1, 1, 1

1, 1, 2, 1, 1, 2, 2, 3, 3, 3, 1

Maj(  $T[1..n]$  )

$x \leftarrow T[\text{uniform}(1, n)]$

$c \leftarrow 0$

for(  $i \leftarrow 1$  to  $n$  )

if(  $x = T[i]$  )  $c++$

return(  $c > n/2$  )

$\frac{1}{2}$ -correct

Repeat Maj(  $T[1..n]$ ,  $k$  )

for(  $i \leftarrow 1$  to  $k$  )

if( Maj( $T$ ) = true )

return true

return false

## Monte Carlo Alg.

$C = AB$  ?       $n \times n$  matrices     $O(n^3)$

- Quí  $X$  •  $1 \times n$  matrix
  - randomly chosen binary vector
- nošou  $XAB = XC$  ?

MMult(  $A, B, C, n$  )

for  $i \leftarrow 1$  to  $n$

$X[i] \leftarrow \text{uniform}(0..1)$

if  $(XA)B = XC$  then return true

else return false

$\frac{1}{2}$ -correct

Repeat MMult(  $A, B, C, n, k$  )

for  $i \leftarrow 1$  to  $k$

if ( MMult(  $A, B, C, n$  ) = false )

return false

return true

## Reductions

- We are given a problem P
  - P seems similar to a known problem Q
  - We reduce (transform) P to Q, solve Q,  
Obtain solution of P.
- 

## Simple String-Matching

problem: Let  $A = a_0 a_1 \dots a_{n-1}$

$B = b_0 b_1 \dots b_{m-1}$

Is B a cyclic shift of A ?

ex: = "somchai" : "mchaiso"

Pattern matching: Let  $A = a_0 a_1 \dots a_{n-1}$

$B = b_0 b_1 \dots b_{m-1}$

Does A contain B ?

ex: "somchai somchai" : "mchaiso"

M: Matrix Multiplication

S: Matrix Squaring.

①  $A^2 = A \times A$

$$S \leq_T^P M$$

---

②  $\begin{bmatrix} AB & 0 \\ 0 & BA \end{bmatrix} = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}^2$

$$M \leq_T^P S$$

---

Polynomial-Time Reductions

ถ้า P can be reduced to Q.

① ถ้ามี algorithm ที่แก้ปัญหา Q

ก็จะ "—" "—" P

② ปัญหา Q ไม่ยากกว่า P

---

## NP-Completeness

Efficient algorithms

running times are bounded.

by some polynomial in the input size

---

Tractable problems

can be solved by efficient alg.

---

## Definition

P is the class of tractable decision problems

- ex.
- shortest path
  - minimum spanning tree
  - selection
  - summation
  - closest point
  - many many more ...

## Nondeterministic Algorithm

```
nd-search ( A[1..n], x )
```

```
    j < nd-choice ( 1, n )
```

```
    if A[j] = x then
```

```
        print(j)
```

```
        success
```

```
    else
```

```
        print(0)
```

```
        failure
```

A nondeterministic algorithm terminates unsuccessfully if and only if there exists no set of choices leading to a success signal.

nd-knapsack(  $P[1..n]$ ,  $W[1..n]$ ,  $M$ ,  $R$ ,  $X[1..n]$  )

for  $i \leftarrow 1$  to  $n$

$X[i] \leftarrow \text{nd-choice}(0, 1)$

$$\text{sum}_W = \sum_{1 \leq i \leq n} W[i] \cdot X[i]$$

$$\text{sum}_P = \sum_{1 \leq i \leq n} P[i] \cdot X[i]$$

if  $\text{sum}_W > M$  or  $\text{sum}_P < R$

then failure

else success

"polynomially verifiable"

## Definition

NP is a class of decision problems solvable by a nondeterministic algorithm in polynomial time.

Theorem :  $P \subseteq NP$

▷ សំណង់សំណុំ

$P = NP ?$

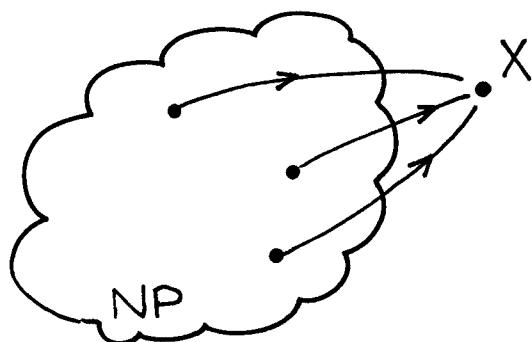
$P \neq NP ?$

## Definition

A decision problem  $X$  is NP-hard

if and only if  $Y \leq_T^P X$

for every problem  $Y \in NP$



# NP-Complete Problems

## Definition

A decision problem  $X$  is NP-Complete if

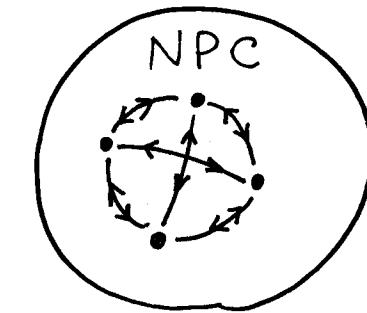
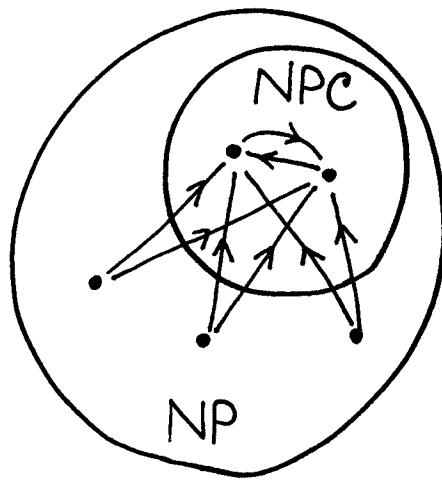
- $X \in \text{NP}$  and
- $X$  is NP-hard.

## Theorem

Let  $X$  be an NP-complete problem.

If  $Z \in \text{NP}$  and  $X \leq_T^P Z$ , then

$Z$  is also NP-complete



Ճանաչվող NPC սույնինեաց .

..... ցէս .....

## Satisfiability Problem

$S$  is Boolean expression

$\wedge \vee \neg \wedge \vee \wedge$  CNF

(Conjunctive Normal Form)

$$\text{Q.E.D. } S = (x+y+\bar{z}) \cdot (\bar{x}+y+z) \cdot (\bar{x}+\bar{y}+\bar{z})$$

Is  $S$  satisfiable?

- SAT-CNF is in NP

Cook's Theorem

SAT-CNF is NP-complete

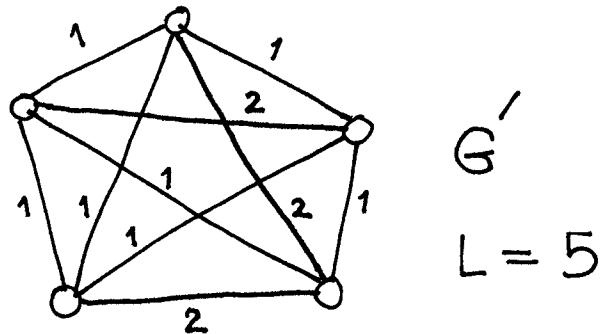
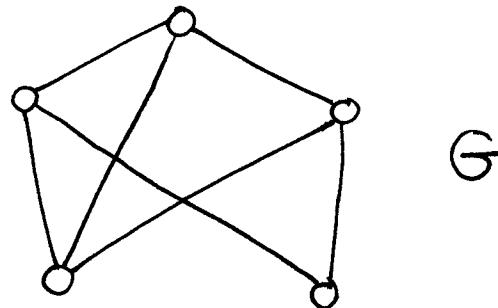
## NP-Complete Problems:

- SAT-3-CNF
- 3COL (Can  $G$  be painted with 3 colours?)
- HAMD (Is there any Hamiltonian cycle in  $G$ ?)
- TSPD (Is there any TS tour in  $G$  of length  $\leq L$ ?)

# Is TSPD NP-complete?

තුවානිත්කුතු නොවාගැනීම HAMD වේ NPC

- ① TSPD is in NP
- ② HAMD  $\leq_T^P$  TSPD



Is there a tour in  $G'$  of length  $\leq 5$ ?