Design and Analysis of Algorithms

การออกแบบและวิเคราะห์อัลกอริทึม

สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
2541

หนังสือ

1. Brassard and Bratley
   "Fundamentals of Algorithmics", Prentice-Hall, 1996

2. Cormen, Leiserson, and Rivest
   "Introduction to Algorithms", The MIT Press, 1990

การให้เกรด

การบ้าน	20%
สอบกลางภาค	40%
สอบปลายภาค	40%

ความรู้เบื้องต้น

- การเขียนโปรแกรมภาษาออบเจ็กต์เอเอร์
- โครงสร้างข้อมูล

การสื่อสาร

somchaip@chula.ac.th

สมชาย ประสิทธิ์ธุรกิจ

Analysis & Design of Algorithms

ಪರಿವಿಡಿ

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 (อัลกอริทึม, ขั้นตอนวิธี)

- 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

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

Output: a reordering $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that

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

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

↓

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

an instance of the sorting problem.

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

Elementary Operation

ต.ย. 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
```

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

Elementary operation :

operation whose execution time can be bounded above by a constant depending only on the particular implementation used
- machine
- programming lang.
- compiler

ต.ย. การบวกใช้เวลา t_a μs
การคูณ " t_m "
assignment " t_s "

อัลกอริทึม หนึ่ง มี การบวก a ครั้ง, การคูณ m ครั้ง
assignment s ครั้ง

$$t = a \cdot t_a + m \cdot t_m + s \cdot t_s$$

$$\leq \max(t_a, t_m, t_s) \times (a+m+s)$$

$$\propto (a+m+s)$$

∴ การวิเคราะห์อัลกอริทึม (เชิงเวลา) เป็นการนับจำนวน elementary operations ที่ทำงาน

ตัวอย่าง 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 ← 1 ; j ← 0
    for k ← 1 to n do  j ← i + j
                       i ← j - i
    return j
```

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

วิธีที่สาม

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

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

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

ถ้าอยากรู้ f_n ก็ หา $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^n$

ถ้าอยากหา $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^n$ ต้อง ฉลาด และ เร็ว

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

แปรตาม $\lg n$

$$\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^{10}=\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^{5}\times\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^{5}$$

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

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

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 length[A]	n
2 do key $\leftarrow$ A[j]	$n-1$
3 $i \leftarrow j-1$	$n-1$
4 while $i > 0$ and $A[i] > key$	$\sum_{j=2}^{n} t_j$
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 $A[i+1] \leftarrow key$	$n-1$

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

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

เวลา $T(n)$ แปรตาม n จำนวนข้อมูล

และ ลักษณะของข้อมูล

Best case : กรณีที่ข้อมูลเรียงลำดับ ตามน้อยไปมากอยู่แล้ว

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

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

"linear function"

Worst case : กรณีที่ข้อมูลเรียงกลับลำดับ (จากมากไปน้อย)

$$t_j = j$$

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

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

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

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

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

$$- (c_2 + c_3 + c_4 + c_8)$$ "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 $t : \mathbb{N} \to \mathbb{R}^{\geq 0}$ and $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$

w.l.u. $t(n) = 27n^2 + \frac{355}{133}n + 12$

$f(n) = n^2$

$t(n) \leq 27n^2 + \frac{355}{133}n^2 + 12n^2$

$= 42\frac{16}{133}n^2$

$= 42\frac{16}{133}f(n)$

$\Big\} \; t(n) \leq c\,f(n)$

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

O-notation

"asymptotic upper bound"

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

w.r.t.

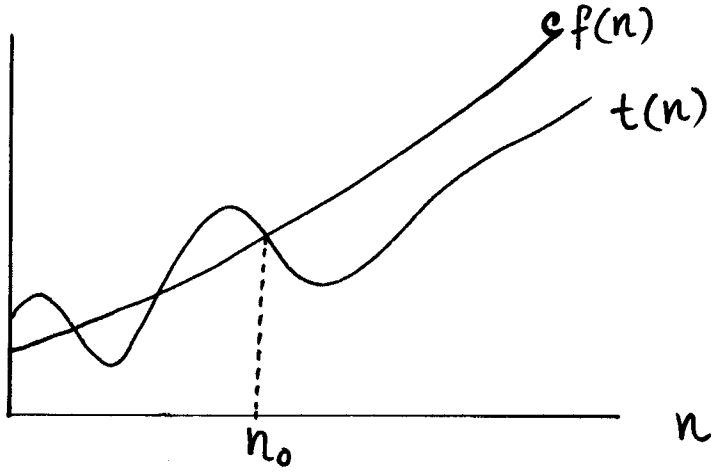
$O(n^2) = \{ n, \frac{1}{5}n, \sqrt{n}, 30n^2, 7n^2+5n-2, \ldots \}$

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

" " FibRec is $O(\phi^n)$

" " FibIter is $O(n)$

" " FibIter is $O(n^9)$



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

Ω - notation

"asymptotic lower bound"

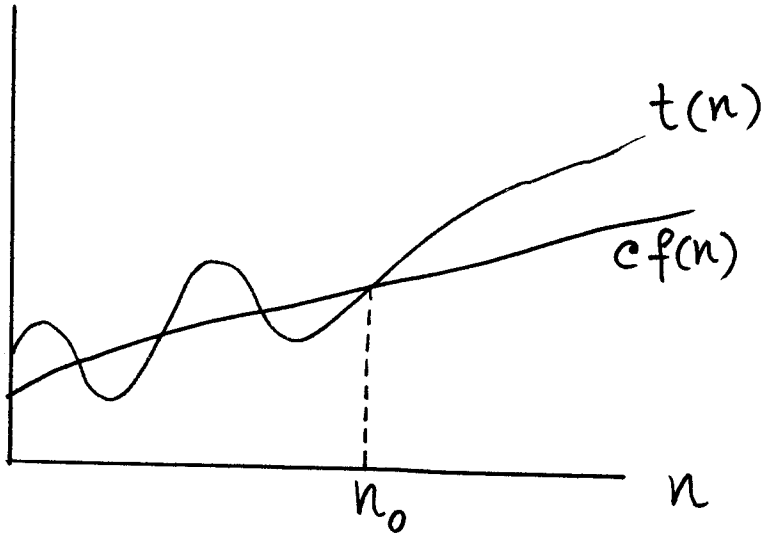
$\Omega(f(n)) = \{ t(n)$: there exist positive constants c and n_0 such that $0 \leq c \cdot f(n) \leq t(n)$ for all $n \geq n_0 \}$

M.V.

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

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

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



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

Θ - notation

"asymptotically tight bound"

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

ಉದಾ.

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

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

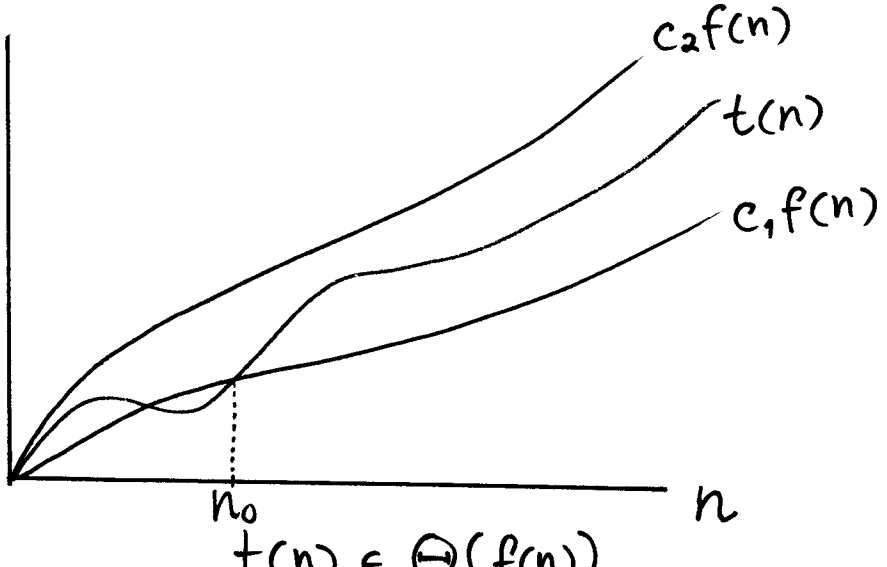
$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$$

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

① $\frac{1}{2} - \frac{3}{n} \leq c_2$ ಸಾಧಿಸಲು ಸಾಧ್ಯ $n \geq 1$ ಇದ್ದಲ್ಲಿ $c_2 \geq \frac{1}{2}$.

② $c_1 \leq \frac{1}{2} - \frac{3}{n}$ ——— $n \geq 7$ ಇದ್ದಲ್ಲಿ $c_1 \leq \frac{1}{14}$.

∴ ಇಲ್ಲಿ $c_1 = \frac{1}{14}$, $c_2 = \frac{1}{2}$, ಮತ್ತು $n_0 = 7$



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

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

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

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

e.g. $2n \in o(n^2)$

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

ω-notation

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

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

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

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

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

② $\sum_{i=1}^{n} i^k \geq \sum_{i=\lceil \frac{n}{2} \rceil}^{n} i^k$

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

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

ต.บ. n^b เทียบกับ a^n a, b - real constants $a > 1$

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

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

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

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

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

ตย. $\lg^b n$ เทียบกับ n^c

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

(b ตัว)

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

แทน n ด้วย $\lg n$

แทน a ด้วย 2^c

ได้ $\lim_{n\to\infty} \frac{\lg^b n}{(2^c)^{\lg n}} = \frac{\lg^b n}{n^{\lg 2^c}} = \frac{\lg^b n}{n^c} = 0$

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

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

---

① $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$ $O(f(n)+g(n)) = O(\max(f(n), g(n)))$

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

③ $\log n! \in \Theta(n \log n)$

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

④ $O(1)$

⑤ 2^n, n^n, $n!$

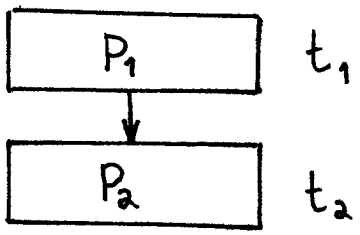
⑥ $2^{n+1} \in O(2^n)$?

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

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 → $\sum_{i=1}^{m} t_i$

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

- "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

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

$$d_0 = n$$

ดังนั้น $d_k \leq n/2^k$

while loop จบเมื่อ $d \leq 1$ $\Big\}$ $k \leq \lceil \lg n \rceil$

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

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

ตย. Euclid's algorithm

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 $n \geq m$ จะพิสูจน์ว่า $n \bmod m < n/2$ เสมอ.

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

$$\therefore n \bmod m = n - m \times (n \div m)$$
$$= n - m$$
$$< n - n/2 = n/2$$

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

```
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

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

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

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

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 $A[i] > key$"

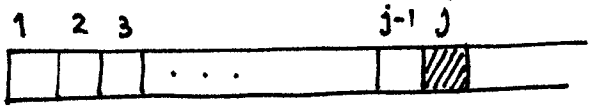
worst case : $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1 = \Theta(n^2)$

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 ทำงานกี่ครั้ง "$i > 0$ and $A[i] > key$"

ที่เป็นไปได้ 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

```
for i ← 1 to n do P
```

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

$\therefore$ ใช้เวลาทั้งหมด $O(n \log n)$ สำหรับ วงวน for นี้.

แต่ อาจใช้เวลา เร็วกว่า ใน worst case

ต.ย. Binary counter : Increment

b_3	b_2	b_1	b_0		
0	0	0	0		0
				1	
0	0	0	1		1
				2	
0	0	1	0		3
				1	
0	0	1	1		4
				3	
0	1	0	0		7
				1	
0	1	0	1		8
				2	
0	1	1	0		10
				1	
0	1	1	1		11
				4	
1	0	0	0		15
				1	
1	0	0	1		16
				2	
1	0	1	0		18
				1	
1	0	1	1		19
				3	
1	1	0	0		22
				1	
1	1	0	1		23
				2	
1	1	1	0		25
				1	
1	1	1	1		26
				5	
1	0	0	0	0	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 " $\lfloor n/4 \rfloor$ "

  ⋮

  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
```

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

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

```
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 )
    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^n

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

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

Solving Recurrences

Master method

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

---

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

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

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

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

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

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

M.U. $T(n) = 3T(n/4) + n\lg n$

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

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

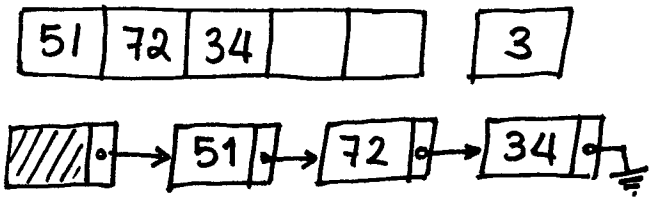
Data Structures

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

Lists

Implementations :

- Contiguous
- linked



ตัวอย่าง Contiguous + Sorted (ascending)

Find :	Binary search	: $O(\log n)$
Insert :	ต้องดันข้อมูล	: $O(n)$
Delete :	" — "	: $O(n)$
FindMin:	ตัวแรกน่อยสุด	: $O(1)$

ต.ย. Contiguous + unordered

Find : Sequential Search : $O(n)$

Insert : ต่อท้าย : $O(1)$

Delete : เอาตัวท้ายมาแทนตัวที่ลบ : $O(1)$

FindMax : ต้องวิ่งเปรียบเทียบทุกตัว : $\Theta(n)$

ต.ย. Linked list + ordered

Find : Sequential search : $O(n)$

Insert : เมื่อรู้ตำแหน่ง ก็เปลี่ยนเปลี่ยน pointers : $O(1)$

Delete : " " : $O(1)$

FindMax : ต้องวิ่งเปรียบเทียบทุกตัว : $\Theta(n)$

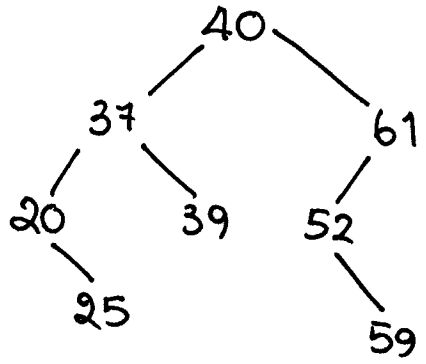
ต.ย.

Stack		Queue	
Push :	$O(1)$	Enqueue :	$O(1)$
Pop :	$O(1)$	Dequeue :	$O(1)$

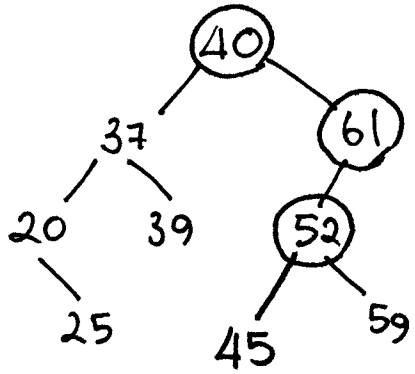
สรุป จะมี บาง operation ของ list ที่เป็น $O(n)$ (Find, Insert, Delete)

Trees

Binary Search Trees



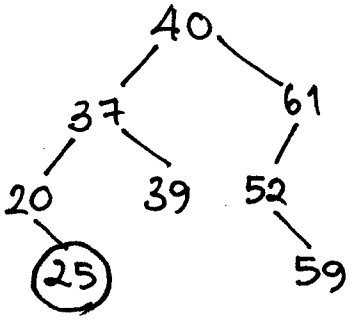
- Find : $O(h)$
- Insert : แทรก 45



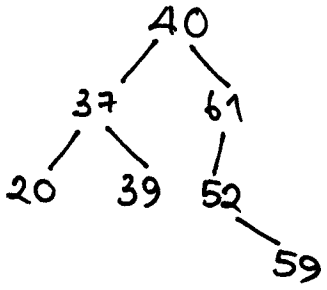
$O(h)$

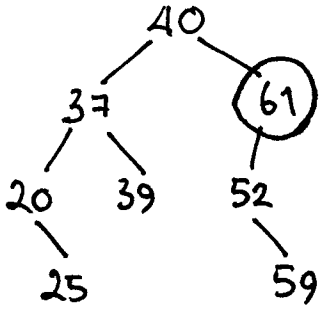
- Delete
  - ลบใบ
  - ลบ node ที่มี 1 ลูก
  - ลบ node ที่มี 2 ลูก

Delete O(h)

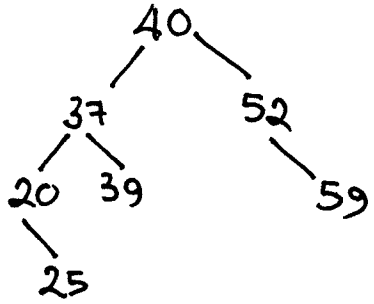


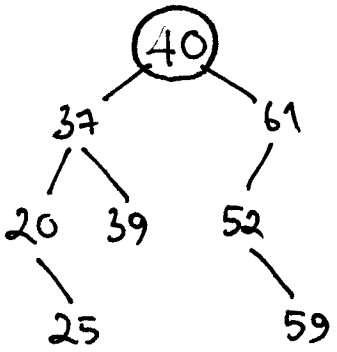
ลบ 25



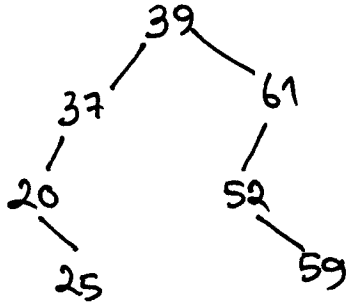


ลบ 61

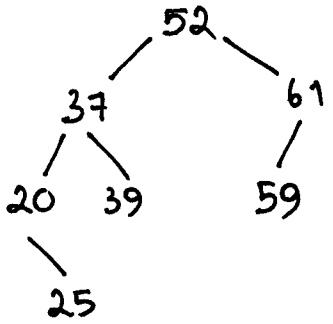




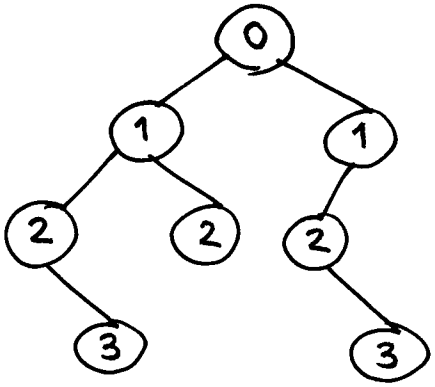
ลบ 40



หรือ



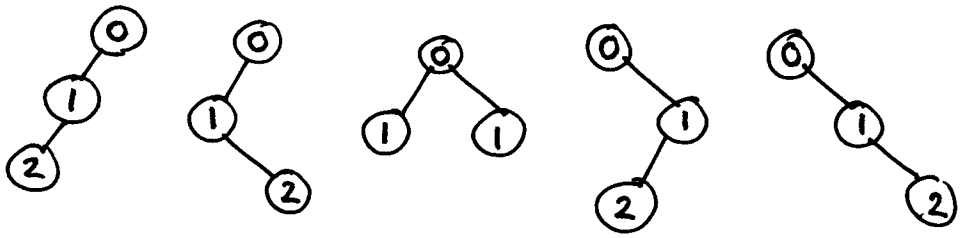
ความลึกของ node ใน BST



ความลึกเฉลี่ยของ node ของต้นไม้รูปนี้ $= \frac{0+1+1+2+2+2+3+3}{8}$

$= \frac{14}{8} = 1\frac{6}{8}$

BST ที่มี 3 nodes



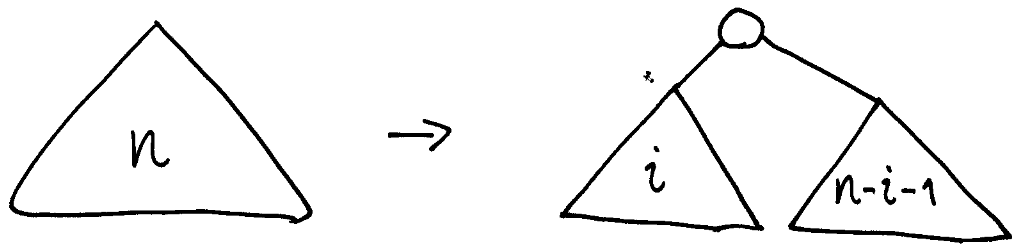
$\frac{0+1+2}{3}$ $\frac{0+1+2}{3}$ $\frac{0+1+1}{3}$ $\frac{0+1+2}{3}$ $\frac{0+1+2}{3}$

ความลึกเฉลี่ยของ node ใน BST ที่มี 3 nodes ต้นใดๆ $= \frac{1+1+\frac{2}{3}+1+1}{5}$

$= \frac{14}{15}$

Expected Depth of node in BST of n nodes

ให้ $D(n)$ = ผลรวมของความลึกของ node ใน BST ที่มี n nodes



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

กำหนดให้ ~~โอกาสที่~~ ต้นไม้ย่อยจะมีขนาดเท่าใดก็ดี มีโอกาสพอๆกัน

$$D(n) = \frac{1}{n}\sum_{i=0}^{n-1}\left[D(i) + D(n-i-1) + n-1\right]$$

$$= \left(\frac{1}{n}\sum_{i=0}^{n-1} D(i)\right) + \left(\frac{1}{n}\sum_{i=0}^{n-1} D(n-i-1)\right) + n-1$$

$$= \left(\frac{2}{n}\sum_{i=0}^{n-1} D(i)\right) + n-1$$

$$= O(n\log n)$$

$\therefore$ Expected depth $= O(\log n)$

Hash Tables

List:

0	1	2	3	4
7	11	5		

3

Table :①

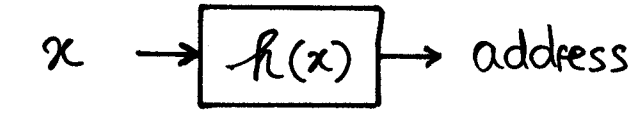
0	1	2	3	4	5	6	7	8	9	10	11
					✓		✓				✓

:②

0	1	2	3	4	5	6
			5	7		11

:③

0	1	2	3
7		11	5



① $h(x) = x$

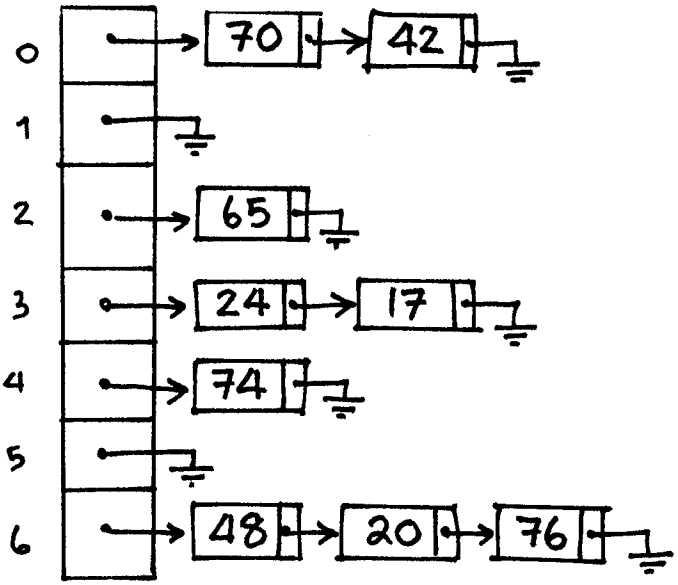
② $h(x) = \lceil \frac{x}{2} \rceil$

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

- hashing function
- Collision resolution

Separate Chaining

ตย. $h(x) = x \bmod 7$



T : Table size

N : # data

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

ถ้า hash function ดี → ข้อมูลจะกระจายลงช่องต่างๆ พอๆกัน

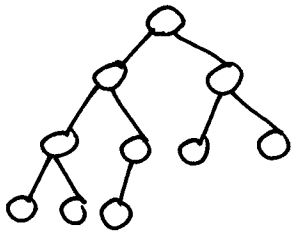
ความยาวของแต่ละ list = λ

Find, Delete → $O(\lambda)$

Insert → $O(1)$

Heapsort

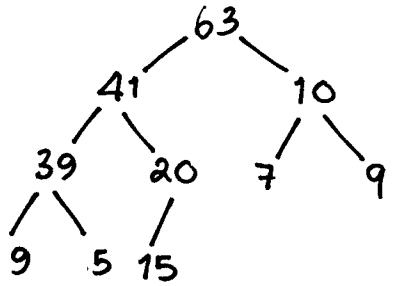
- Binary heap
  - structure property



n nodes

$h = \lfloor \lg n \rfloor$

  - Heap order property



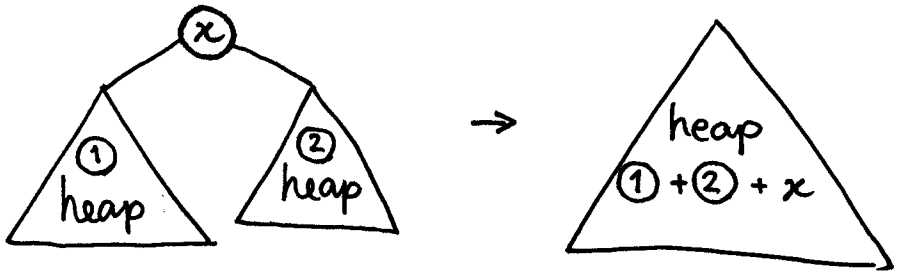
  - Array implementation

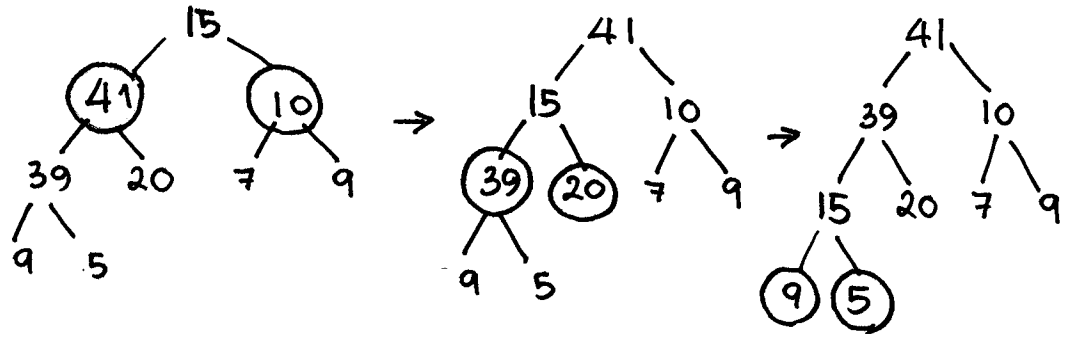
10

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

- root อยู่ที่ index 1
- ลูกทางซ้ายของ node ที่อยู่ที่ index k เก็บอยู่ที่ index $2k$
- " ขวา " " k " $2k+1$
- พ่อ/แม่ " " k " $\lfloor k/2 \rfloor$

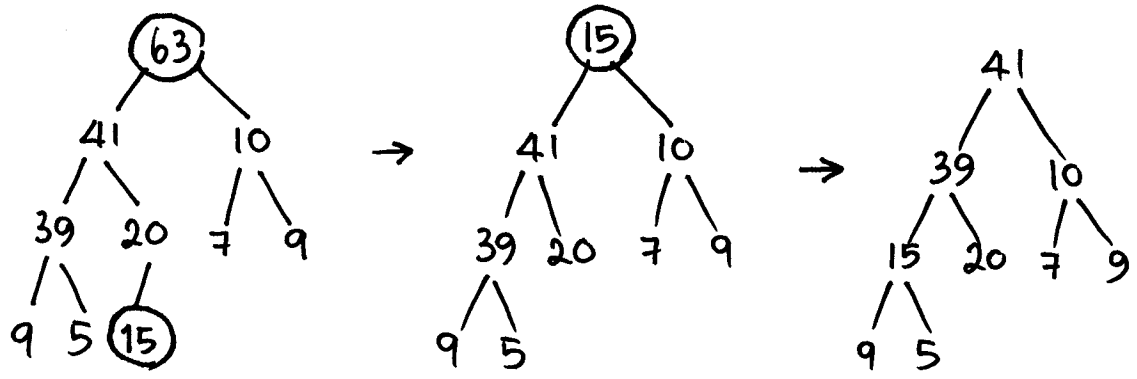
Heapify





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

DeleteMax



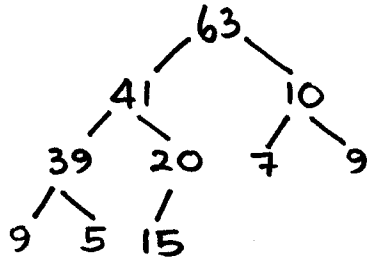
$O(\lg n)$

Heap sort

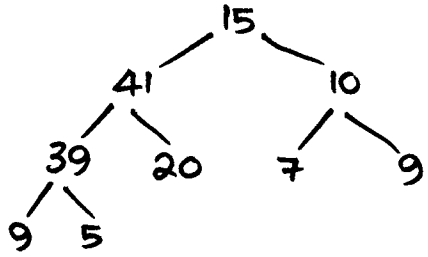
	1	2	3	4	5	6	7	8	9	10
10	9	15	63	5	20	7	10	9	39	41

"BuildHeap"

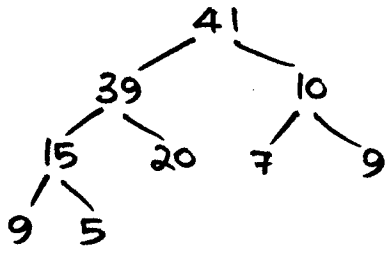
10	63	41	10	39	20	7	9	9	5	15



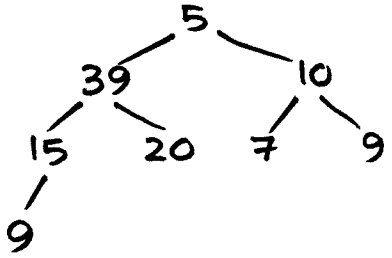
9	15	41	10	39	20	7	9	9	5	63



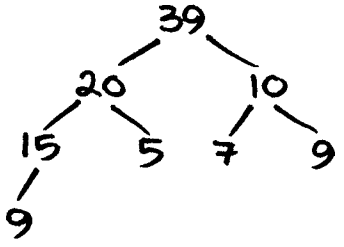
9	41	39	10	15	20	7	9	9	5	63



8	5	39	10	15	20	7	9	9	41	63

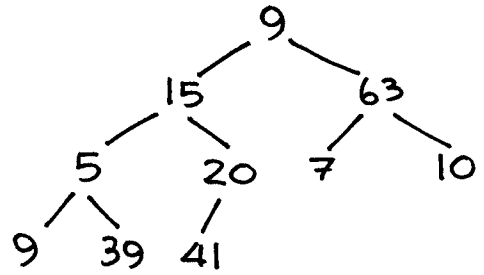


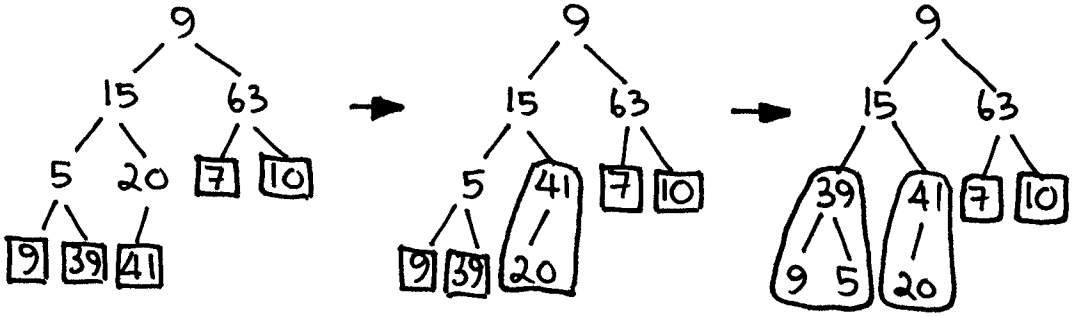
8	39	20	10	15	5	7	9	9	41	63

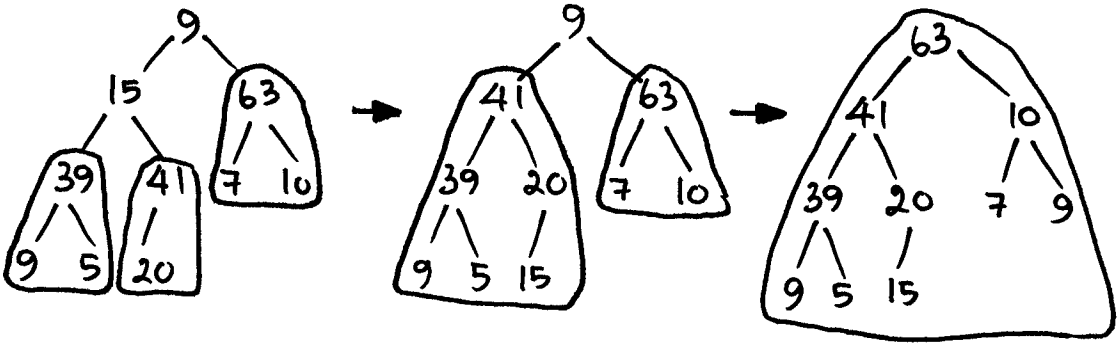


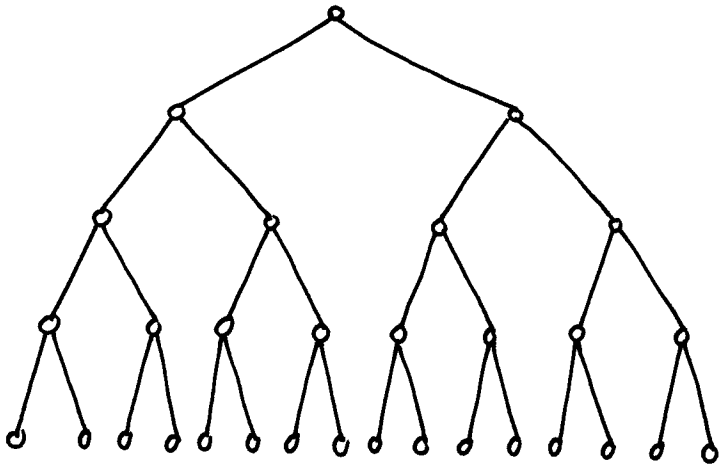
BuildHeap

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









height	#trees
4	1
3	2
2	4
1	8
0	16

ถ้ามี n nodes ต้นไม้มีความสูง $\lfloor \lg n \rfloor$

∴ มีต้นไม้ที่สูง $\lfloor \lg n \rfloor$ จำนวน 1 ต้น

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

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

⋮ ⋮ ⋮

" 2 " $2^{\lfloor \lg n \rfloor - 2}$ "

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

" 0 " $2^{\lfloor \lg n \rfloor}$

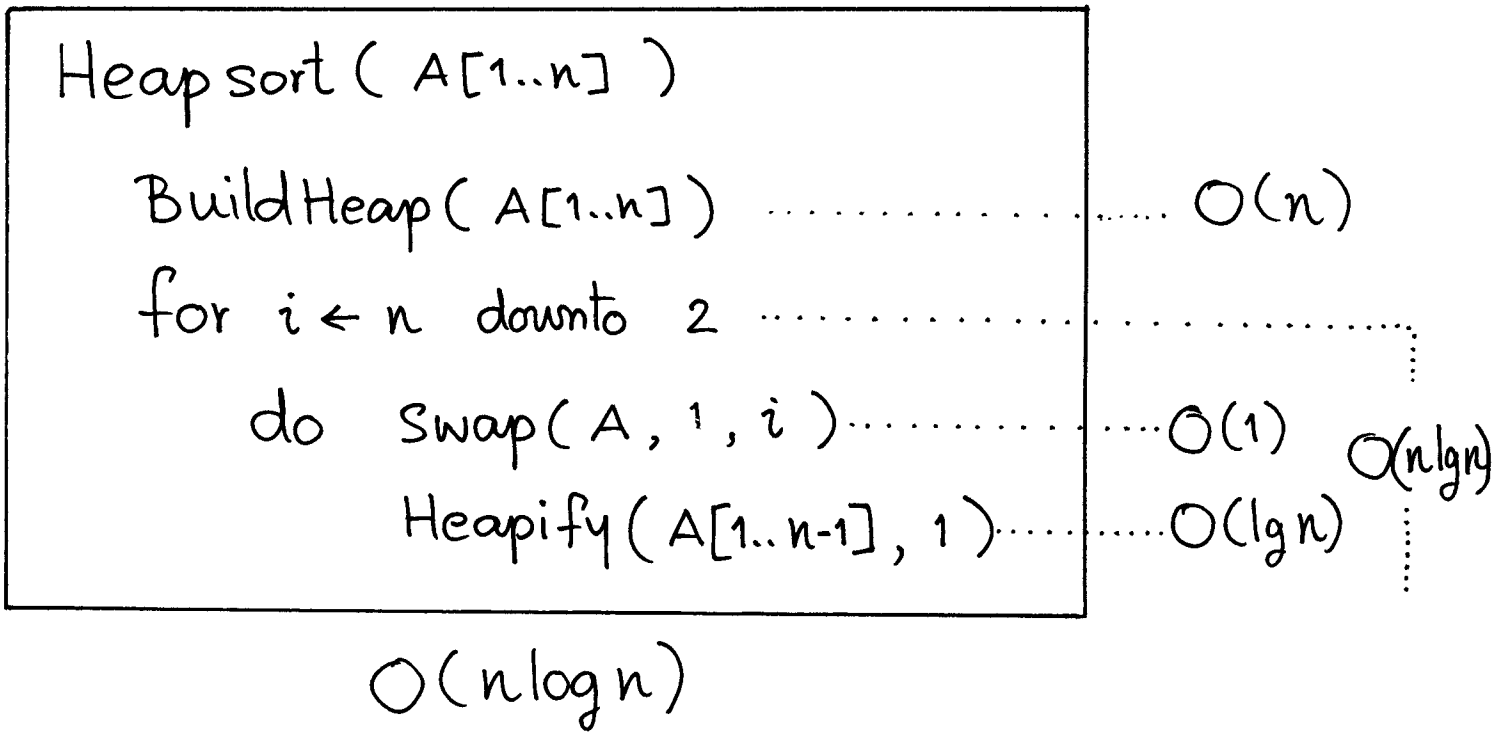
$$\therefore \text{เวลา} \Rightarrow \sum_{h=0}^{\lfloor \lg n \rfloor} \left[\left(2^{\lfloor \lg n \rfloor - h} \right) \cdot O(h) \right]$$

$$= \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{n}{2^h} \cdot O(h)$$

$$= O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right)$$

$$= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) = O(2n) = O(n)$$

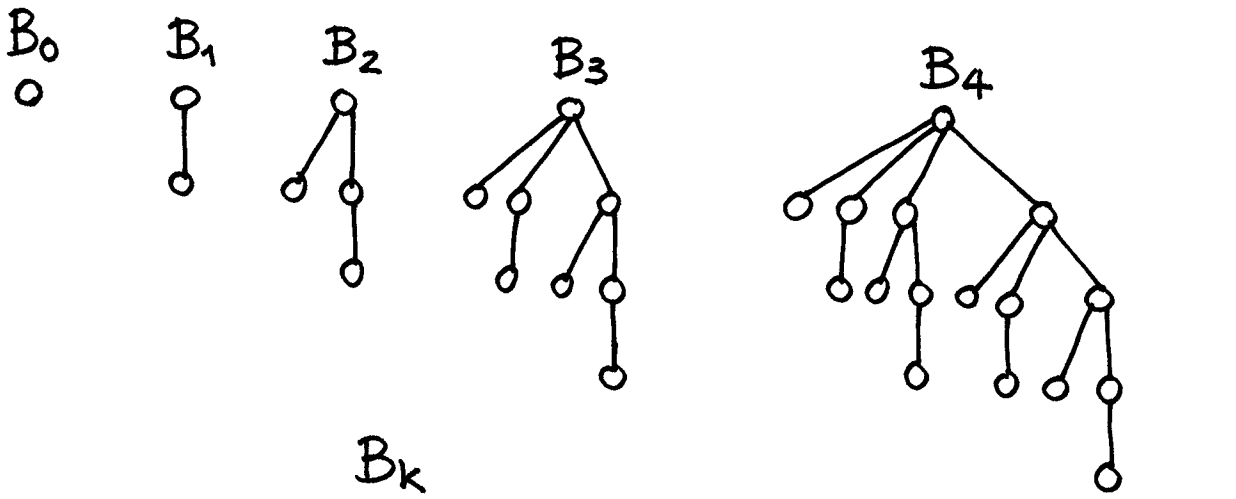
$$\sum_{h=0}^{\infty} h x^h = \frac{x}{(1-x)^2}$$

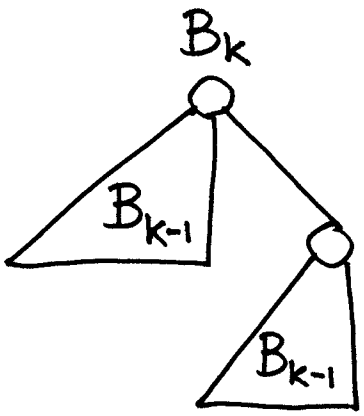

Heapsort (A[1..n])
BuildHeap (A[1..n])
O(n)
for i ← n downto 2
do Swap (A , 1 , i)
O(1)
Heapify (A[1..n-1] , 1)
O(lg n)
O(n lg n)
O(n log n)

Binomial Heaps

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

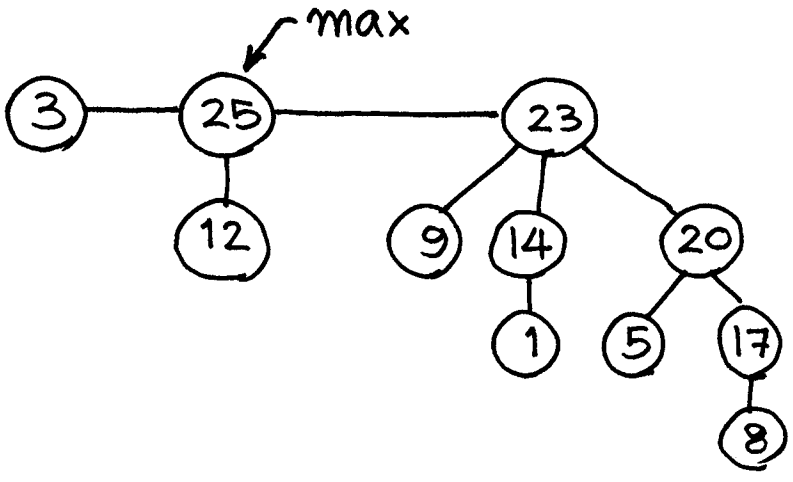
- Binomial trees





B_k has 2^k nodes

- Binomial heaps



$n = 11$

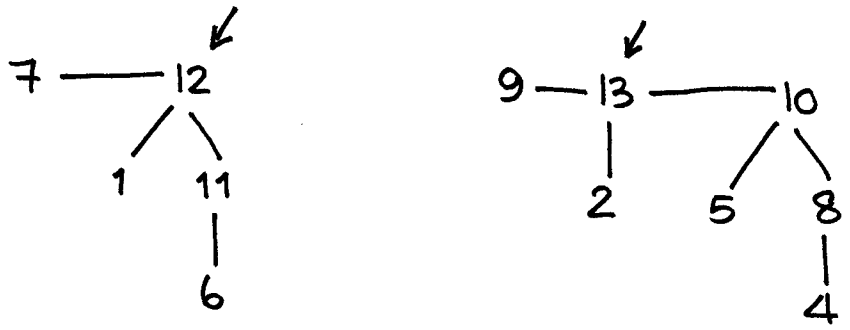
B_0, B_1, B_3

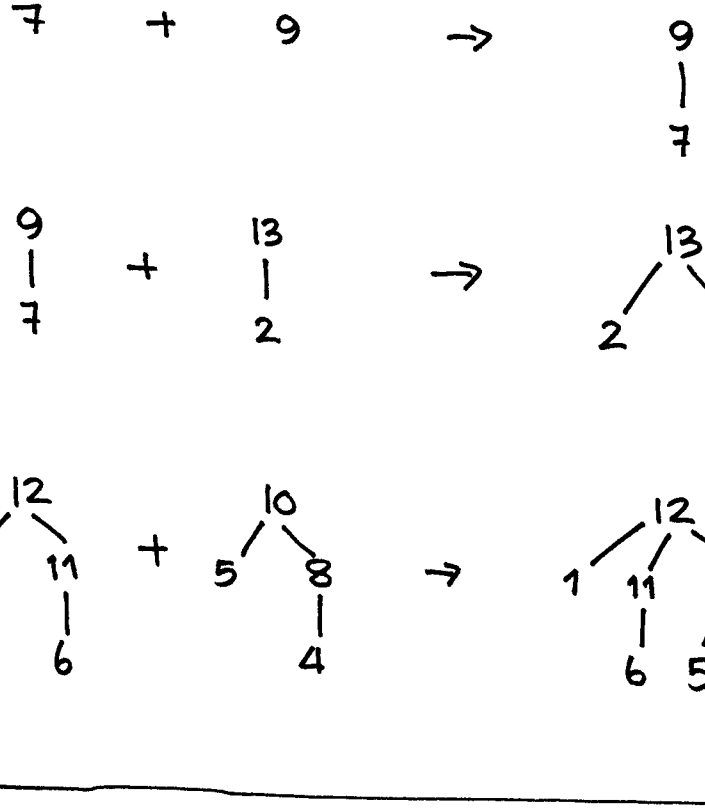
$11 = 2^0 + 2^1 + 2^3$

$= (1011)_2$

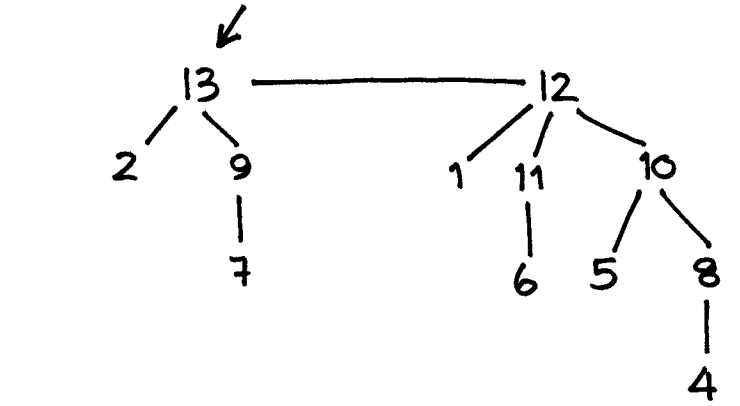
$B_3\ B_1\ B_0$

- merge





B_3	B_2	B_1	B_0	
0	1	0	1	
0	1	1	1	+
1	1	0	0	



- Binomial heap มี n nodes

  มี binomial trees อย่างมาก $\lfloor \lg n \rfloor + 1$ ต้น.

ต.ย. $n = 25$ $\quad 25_{10} = (11001)_2$

$= 2^4 + 2^3 + 2^0$

∴ มี 25 nodes heap : B_4, B_3, และ B_0

∴ ใช้ 25_{10} ต่อกัน 5 ปีก = $\lfloor \lg 25 \rfloor + 1$

$= 4 + 1$

- merge binomial heaps

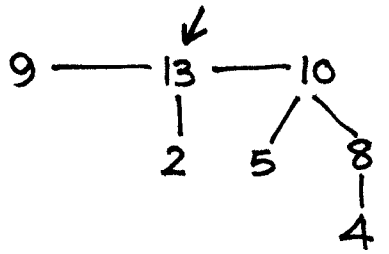
  - เชื่อมบวกเลขฐานสอง ใช้เวลา $O($จำนวนปีก$)$
  - ใช้เวลา $O($จำนวนต้น$) \rightarrow O(\log n)$

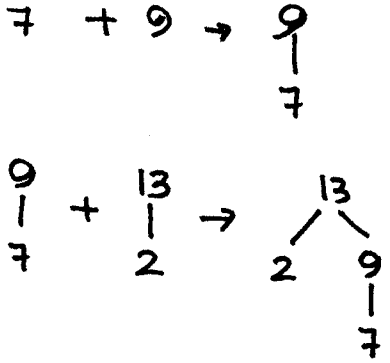
- Find Max

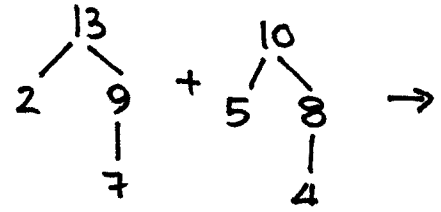
  - ดูที่ node ที่ max ก็

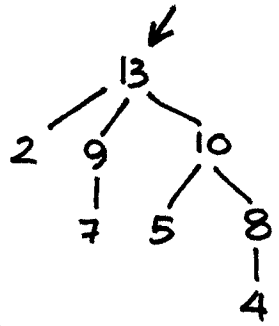
$$O(1)$$

- Insert
  - เปลี่ยนข้อมูลใหม่เป็น binomial heap ต้นเล็กๆ ที่มีแค่ B_0 (มีข้อมูลตัวเดียว)
  - merge heap ใหม่นี้ กับ heap เดิมที่มีอยู่.
  - ตย. เพิ่ม 7 ใน



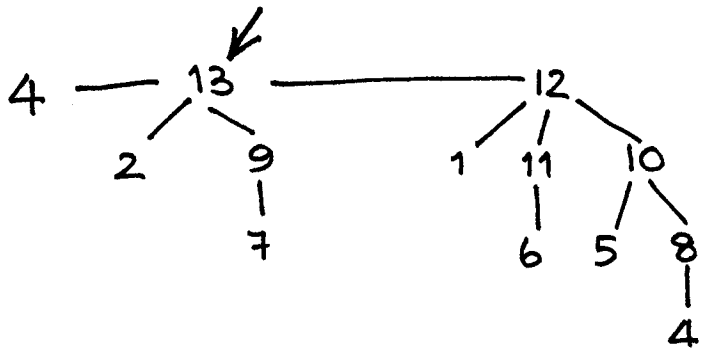


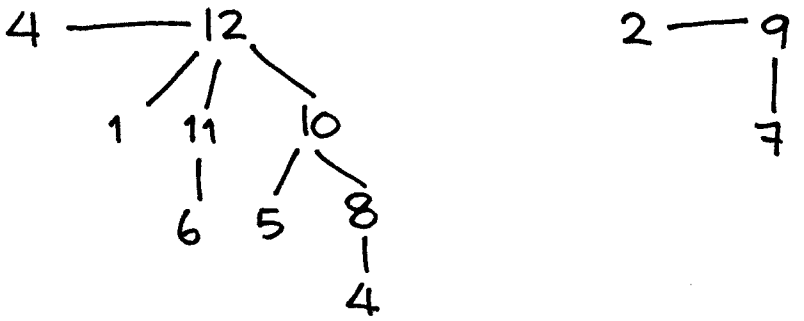




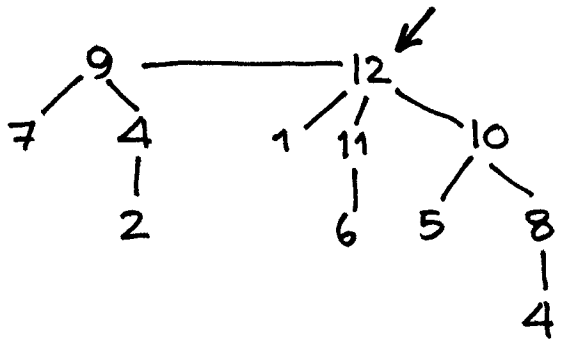
  - Insert ใช้เวลา เหมือน เพิ่ม เลขฐานสอง อีก 1
    - เหมือน บวกเลข → $O(\log n)$
    - 111111 + 1 ช้า
      111100 + 1 เร็ว.
    - เหมือน increment ใน binary counter amortized cost $O(1)$

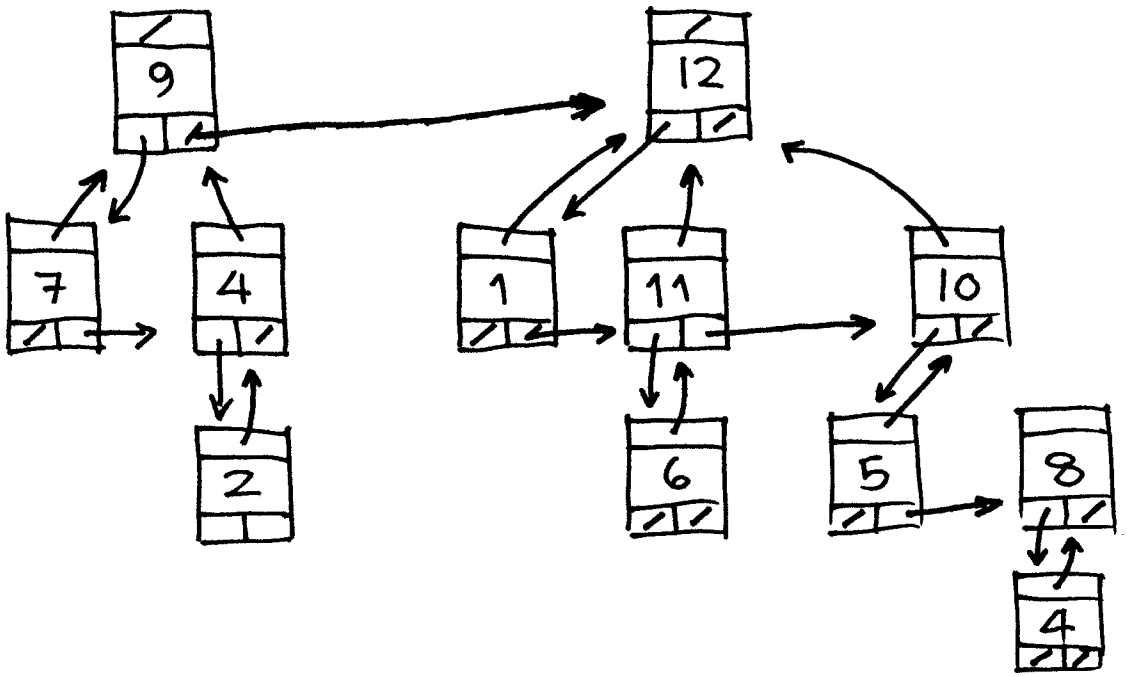
- DeleteMax





$O(\log n)$

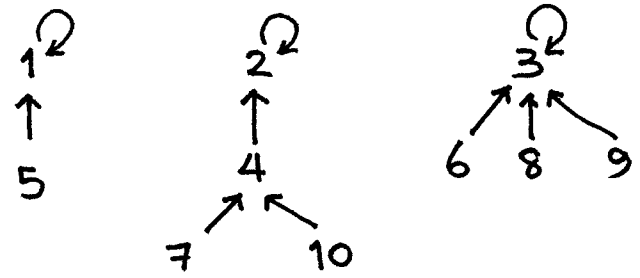




Disjoint Set

- มีข้อมูลเป็นตัวเลข 1 ถึง N
- ต้องการรวมข้อมูลเหล่านี้เป็น sets
- operations
  - find : หาหมายเลข set ที่ข้อมูลเป็นสมาชิก
  - union : รวม set 2 sets

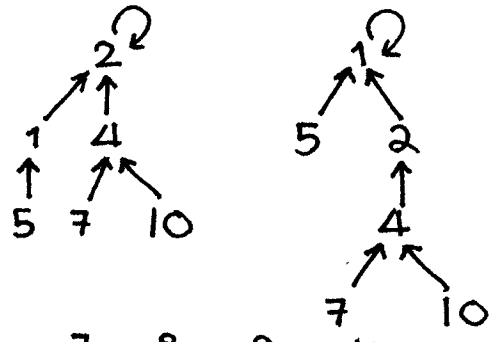
Implementation



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

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

{1,5} U {2,4,7,10}



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

- Union-by-height

  - ต้นไม้ที่มี k nodes มีความสูงอย่างมาก $\lfloor \lg k \rfloor$

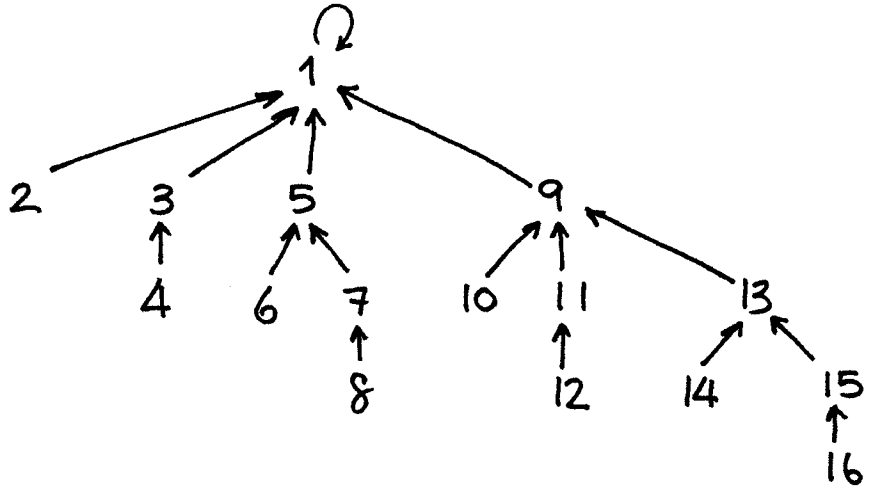
```
function find(x)
    if( s[x] <= 0)
        return x;
    else
        return( find( s[x]));
```

$O(\log n)$

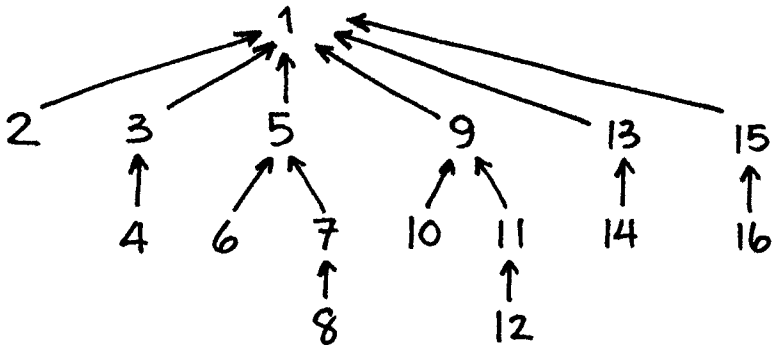
```
Union( a , b )
    if( |s[a]| < |s[b]| )
        s[a] = b;
    else {
      if (s[a] == s[b] )
        s[a] -- ;
      s[b] = a;
    }
```

$O(1)$

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 $O(m \log^* n)$ running time

---

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

  $\log_2^* \underbrace{2^{2^{\cdots^2}}}_{k} = k$

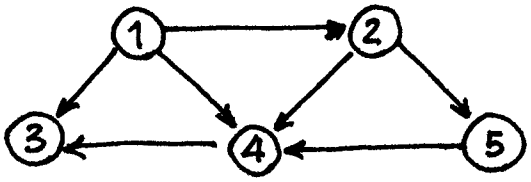
  $\log_2^* 2^{2^{2^{2^2}}} = 5$; $2^{2^{2^{2^2}}}$ is a 20,000 digit number!

  $O(m \log^* n)$ ประมาณได้ว่า $O(m)$

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

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

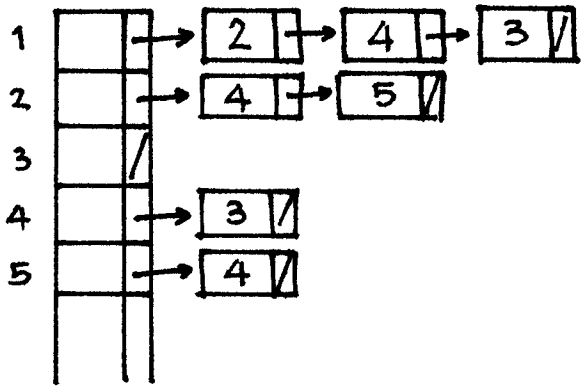
Graphs



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

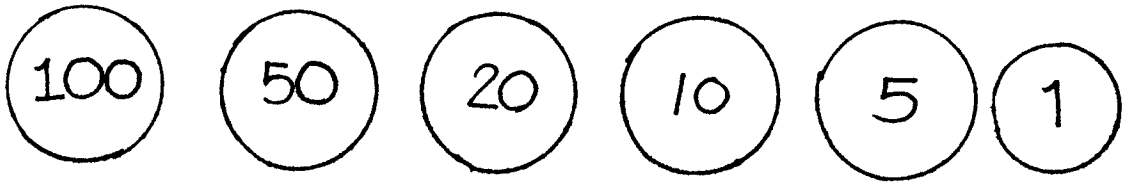


Greedy Algorithms

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

Making Changes

- จ่ายเหรียญ จำนวนน้อยที่สุดในการแลกเงิน.



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

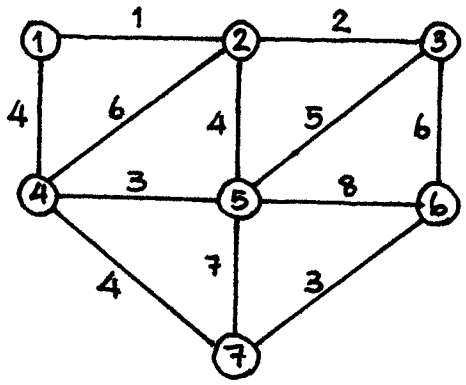
6 เหรียญ

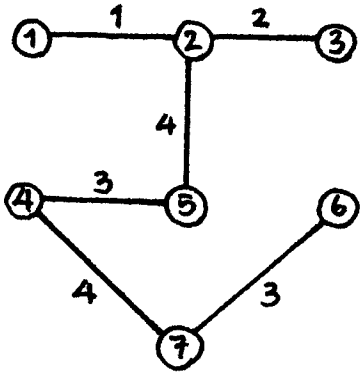
```
function Greedy (C : set) : set
    S ← ∅
    while C ≠ ∅ and not Solution(S) do
        x ← Select(C)
        C ← C - {x}
        if feasible(S ∪ {x}) then S ← S ∪ {x}
    if solution(S) then return S
                   else return "no solution"
```

C - candidate set

S - solution set

Minimum Spanning Trees

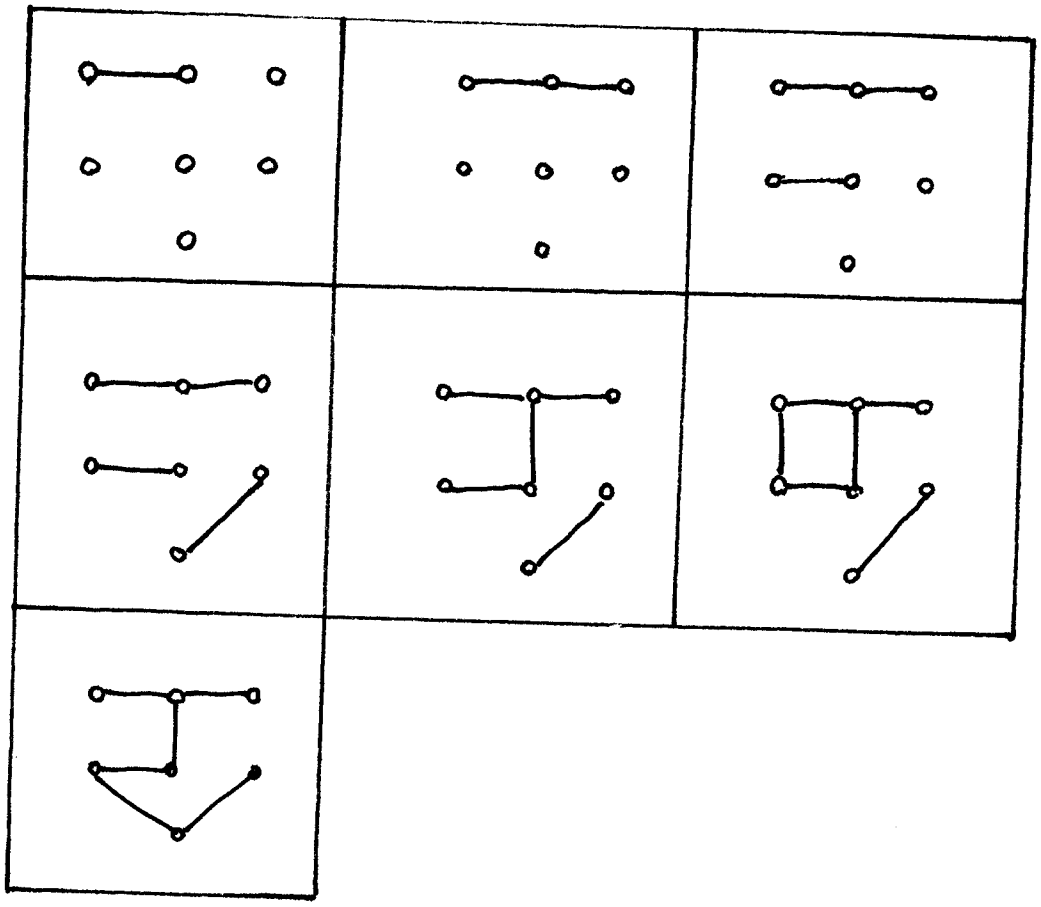




Kruskal's algorithm

- เลือก edge สั้นสุด
- รวม edge นี้ ถ้าไม่เกิดวงจร

เลือกได้ n-1 edges



```
Kruskal ( G , w )
    T ← ∅
    for each vertex v ∈ V(G) do
        MAke_Set (v)
    Sort the edges of E(G) by nondecreasing
                                weight w
    for each edge (u,v) ∈ E(G)
        in order by nondecreasing weight do
        x ← find (u)
        y ← find (v)
        if x ≠ y then
            Union (x, y)
            T ← T U { (u,v) }
    return T
```

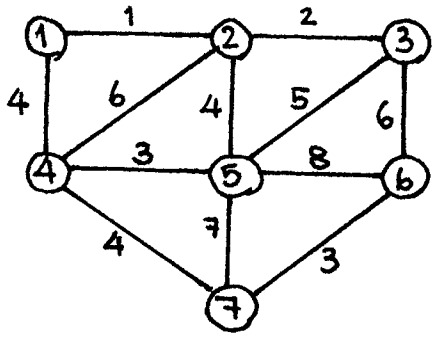
Sort : $O(|E| \log |E|) \rightarrow O(|E| \log |V|)$

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

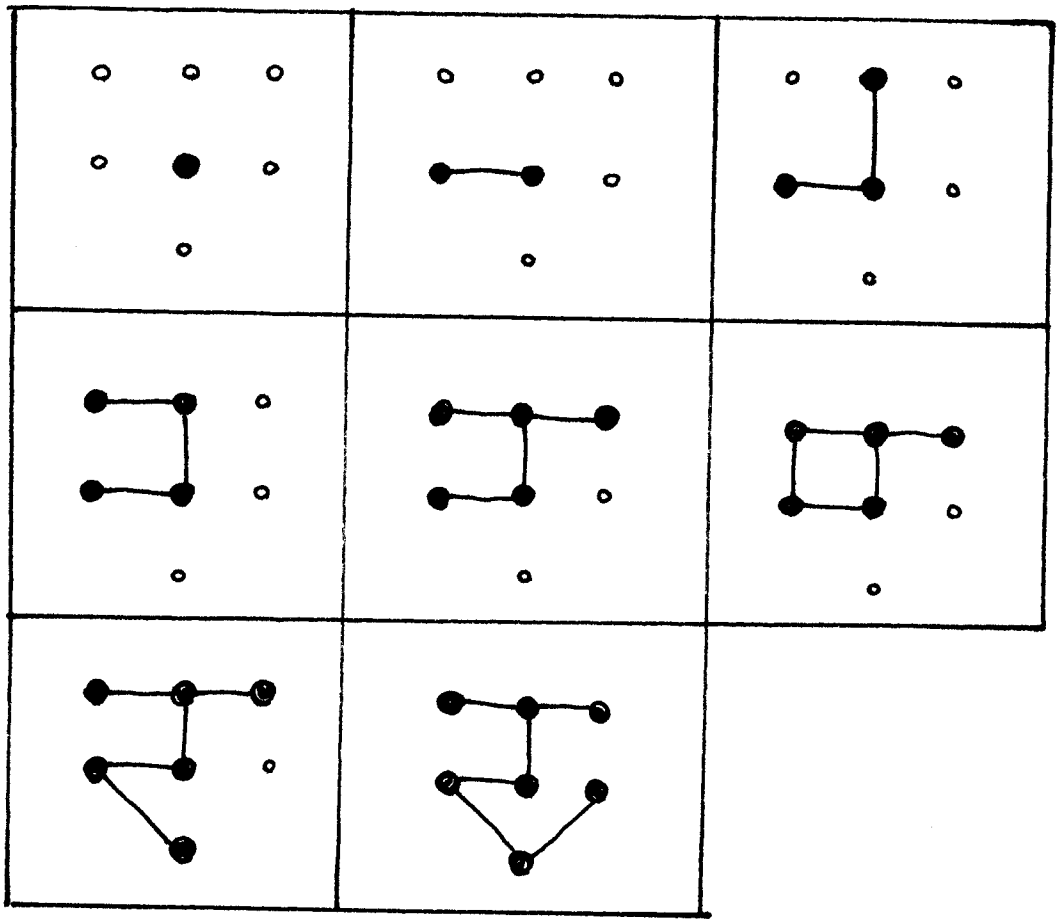
At most $2|E|$ find operations
$|V|-1$ Union " — " $\Big\}\ O(|E| \log^* |V|)$

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

Prim's algorithm



- เลือก 1 จุด
- เลือก edge สั้นสุดที่ต่อกับที่ได้เลือกไว้ ← ครบทุก node
- รวม edge นี้ ถ้าไม่เกิดวงจร

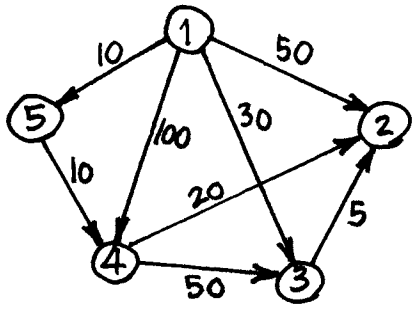


```
Prim( G, ω )
    Q ← V(G) /* priority queue */
    for each u ∈ Q do key[u] ← ∞
    r ← an arbitary node in V(G)
    key[r] ← 0
    π[r] ← NIL
    while Q ≠ ∅ do
        u ← DeleteMin( Q )
        for each v ∈ Adj[u] do
            if v ∈ Q and ω(u,v) < key[v]
                then π[v] ← u
                     key[v] ← ω(u,v)
```

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

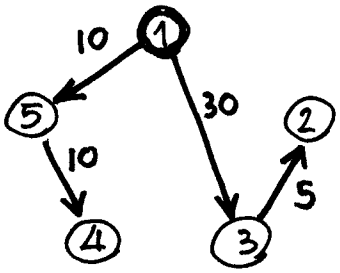
$\therefore$ $O(|E| \log |V|)$

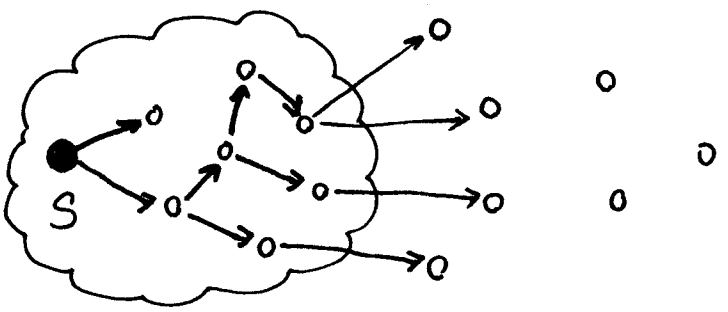
Shortest Paths (Single Source)



รูปที่ 10.1

- หาเส้นทางจาก source node ไปยัง node อื่นๆ ที่มีระยะสั้นสุด.





Dijkstra's Algorithm

```
Dijkstra( G, w, s )
    for each vertex v ∈ V(G) do
        d[v] ← ∞
        π[v] ← NIL
    d[s] ← 0;
    ST ← ∅
    Q ← V(G)
    while Q ≠ ∅ do
        u ← DeleteMin(Q)
        ST ← ST ∪ {u}
        for each vertex v ∈ Adj(u) do
            if d[v] > d[u] + w(u,v) then
                d[v] ← d[u] + w(u,v)
                π[v] ← 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

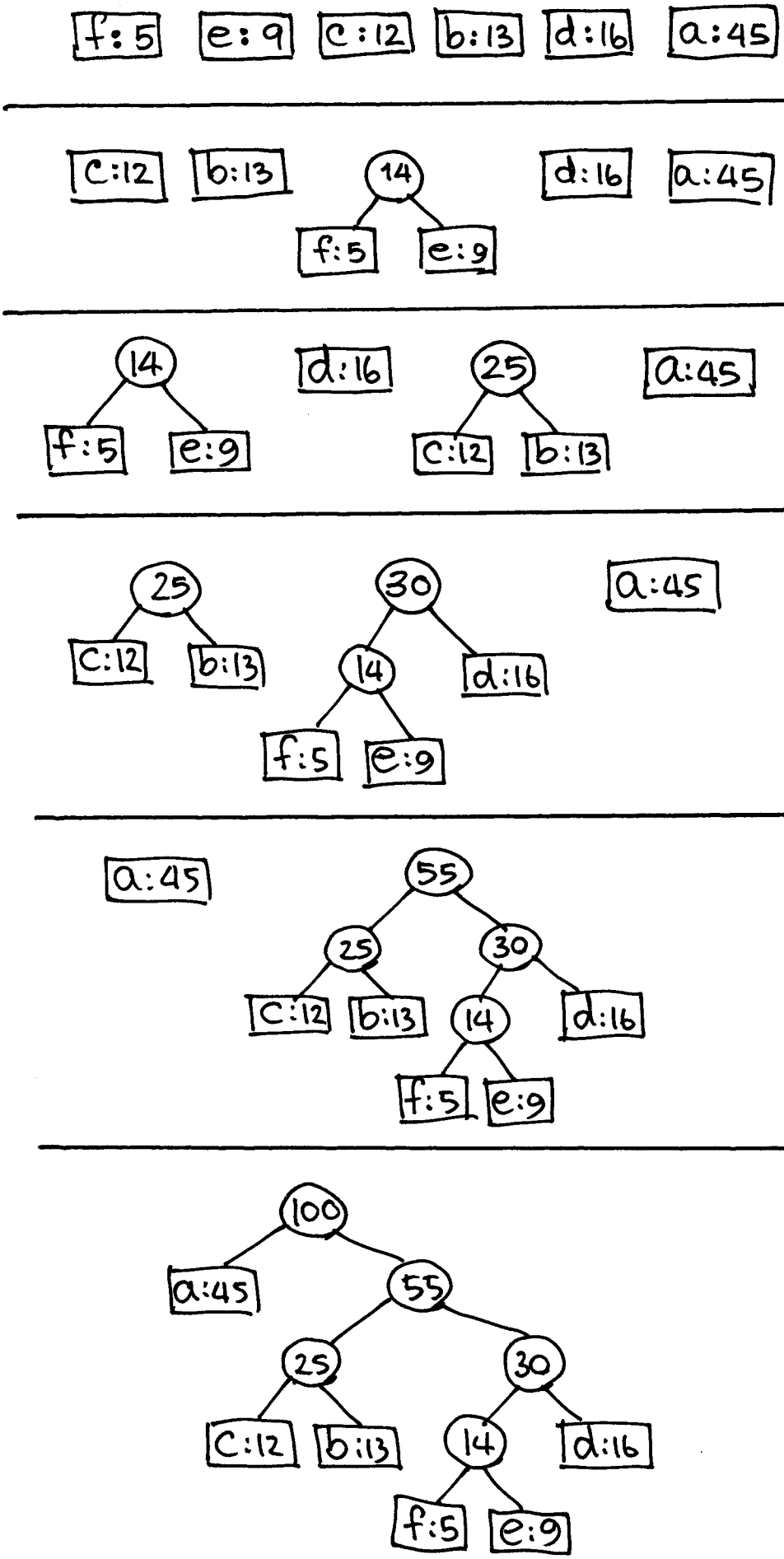
① ใช้กี่บิต 300,000 bits

② " 224,000 "

Prefix code :

no codeword is a prefix of some other codeword

Huffman code ⇒ optimal prefix code


f:5
e:9
c:12
b:13
d:16
a:45
c:12
b:13
14
f:5
e:9
d:16
a:45
14
f:5
e:9
d:16
25
c:12
b:13
a:45
25
c:12
b:13
30
14
f:5
e:9
d:16
a:45
a:45
55
25
30
c:12
b:13
14
d:16
f:5
e:9
100
a:45
55
25
30
c:12
b:13
14
d:16
f:5
e:9

```
Huffman( C )
    n ← |C|
    Q ← C
    for i ← 1 to n-1 do
        z ← Allocate_Node()
        x ← left[z] ← DeleteMin(Q)
        y ← right[z] ← DeleteMin(Q)
        f[z] ← f[x] + f[y]
        Insert(Q, z)
    return DeleteMin(Q)
```

BuildHeap : $O(n)$

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

$\therefore \quad O(n \log n)$

Knapsack

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

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

subject to

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

$$\left.\begin{array}{l} 0 \leq x_i \leq 1 \\ v_i > 0 \\ w_i > 0 \end{array}\right\} \; 1 \leq i \leq n$$

- solution : a vector $(x_1, x_2, \ldots, x_n)$

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 ← 1 to n do x[i] ← 0
    for i ← 1 to n do vw[i] ← v[i]/w[i]
    Sort vw in nondecreasing order
    wght ← 0 ; i ← 1
    while wght < W do
        if wght + w[i] ≤ W then x[i] ← 1
                                wght += w[i]
        else x[i] ← (W - wght) / w[i]
             wght ← W
```

```
knapsack ( w[1..n], v[1..n], W )

    for i ← 1 to n do
        x[i] ← 0
        p[i] ← v[i] / w[i]
    sort p in nondecreasing order
    wght ← 0
    i ← 1
    while (wght < W) do
        if wght + w[i] ≤ W then
            x[i] ← 1
            wght += w[i]
        else
            x[i] ← (W - wght) / w[i]
            wght = W
        i ← i + 1
    return x
```

$O(n \log n)$

Scheduling

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

e.g. $n = 3$, $t_1 = 5$, $t_2 = 10$, $t_3 = 3$

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

- แบ่ง instance ของปัญหา เป็น subinstances จำนวนหนึ่ง ของปัญหานั้น
- หาคำตอบต่างๆ ของ subinstances
- รวมคำตอบย่อยทั้งหลาย เป็น คำตอบของ instance เดิม.

Multiplying large integers

① $0981 \times 1234 = (10^2 \times 09 + 81) \times (10^2 \times 12 + 34)$

$= 10^4 \times 09 \times 12 + 10^2 \times (09 \times 34 + 81 \times 12) + 81 \times 34$

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

$= \Theta(n^2)$

---

② $p = 09 \times 12$, $q = 81 \times 34$

$r = (09 + 81) \times (12 + 34)$

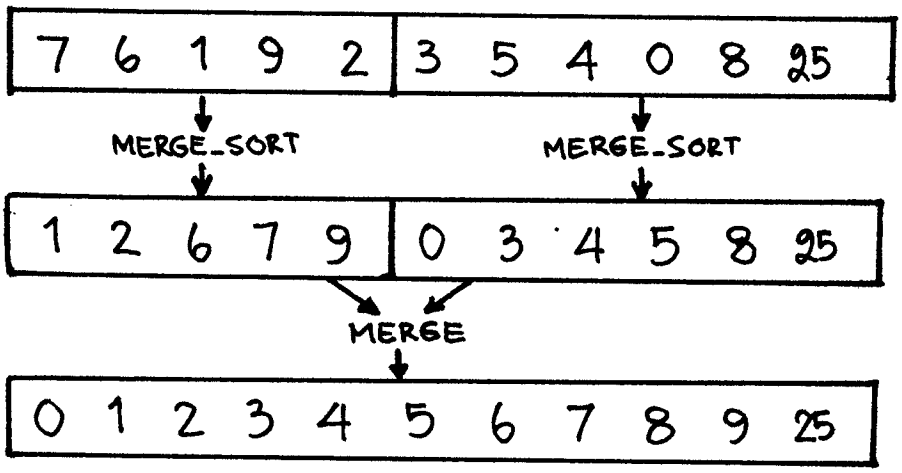
$0981 \times 1234 = 10^4 p + 10^2 (r - p - q) + q$

$t(n) = 3t(n/2) + \Theta(n)$

$= \Theta(n^{\lg 3}) = \Theta(n^{1.585})$

MergeSort

```
MERGE_SORT ( A , p , r )
    if  p < r  then
        q ← ⌊(p+r)/2⌋
        MERGE_SORT ( A , p , q )
        MERGE_SORT ( A , q+1 , r )
        MERGE ( A , p , q , r )
```

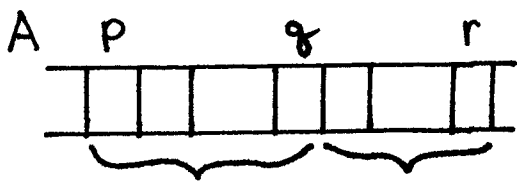


$$t(n) = 2t(n/2) + \Theta(n)$$

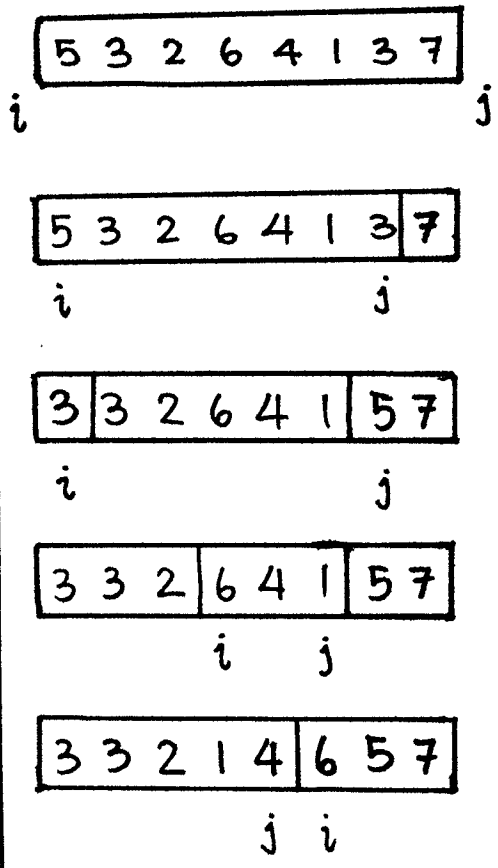
$$= \Theta(n \log n)$$

Quick sort

```
QUICK_SORT ( A, p, r )
    if p < r then
        q ← PARTITION (A, p, r)
        QUICK-SORT ( A, p, q )
        QUICK-SORT ( A, q+1, r )
```




```
PARTITON ( 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.

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

Randomized Quicksort

```
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.

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

Solving the recurrence using substitution method.

เดาว่า $\rightarrow$ $t(n) \leq a n \lg n + b \rightarrow t(1) \leq an \lg n + b$

$$t(n) = \frac{2}{n}\sum_{k=1}^{n-1} t(k) + \Theta(n)$$

$$\leq \frac{2}{n}\sum_{k=1}^{n-1} (ak \lg k + b) + \Theta(n)$$

$$= \frac{2a}{n}\sum_{k=1}^{n-1} k \lg k + \frac{2b}{n}(n-1) + \Theta(n)$$

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

$$t(n) \leq \frac{2a}{n}\left(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n)$$

$$\leq an \lg n - \frac{a}{4}n + 2b + \Theta(n)$$

$$= an \lg n + b + \left(\Theta(n) + b - \frac{a}{4}n\right)$$

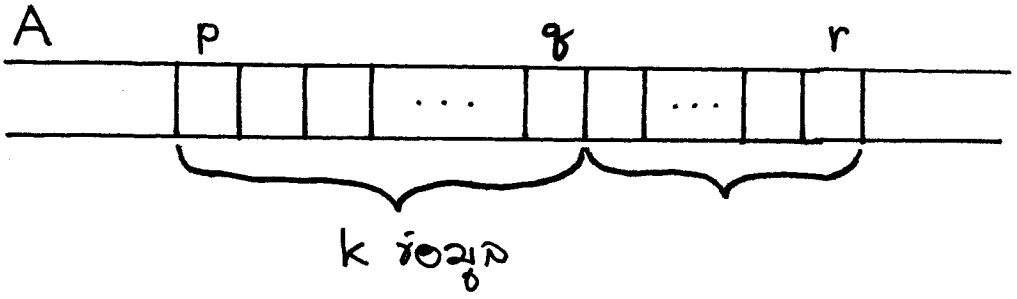
$$\leq an \lg n + b$$

$$= O(n \lg n)$$

เลือก a ที่ใหญ่พอที่ $\frac{an}{4}$ เกิน $\Theta(n)+b$

Selection (find the i^{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)
```



กรณี RAND_PARTITION · → expected linear time

$$t(n) = \frac{1}{n}\left[t(\max(1,n-1)) + \sum_{k=1}^{n-1} t(\max(k,n-k))\right] + O(n)$$

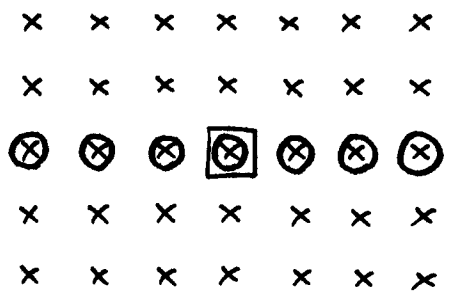
$$\leq \frac{1}{n}\left[t(n-1) + 2\sum_{k=\lceil n/2 \rceil}^{n-1} t(k)\right] + O(n)$$

$$= O(n)$$

ถ้าต้องการ worst-case linear time

- guarantee a good partition
- median of median of five

ตา.บ.

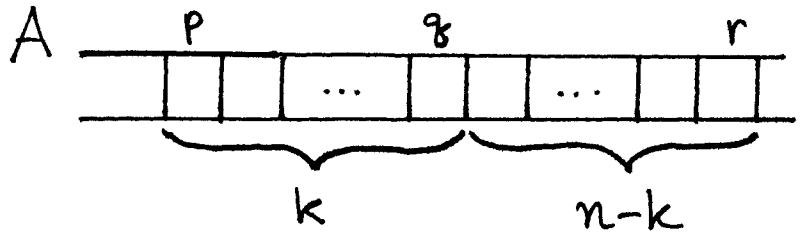


$n = 35$ • แบ่งเป็นกลุ่มๆละ 5 ตัว ได้ 7 กลุ่ม. — $O(n)$

• หา median ของแต่ละกลุ่ม ○ — $O(n)$

• หา median ของ median ต่างๆที่หาได้ ◎ — $t(n/5)$

- มีทั้งหมด $\lfloor n/5 \rfloor$ กลุ่ม
- มี $\frac{3}{2}\lfloor n/5 \rfloor$ ข้อมูล ที่มีค่าน้อยกว่าหรือเท่ากับ mm5
- $\because \lfloor \frac{n}{5} \rfloor \geq \frac{(n-4)}{5}$, $\therefore \frac{3n-12}{10}$
- มีอย่างมาก $n - \left(\frac{3n-12}{10}\right) = \frac{7n+12}{10}$ ข้อมูล ที่มีค่ามากกว่า mm5



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

กรณี mm5 $\quad \max(k, n-k) \leq \frac{7n+12}{10}$

$$t(n) \leq t\left(\frac{7n+12}{10}\right) + t(\text{จำนวน mm5}) + O(n)$$

$$\leq t\left(\frac{7n+12}{10}\right) + t(\lfloor n/5 \rfloor) + O(n)$$

$$= O(n)$$

> 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 $O(n)$

ตัวอย่าง $\quad t(n) = t(0.1n) + t(0.3n) + t(0.59n) + O(n)$

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

Matrix Multiplication

ಇಲ್ಲಿ A and B ಎರಡೂ $n \times n$ matrices

$$C = A \times B$$

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

ಹಾಗೆ c_{ij} ಅಲ್ಲಿ n^2 ಇರುತ್ತೆ

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

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

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

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

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

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

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

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

Strassen (1969)

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

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

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

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

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

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

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

$$C = A \times B$$

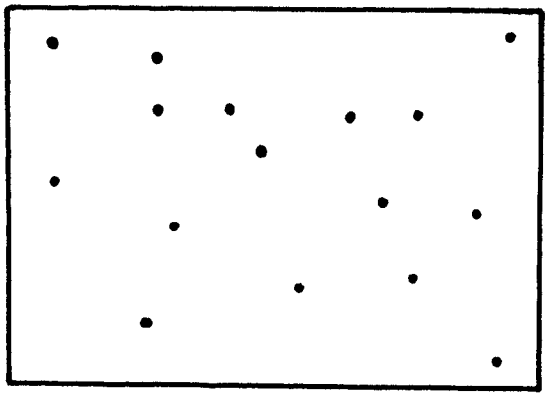
$$= \begin{pmatrix} m_2 + m_3 & m_1 + m_2 + m_5 + m_6 \\ m_1 + m_2 + m_4 - m_7 & m_1 + m_2 + m_4 + m_5 \end{pmatrix}$$

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

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

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

Closest Pair



จุดคู่ใดอยู่ใกล้กันที่สุด?

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

D & Q : ① แบ่งจุดเป็น 2 ชุด ซีกซ้าย, ซีกขวา

② หา closest pair ซีกซ้าย

③ " ——— ซีกขวา

④ " ——— ที่ข้ามเขต

$$t(n) = 2t(n/2) + t(④)$$

ถ้า $t(④) = \Theta(n)$,

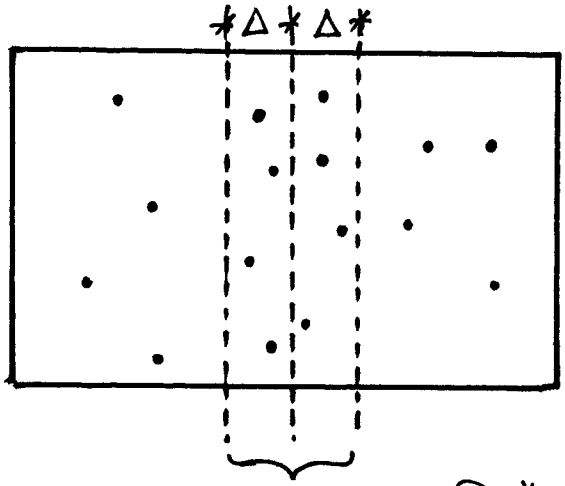
$$t(n) = \Theta(n \log n)$$

(I) ให้ d_ℓ คือ ระยะทางของ closest pair ฝั่งซ้าย

d_r คือ " ——— " ——— " ขวา

$\Delta = \min(d_\ell, d_r)$

หา (4) ได้



สนใจเฉพาะจุดใน"แถบ"นี้

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

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

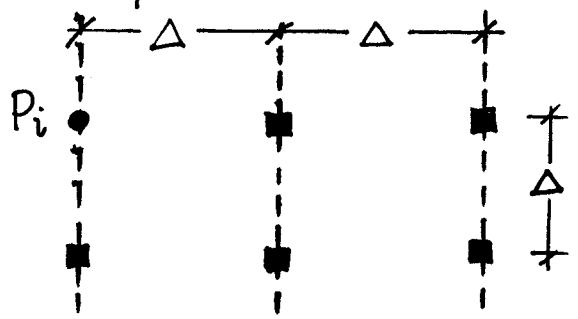
(II) เปรียบเทียบ เฉพาะ: จุดที่สังสัย

คู่จุดที่ ห่างเขต ที่จะสังสัย $< \Delta$

ต้องมี ผลต่าง ของ s:y: ทางในแนวตั้ง $< \Delta$

```
for (i ← 1 to s)
    for( j ← i+1 to s)
        if ( |y(Pi) - y(Pj)| > Δ )
            break  // inner loop
        else
        if (distance(Pi , Pj) < Δ)
            Δ = distance(Pi, Pj)
```

"จุด P_i แต่ละจุด จะเปรียบเทียบกับ P_j ไม่เกิน 7 จุด"



∴ (4) หาได้ในเวลา $\Theta(n)$

แต่ต้อง sort จุดใน แถบ ตามค่า y. $\Theta(n \log n)$

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

Dynamic Programming

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

ตย. Fibonacci number

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

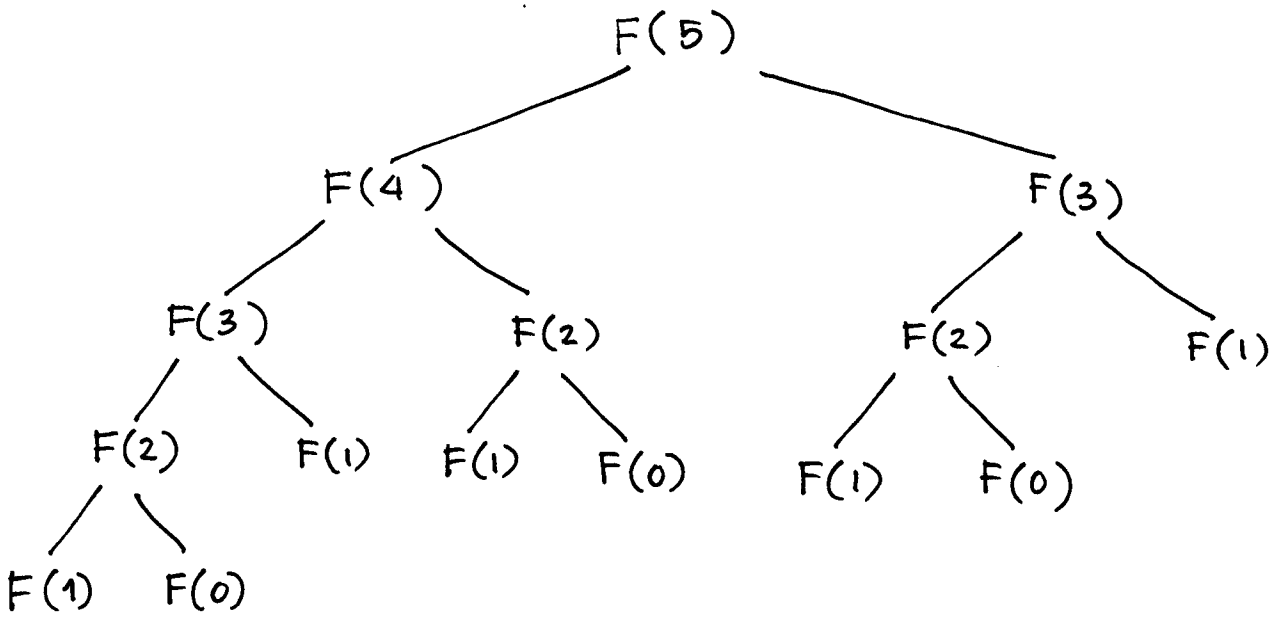
$$f_0 = 0 \,,\; f_1 = 1$$

(1)

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

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

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



→ เรียก Fib1() 15 ครั้ง.

---

(2)

```
Fib2( int n )

    F[0] ← 0
    F[1] ← 1
    for i = 2 to n
        F[i] = F[i-1] + F[i-2]
    return n
```

$t_n = \Theta(n)$

Making Change

- มีเหรียญ n ประเภท.

  d_i เป็นค่าของเหรียญประเภทที่ i

- ต้องการทอนเงินจำนวน N โดยใช้จำนวนเหรียญน้อยสุด

สร้างตาราง $C[1..n, 0..N]$

$C[i,j]$ เก็บจำนวนเหรียญน้อยสุดในการทอนเงิน j โดยใช้เหรียญประเภทที่ 1 ถึง i

---

① $C[i,0] = 0 \qquad 1 \le i \le n$

② $C[i,j]$ แยกเป็น

- $C[i-1, j]$ ไม่ใช้เหรียญแบบ i
- $1 + C[i, j-d_i]$ ใช้เหรียญแบบ i 1 เหรียญ

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

T.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	**2**

$\Theta(nN)$ time

Knapsack Problem (0/1)

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

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

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

$$x_i = 0, 1$$

สร้างตาราง $V[1..n, 0..W]$

$V[i,j]$ เก็บ value รวมมากสุดที่ได้เมื่อ
พิจารณา object 1 ถึง i และ
knapsack จุได้ j

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

Matrix-chain Multiplication

	$A_1\ A_2\ A_3$	$[10\times100][100\times5][5\times50]$
①	$((A_1 A_2) A_3)$	$10\cdot100\cdot5 + 10\cdot5\cdot50 = 7{,}500$
②	$(A_1 (A_2 A_3))$	$100\cdot5\cdot50 + 10\cdot100\cdot50 = 75{,}000$

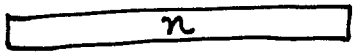
$A_1 A_2 A_3 A_4$

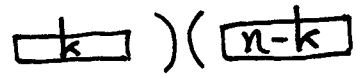
- $(A_1 (A_2 (A_3 A_4)))$
- $(A_1 ((A_2 A_3) A_4))$
- $((A_1 A_2)(A_3 A_4))$
- $((A_1 (A_2 A_3)) A_4)$
- $(((A_1 A_2) A_3) A_4)$

จำนวนการใส่วงเล็บ

กำหนดให้ $P(n)$ คือ จำนวนการใส่วงเล็บให้กับ

การคูณ matrice n ตัว

()

()( )

$$P(n) = \begin{cases} 1 & \text{if } n=1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2. \end{cases} \quad \Omega\left(\frac{4^n}{n^{3/2}}\right)$$

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

ตัวอย่าง $A_1 \cdot A_2 \cdot A_3$

10	100	5	50
P_0	P_1	P_2	P_3

Let $m[i,j]$ คือ จำนวนการคูณแบบ scalar ที่น้อยสุด ในการคูณ matrix chain $A_i A_{i+1} \cdots A_j$

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

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

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

ให้ $s[i,j]$ คือ ค่า k ในการแบ่ง $A_i A_{i+1} \cdots A_j$ แล้วได้ จำนวนการคูณ scalar ที่น้อยสุด

ตัวอย่าง $A_1\ A_2\ A_3\ A_4$

13 5 89 3 34

$P_0\ P_1\ P_2\ P_3\ P_4$

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

① หา $m[1,2]$, $m[2,3]$, $m[3,4]$

$m[1,2] = m[1,1] + m[2,2] + 13\times5\times89 = 5785$

$m[2,3] = m[2,2] + m[3,3] + 5\times89\times3 = 1335$

$m[3,4] = m[3,3] + m[4,4] + 89\times3\times34 = 9078$

② หา $m[1,3]$, $m[2,4]$

$m[1,3] = \min\{ (m[1,1]+m[2,3] + 13\times5\times3),$
$(m[1,2]+m[3,3] + 13\times89\times3) \}$

$= \min\{ 1530, 9256 \} = 1530$

$m[2,4] = \min\{ (m[2,2]+m[3,4] + 5\times89\times34),$
$(m[2,3]+m[4,4] + 5\times3\times34) \}$

$= \min\{ 24208, 1845 \} = 1845$

③ ค่า $m[1,4]$

$$m[1,4] = \min\{ (m[1,1]+m[2,4]+13\times5\times34),\ (m[1,2]+m[3,4]+13\times89\times34),\ (m[1,3]+m[4,4]+13\times3\times34) \}$$

$$= \min\{ 4055, 54201, 2856 \} = 2856$$

```
MATRIX-CHAIN( p, n)
    for i ← 1 to n do m[i,i] ← 0
    for l ← 2 to n do
        for i ← 1 to n-l+1 do
            j ← i+l-1
            m[i,j] ← ∞
            for k ← i to j-1 do
                q ← m[i,k] + m[k+1,j] +
                    p_{i-1} · p_k · p_j
                if q < m[i,j] then
                    m[i,j] ← q
                    s[i,j] ← k
    return m and s
```

$\Theta(n^3)$ time

$\Theta(n^2)$ space

Memoization

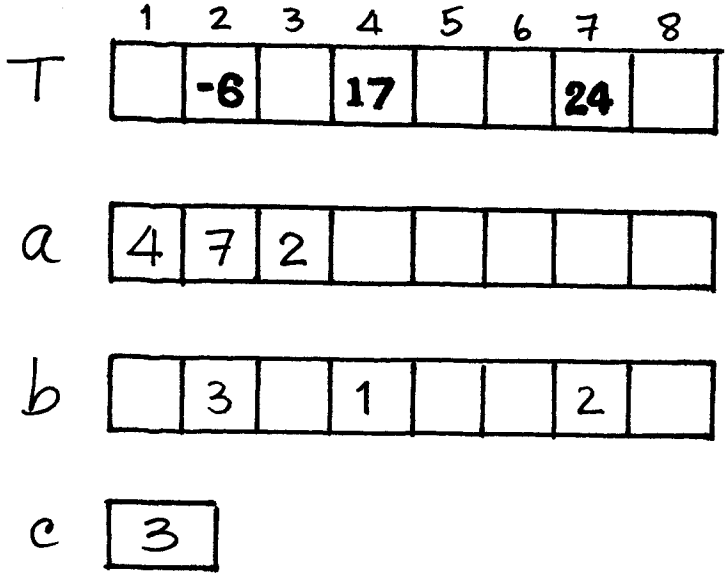
- memory function

```
MEMOIZED_MATRIX-CHAIN ( p, n )
    for i ← 1 to n
        for j ← i to n
            m[i, j] ← ∞
    return LOOKUP_CHAIN ( p, 1, n )
```

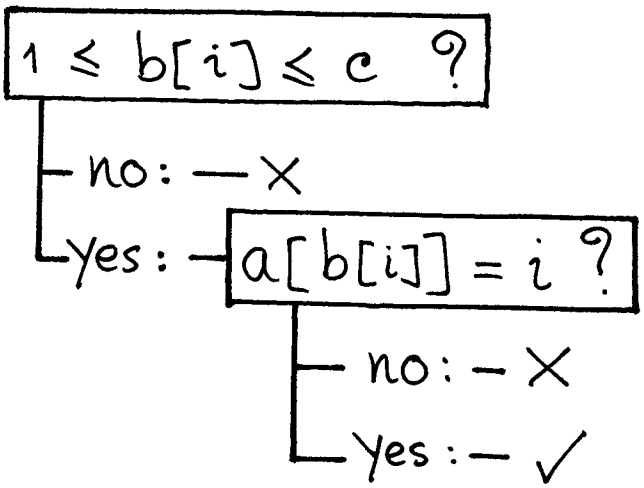
```
LOOKUP-CHAIN ( p, i, j )
    if m[i, j] < ∞ then return m[i, j]
    if i = j then m[i, j] ← 0
    else for k ← i to j-1
            q ← LOOKUP_CHAIN ( p, i, k) +
                LOOKUP_CHAIN ( p, k+1, j ) +
                p_{i-1} · p_k · p_j
            if q < m[i, j] then m[i, j] ← q
    return m[i, j]
```

Virtual Initialization

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



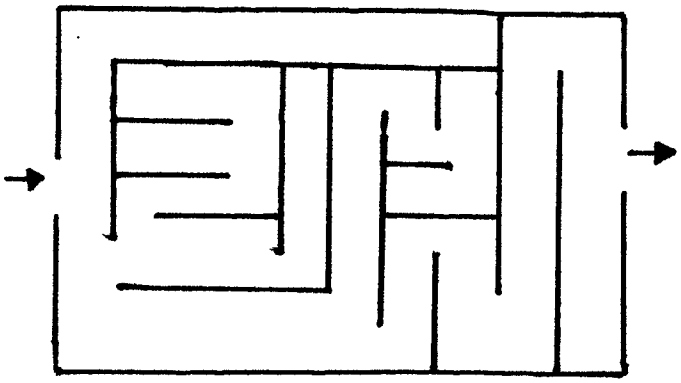
อยากรู้ว่า T[i] มีค่าเก็บอยู่หรือไม่

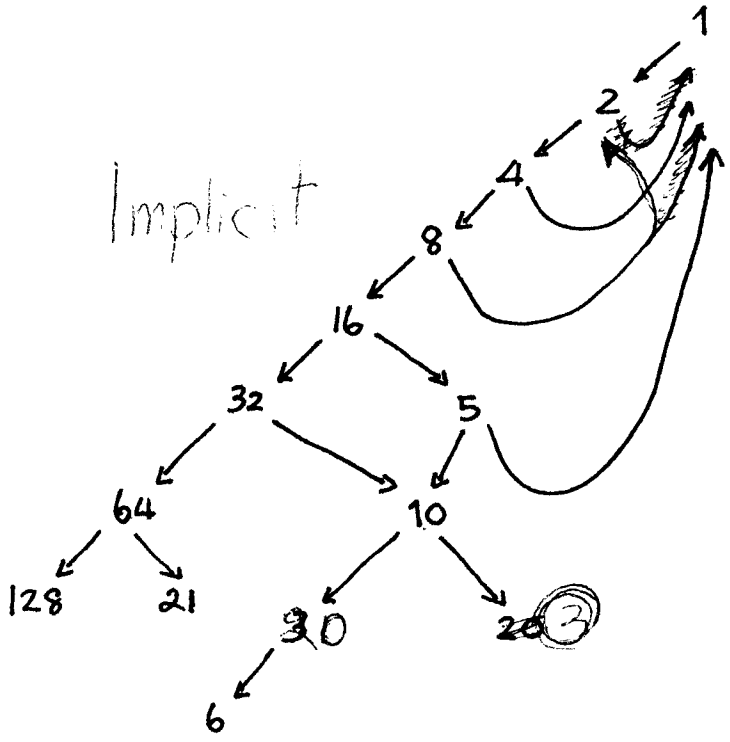


Dynamic Programming

- Characterize the structure of an optimal soln.
  "optimal substructures"
- recursively define the value of an optimal soln.
  "overlapping subproblems"
- compute the value in a bottom-up fashion.
- construct an optimal soln. from computed info.

Search and Traversal



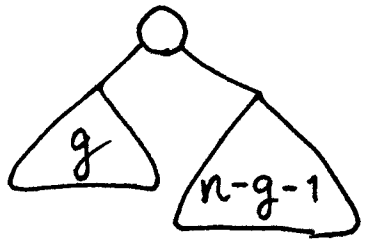


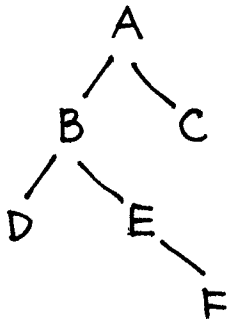
Implicit

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

Tree Traversal

- preorder
- inorder
- postorder





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

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

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

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

Let $T(n) \leq an + b$

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

Induction : $T(n) \leq \max_{0 \leq g \leq n-1} \{ (ag+b) + (a(n-g-1)+b) + c \}$

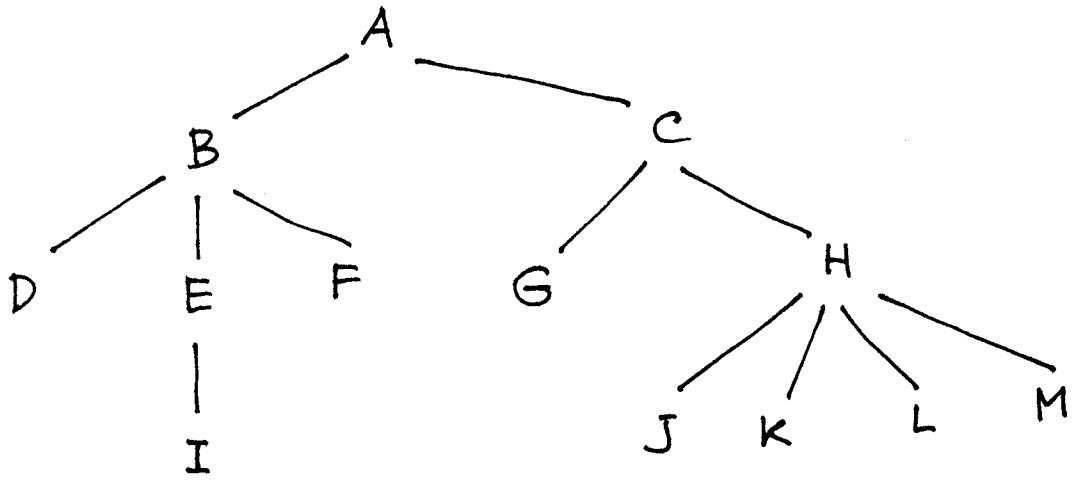
$$\leq an + 3b - a$$

choose $a \geq 2b$

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

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

Ancestry in a rooted tree



Determine whether v is an ancestor of w

- $\Omega(n)$ alg.
- preprocessing in $\Theta(n)$

  subsequent query in $O(1)$

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

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

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

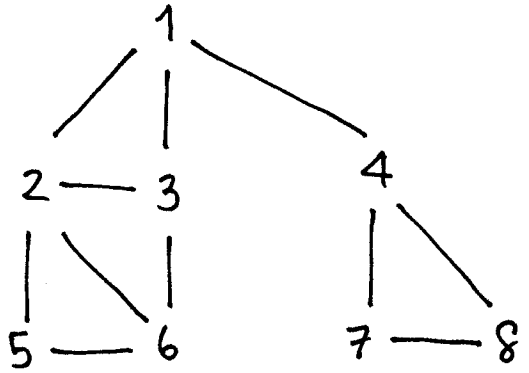
- v is an ancestor of w or
- v is to the right of w

Graph Search & Traversal

- Depth-first
- Breadth-first

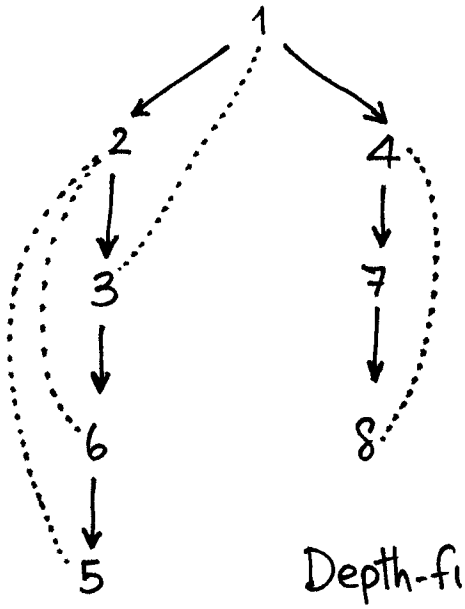
Depth-first Search : Undirected graphs

```
DFS ( v )
    mark [v] ← visited
    for each node w adjacent to v do
        if mark [w] ≠ visited then DFS(w)
```



DFS (1)

```
DFSearch ( G )
    for each v ∈ V do mark [v] ← not_visited
    for each v ∈ V do
        if mark [v] ≠ visited then DFS (v)
```



↓ tree edge

⋮ back edge

non-tree

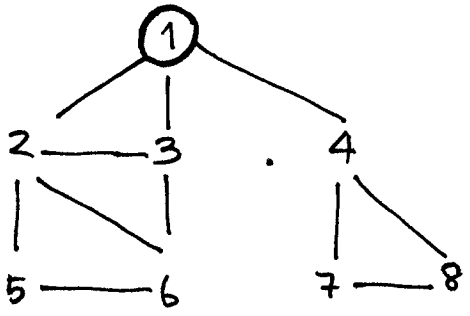
Depth-first search tree
(a spanning tree)

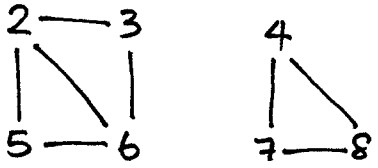
$G = (V, E)$ has v vertices, e edges.

- visit node ಒಂದನ್ನು → $\propto v$
- ಪ್ರತಿ neighbouring nodes → $\propto e$
  (ಗೆ adjaceney list)

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

Articulation Points





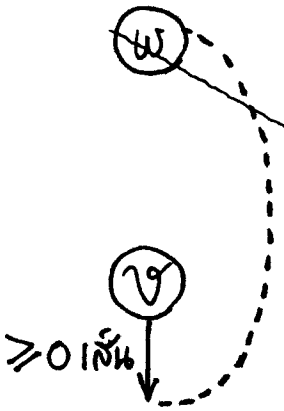
กราฟ G connected

ณ articulation point ของ G

ทำให้ได้ subgraph ที่มากกว่า 1 component

Biconnected graph ⇒ กราฟที่ไม่มี

articulation point .

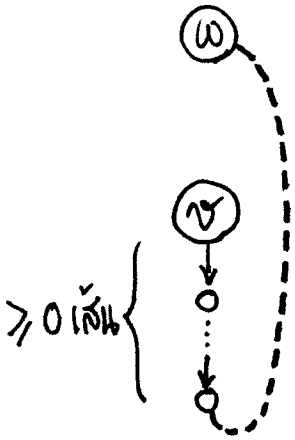


highest[v] = prenum[w]

- w อาจเป็น ancestor ของ v

- v ไม่มีลูก

- x เป็นลูก v แล: highest[x] < prenum[v]



← อยู่สูงสุด

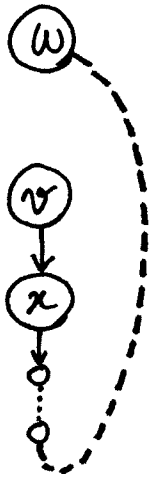
highest[v] = prenum[w]

w ต้องเป็น ancestor ของ v

---

① ถ้า v ไม่มีลูก → v ไม่ใช่ articulation point.

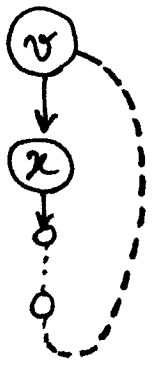
② ถ้า x เป็นลูกของ v และ: highest[x] < prenum[v]

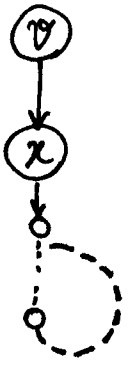


v ไม่ใช่รากด้วย, x ยังต่อกับ ancestor ของ v

∴ v ไม่ใช่ a.p.

③ ถ้า x เป็นลูกของ v และ: highest[x] ≥ prenum[v]

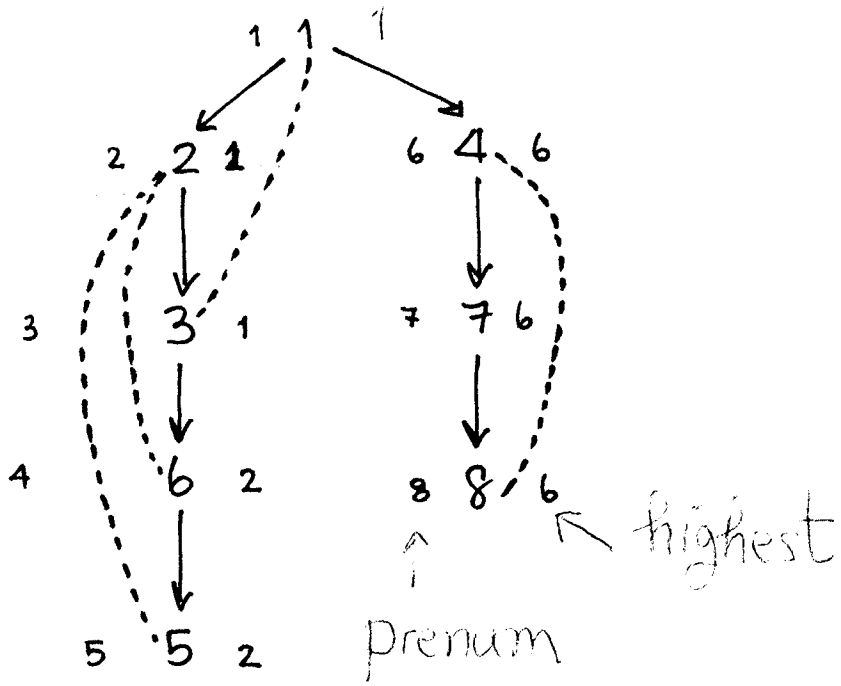


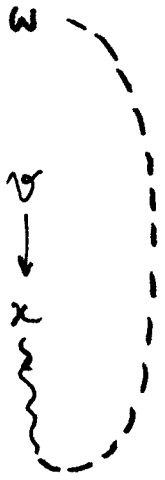


ลบ v

v และลูกหลานเป็น "หลุด"

∴ v เป็น a.p.


1 1 1
2 2 1
6 4 6
3 3 1
7 7 6
4 6 2
8 8 6
highest
5 5 2
prenum


w
v
x

Finding Articulation Points

① prenum หาได้จาก DFS

```
pnum++;
prenum[v] ← pnum
```

---

② traverse DFS tree in postorder

highest[v] = min. of

- prenum[v]
- prenum[w] (v,w) is a back edge
- highest[x] x เป็นลูกของ v

---

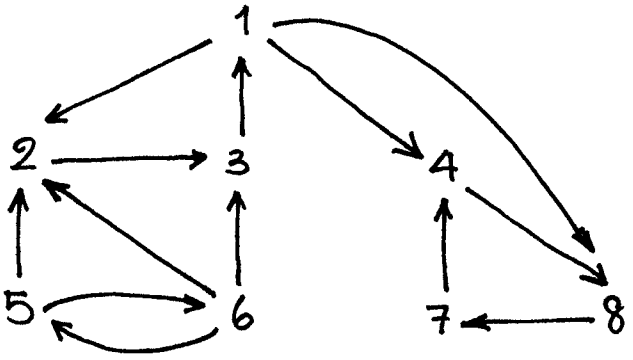
③ • root ของ DFS tree เป็น a.p. ถ้ามี #ลูก > 1

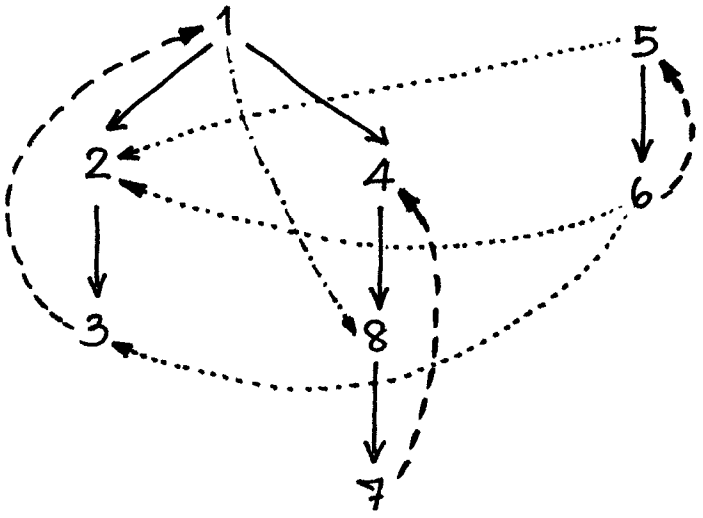
• node v เป็น a.p. ก็ต่อเมื่อ ∀ลูก x ที่

highest[x] ≥ prenum[v]

Depth-first Search : Directed graphs

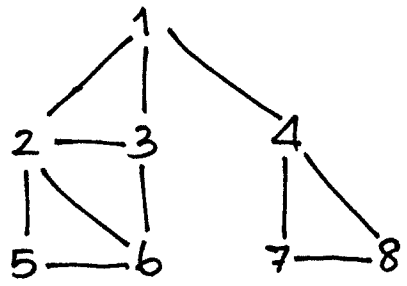
- คล้ายวิธีการทำ DFS กับ undirected graph.
- "adjacent" หมายถึงของ edge ด้วย

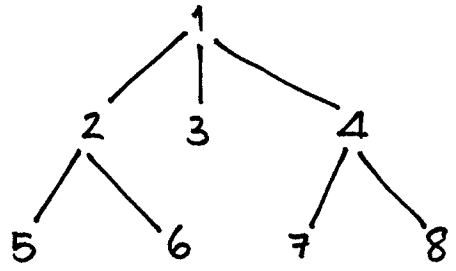




- tree edge
- back edge
- forward edge
- cross edge

Breadth-first Search





BFS tree

```
BFS(v)
    Q ← ∅
    mark[v] ← visited
    enqueue(Q, v)
    while Q ≠ ∅ do
        u ← dequeue(Q)
        for each node w adj. to u do
            if mark[w] ≠ visited then
                mark[w] ← visited
                enqueue(Q, w)
```

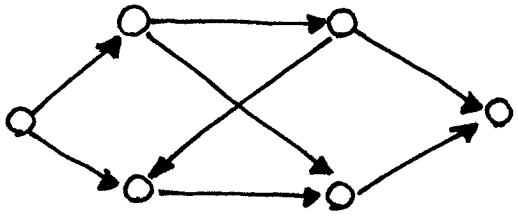
$\Theta(\max(e,v))$

- adjacency list

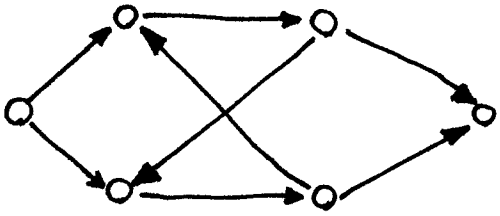
BFSearch เหมือน DFSearch

เปลี่ยน DFS เป็น BFS

Directed Acyclic Graphs



✓

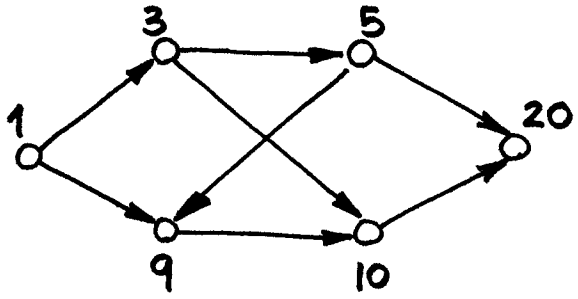


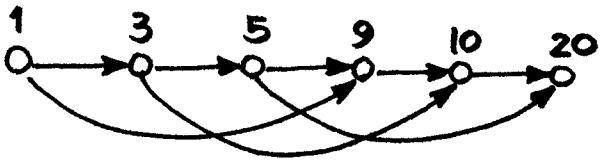
✗

Is G a DAG ?

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

Topological Sorting





---

① แฟ้ม "push ข" ที่บรรทัดท้ายสุดของ DFS

② เรียก DFSearch (G)

③ ลำดับข้อมูลที่ได้จากการ pop stack จนหมด

Backtracking

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

  x_i เลือกมาจาก finite set S_i

- satisfy criterion function

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

Brute force :

กำหนดให้ S_i มี m_i

∴ มี $m_1 m_2 \cdots m_n$ possible candidates

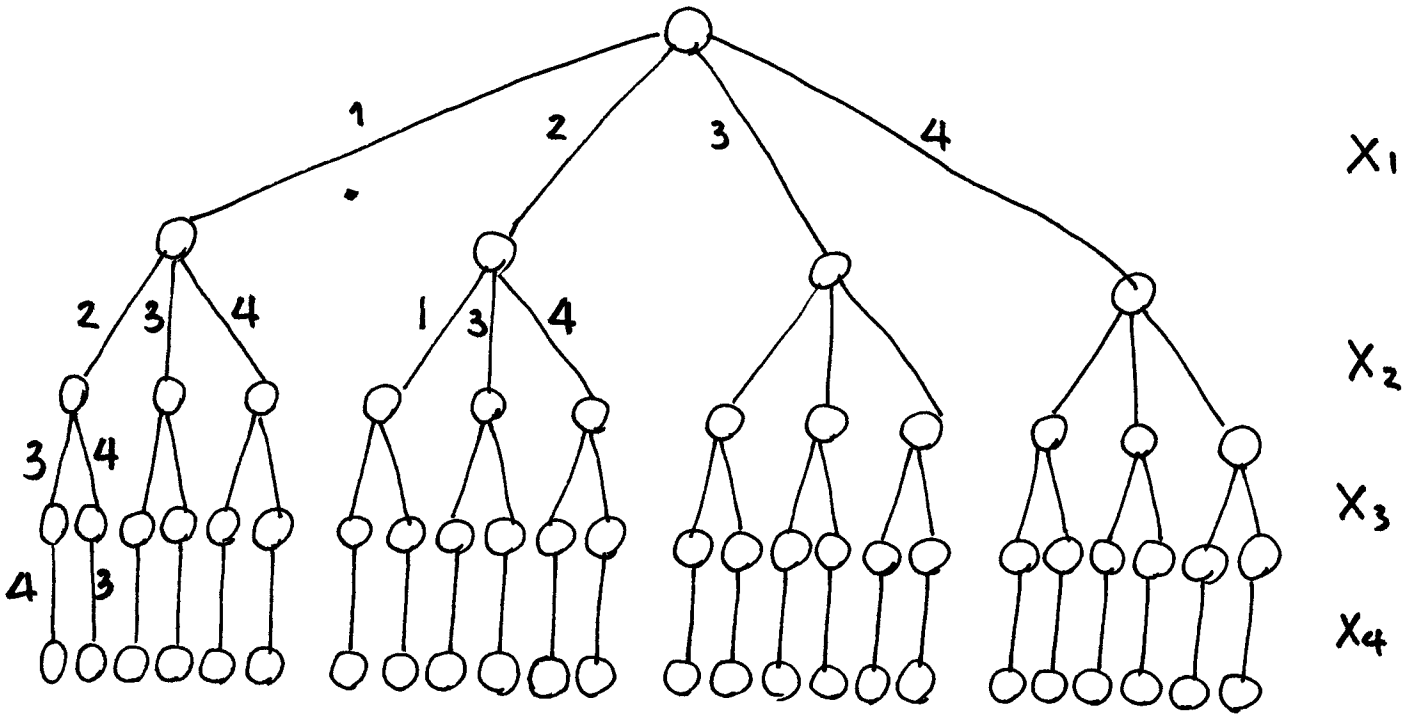
Backtracking :

- พิจารณา candidates เป็นจำนวน
  น้อยกว่าแบบ brute force มาก
- พิจารณา "partial" vector

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

  ถ้าไม่มีทางนำไปสู่คำตอบได้แน่
  ก็ไม่ต้องดูอีก $m_{i+1} \cdots m_n$ vectors.

4 - Queen



Sum of Subsets

ปัญหา ให้ $\{w_1, w_2, w_3, \ldots, w_n\}$ และ M

จงหา subset ที่ผลรวมของสมาชิก

สมกับเท่ากับ M

ต.ย. $\{2, 1, 3, 5, 10, 4\}$, 8

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

รูปแบบของคำตอบ

① $(x_1, x_2, \ldots, x_k)$ $1 \le k \le n$

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

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

② $(x_1, x_2, \ldots, x_n)$

- $\sum_{i=1}^{n} w_i * x_i = M$
- $x_i = 0$ หรือ 1

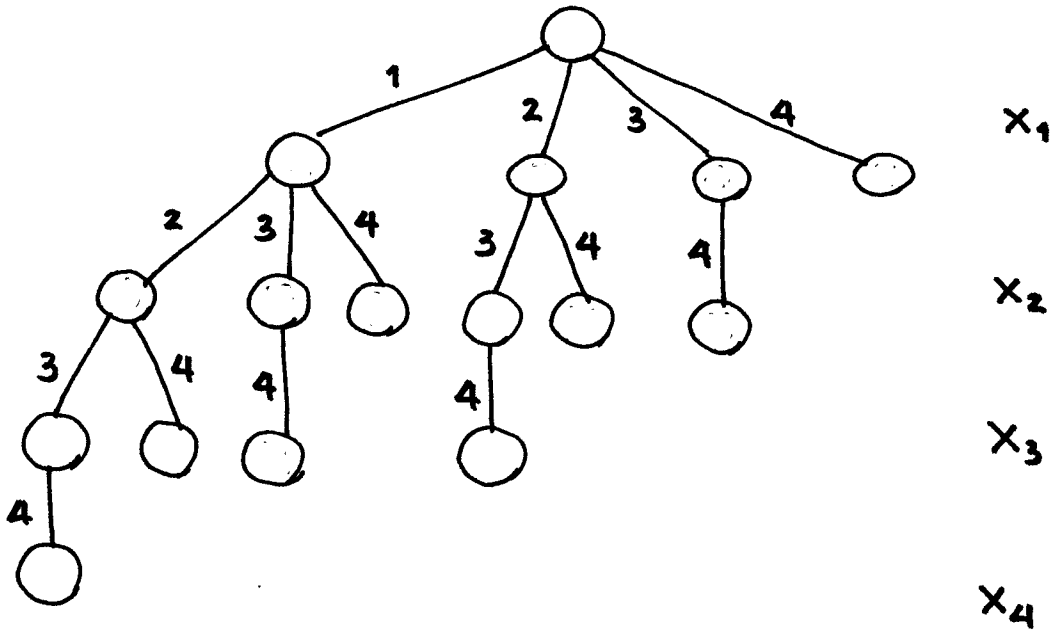
(0,1,1,0,0,1)
(0,0,1,1,0,0)
(1,1,0,1,0,0)

Solution space ของ ① คือ 2^n

State Space Trees

Sum of Subsets

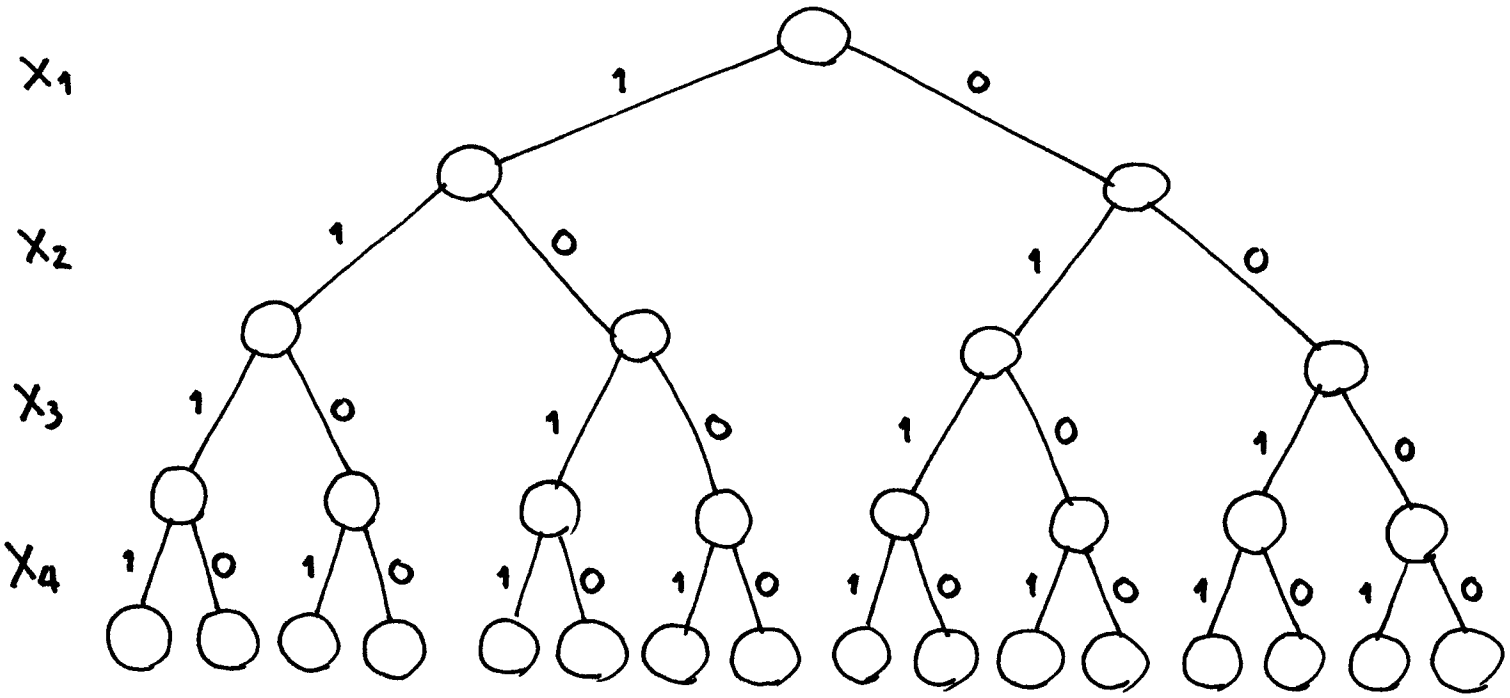
- $n = 4$
- รูปแบบคำตอบ (1)



ตย. $\{10, 3, 5, 9\}$, 12.

Sum of Subsets

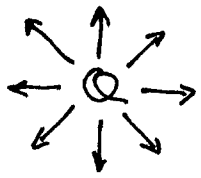
- รูปแบบคำตอบ (2)



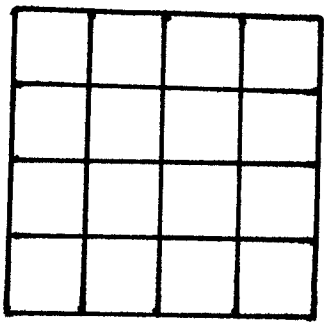
- problem states
- solution states
- answer states

Eight-Queen Problem

- วาง Queen จำนวน 8 ตัว
  บนตาราง 8×8 โดยไม่มี
  queen ของตัวใด "attack" กัน



- 4-Queen



① เลือกวาง 4 ช่อง จาก 16 ช่อง → $\binom{16}{4}$ =

② แต่ละ: ตัว → 4^4 =

③ แต่ละ: ตัว ต่อมันละ: ตัว → $4!$ =

"Solution space" ⟶

- 8-Queen
  - permutation ของ (1,2,3,4,5,6,7,8)
  - ไม่มี queen "เดิน" กันในแนวทแยง

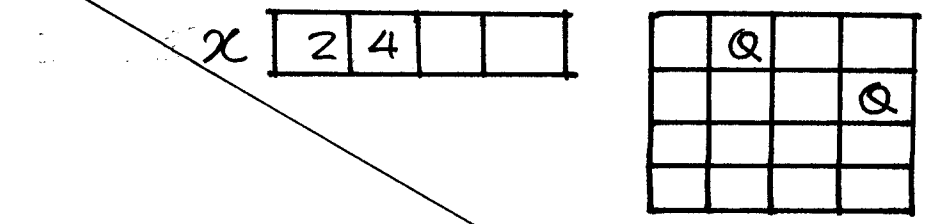
Backtracking

- 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



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 : (รูปแบบอย่างของ ②)

assume w_i are initially
in nondecreasing order.

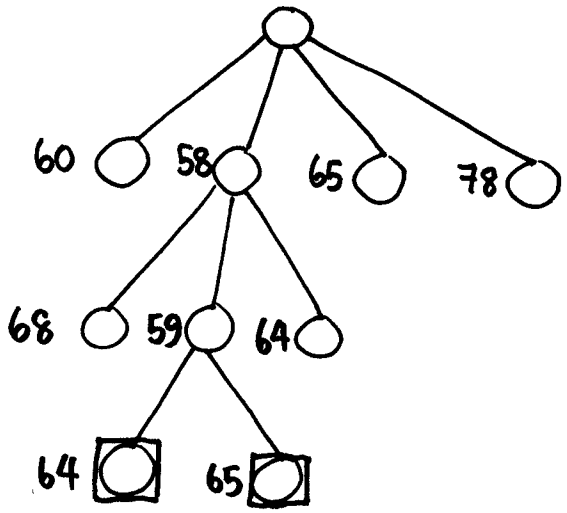
$x[1..k]$ ไม่สามารถนำไปสู่คำตอบได้ ถ้า

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

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

Branch-and-Bound

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

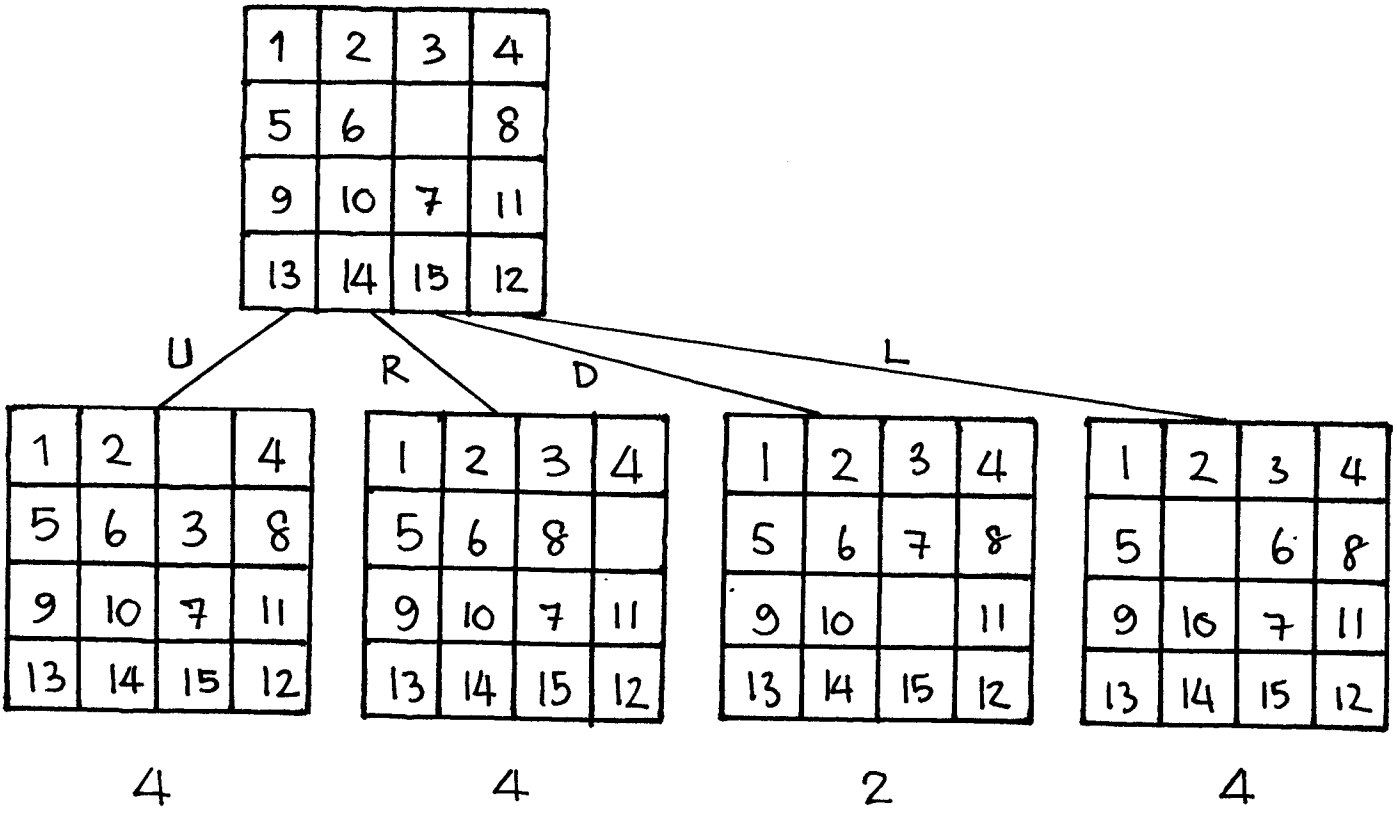


LC-Search + Bounding function

หยิบได้ 1 ก้อนที่มีค่า 64

สามารถตัด node อื่นที่ > 64 ออกได้

15-puzzle



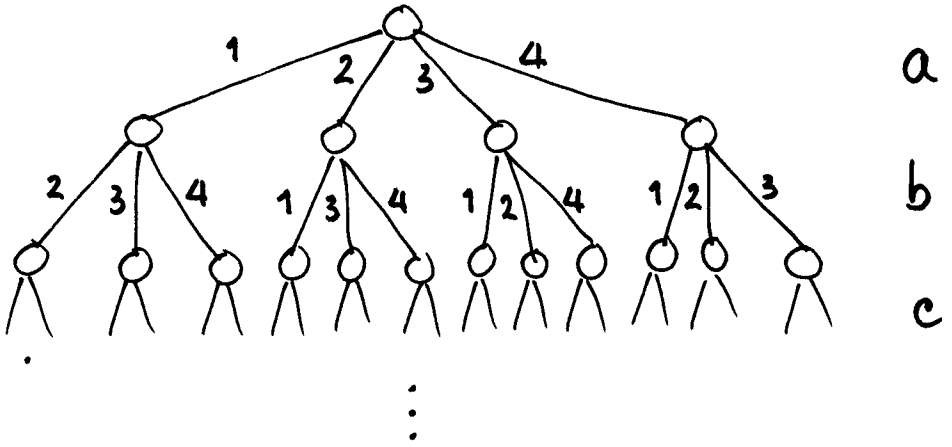
4 4 2 4

Assignment Problem

- มีงาน n งาน ให้คน n คน
- 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



เหมือน 4-Queen

- Bounding function
  - upper bound.

    เลือกมา 1 feasible solution

    a→1, b→2, c→3, d→4

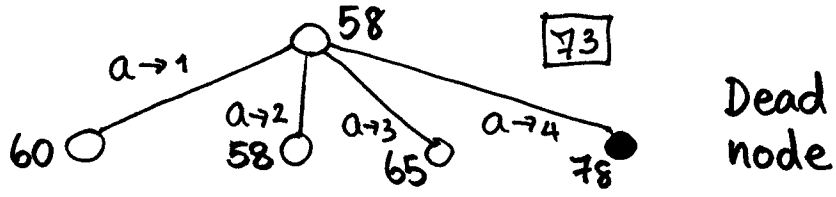
    11 + 15 + 19 + 28 = 73 "อยากได้น้อยๆ"
  - lower bound

    ผลรวมของค่าน้อยสุดในแต่ละ column

    11 + 12 + 13 + 22 = 58 "อยากได้มากๆ"
  - answer node ต้องมี cost อยู่ในช่วง

    [58 .. 73]

- Branching

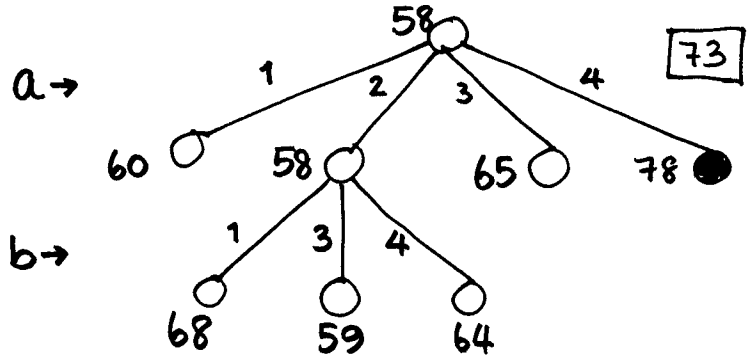


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

a→1	11 + 14 + 13 + 22 = 60
a→2	12 + 11 + 13 + 22 = 58
a→3	18 + 11 + 14 + 22 = 65
a→4	40 + 11 + 14 + 13 = 78 *

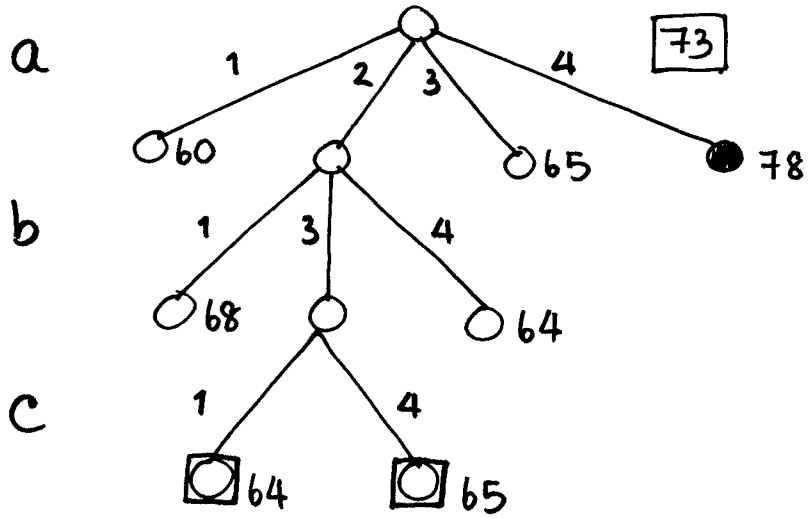
11 + 14 + 13 + 22
a→1

12 + 11 + 13 + 22
a→2



	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→2, b→1	12 + 14 + 19 + 23 = 68
a→2, b→3	12 + 13 + 11 + 23 = 59
a→2, b→4	12 + 22 + 11 + 19 = 64



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

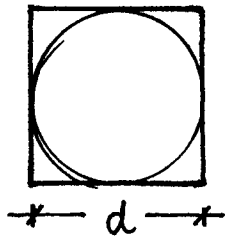
a→2, b→3, c→1 (d→4)	12+13+11+28 = 64
a→2, b→3, c→4 (d→1)	12+13+23+17 = 65

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.



พ.ท วงกลม = $\frac{\pi d^2}{4}$

พ.ท สี่เหลี่ยม = d^2

A = # ลูกดอกในวงกลม

B = # ″ — ″ นอกวงกลม.

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

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

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 ← T[ uniform(1,n) ]
    c ← 0
    for( i←1 to n )
       if ( x = T[i]) c++
    return ( c > n/2 )
```

½-correct

```
Repeat Maj ( T[1..n], k )
    for( i ← 1 to k )
        if( Maj( T ) = true )
              return true
    return false
```

Monte Carlo Alg.

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

- Pick X
  - $1 \times n$ matrix
  - randomly chosen binary vector
- Notice $XAB = XC$?

```
MMult(A,B,C,n)
  for i ← 1 to n
      X[i] ← uniform(0..1)
  if (XA)B = XC then return true
              else return false
```

½-correct

```
RepeatMMult(A,B,C,n,k)
  for i ← 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 $A = a_0 a_1 \cdots a_{n-1}$

$B = b_0 b_1 \cdots b_{n-1}$

Is B a cyclic shift of A?

ex: "somchai" : "mchaiso"

pattern matching: Let $A = a_0 a_1 \cdots a_{n-1}$

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

Does A contain B?

ex: "somchaisomchai" : "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 ถ้ารู้ algorithm ที่แก้ปัญหา Q

ก็รู้ " ——— " ——— P

② ปัญหา Q ไม่ง่ายกว่า 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 ← 1 to n
        X[i] ← nd_choice(0,1)
    sumW = Σ_{1≤i≤n} W[i]·X[i]
    sumP = Σ_{1≤i≤n} P[i]·X[i]
    if sumW > M or sumP < R
        then failure
        else success
```

"polynomially verifiable"

Definition

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

Theorem : $P \subseteq NP$

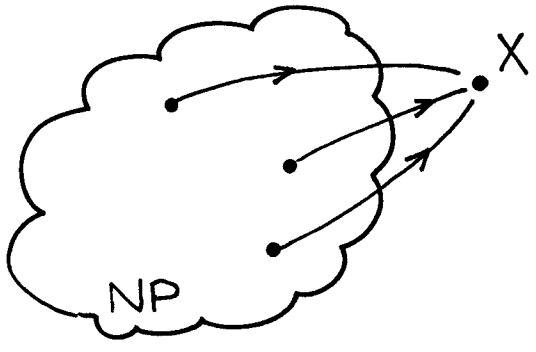
ปัญหาที่ยังรอให้แก้

$P = NP$?

$P \neq NP$?

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

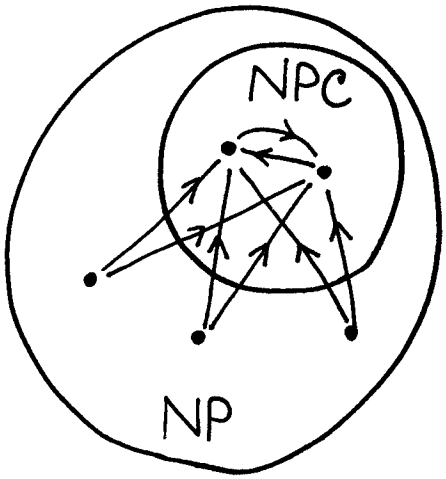
- $X \in NP$ and
- X is NP-hard.

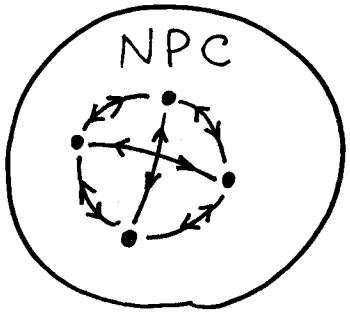
Theorem

Let X be an NP-complete problem.

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

Z is also NP-complete





ปัญหาใน NPC ยากเท่ากันหมด.

ง่าย

Satisfiability Problem

S ให้เป็น Boolean expression

ที่อยู่ในรูป CNF

(Conjunctive Normal Form)

ตัวอย่าง $S = (x+y+\bar{z})\cdot(\bar{x}+y+z)\cdot(\bar{x}+\bar{y}+\bar{z})$

Is S satisfiable ?

- SAT-CNF is in NP

> Cook's Theorem
>
> SAT-CNF is NP-complete

NP-Complete Problems:

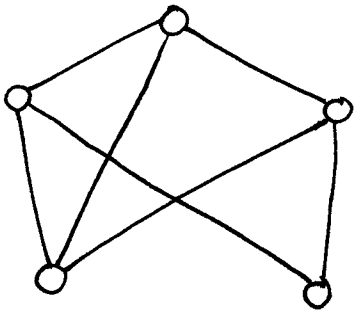
- SAT-3-CNF
- 3COL (Can G be painted with 3 colours ?)
- HAMD (Is there any Hamiltonian cycle in G ?)
- TSPD (Is there any TS tour in G of length $\leq L$?)

Is TSPD NP-complete?

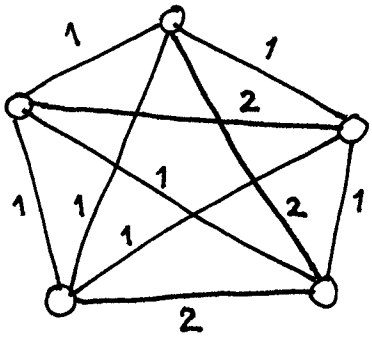
ತಪ್ಪಾಗಿರುವುದನ್ನು ತೋರಿಸುವುದು HAMD ಯು NPC

① TSPD is in NP

② HAMD $\leq_T^P$ TSPD



G



G'

$L = 5$

Is there a tour in G' of length ≤ 5 ?