---------- 1 & 2 ----------

Introduction & Getting Started

1.1 Algorithms

電腦可以執行

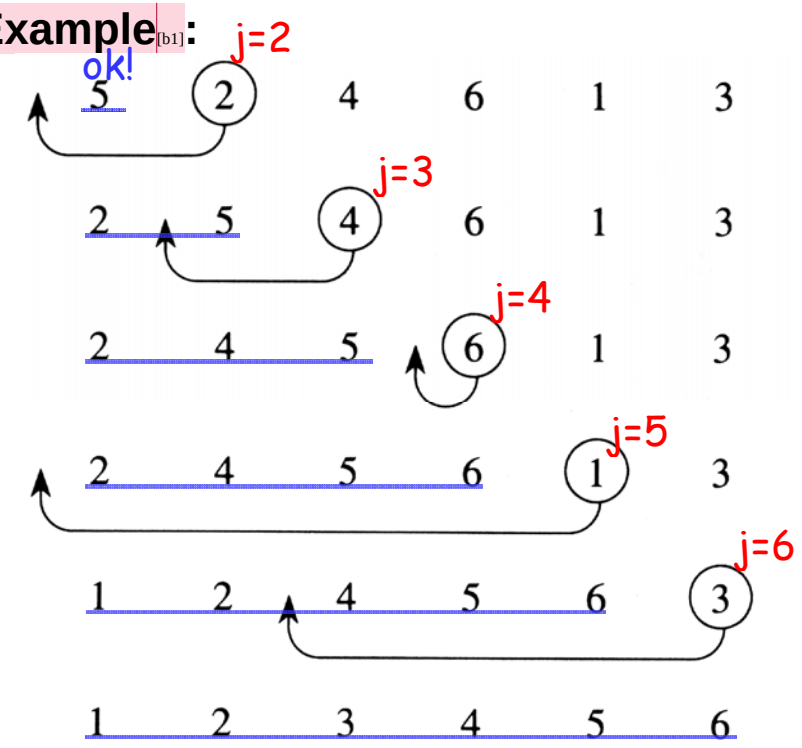
Algorithm*:** A sequence of computational steps that transform the ***input of a ***computational problem*** into the ***output***.

* best for $n \le 30$

2.1 Insertion Sort: An efficient algorithm for sorting a small number of elements.

(Sort a hand of cards)

Example[b1]:



1&2-1a

2.2 Analyzing Algorithms

Model

***RAM*:** *Random-access machine*, in which each memory access takes unit time and instructions are executed one by one.

Running time*:** number of steps, which is a function of the ***input size.

Primitive operations

(memory access, +, -, *, /, ...)

a := b + c; ⇨ 4 steps

qsort(...) ⇨ 1 step???

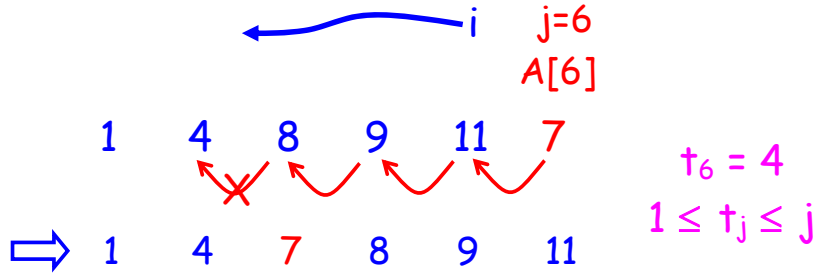
Example: Insertion Sort

	INSERTION-SORT(A)	cost	times
1	**for** $j \leftarrow 2$ **to** $length[A]$	c_1	n
2	**do** $key \leftarrow A[j]$	c_2	$n-1$
3	▷ Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4	$i \leftarrow j-1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	**do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] \leftarrow key$	c_8	$n-1$

n

constants

1&2-2a



$$T(n) = c_1 n + (c_2+c_4+c_8)(n-1) + c_5 \sum_{j=2}^{n} t_j +$$

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

***Best-case*:**

Each t_j=1. (The input *A* is sorted.)

$T(n)$ = $(c_1+c_2+c_4+c_5+c_8)n-(c_2+c_4+c_5+c_8)$

= $\Theta(n)$ [b2][b3]

(***rate of growth***, ***order of growth***)

最 大 項

***Worst-case*: (upper bound)**

Each $t_j=j$.

$T(n)$ = $k_1 n^2 + k_2 n + k_3$

= $\Theta(n^2)$

***Average-case*: (Expected running time)**

Each $t_j=j/2$.

$T(n)$ = $t_1 n^2 + t_2 n + t_3$

= $\Theta(n^2)$

1&2-3a

2.3 Designing Algorithms

Divide-and-Conquer [b4]: 1&2-4a

*not similar

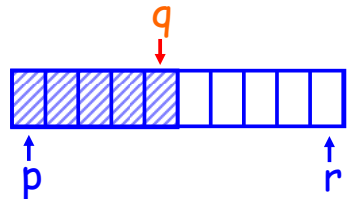
Divide: (into the same problems of [bf5] smaller size)

Conquer [b6]: recursively solve each subproblems

Combine:

*A simple example: Finding maximum 1&2-4b

Example: **Merge Sort** 1&2-4c

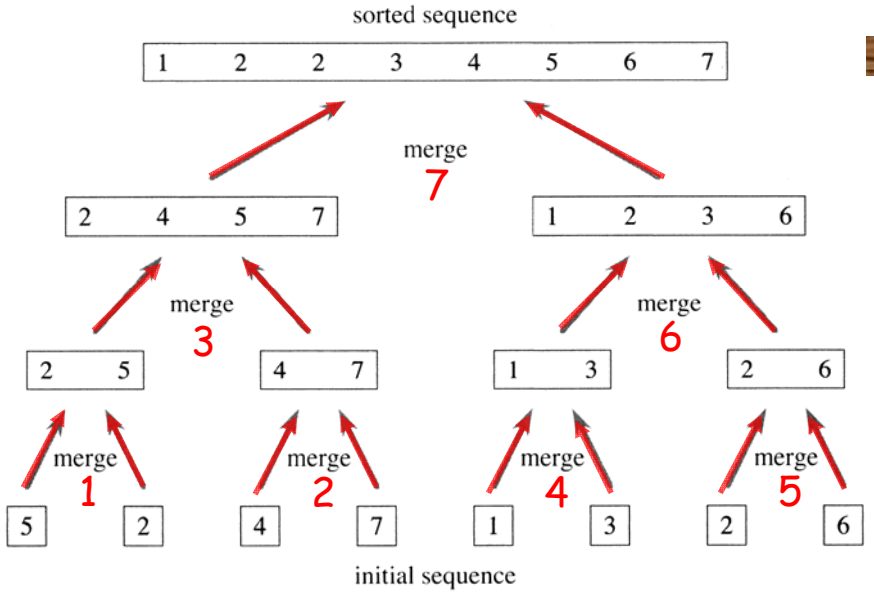


[b7]

```
MERGE-SORT(A, p, r)
1  if p < r                          → n ≥ 2
2     then q ← ⌊(p + r)/2⌋
3          MERGE-SORT(A, p, q)
4          MERGE-SORT(A, q + 1, r)
5          MERGE(A, p, q, r)         → O(r-p+1)
```

p ≥ r (n ≤ 1) "termination condition" (do nothing)

else



1&2-5a

Figure 2.4 The operation of merge sort on the array $A = \langle 5, 2, 4, 7, 1, 3, 2, 6 \rangle$. The lengths of the sorted sequences being merged increase as the algorithm progresses from bottom to top[b8].

Analysis: (*recurrence*)

constant time

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

How to merge in Θ(n) time?

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

Homework: Pro. 2-1 and 2-4[b9].