歸納與遞歸 Induction and Recursion

第五章

Outline

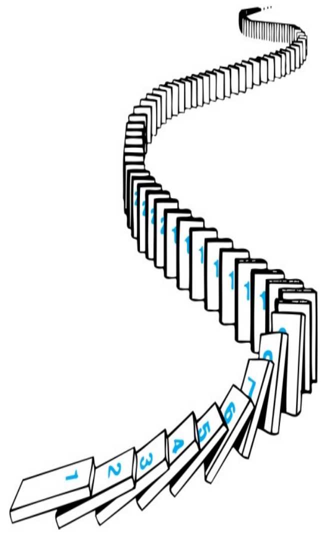
- 數學歸納法 (Mathematical Induction)
- 遞歸定義 (Recursive Definitions)
- 遞歸算法 (Recursive Algorithms)

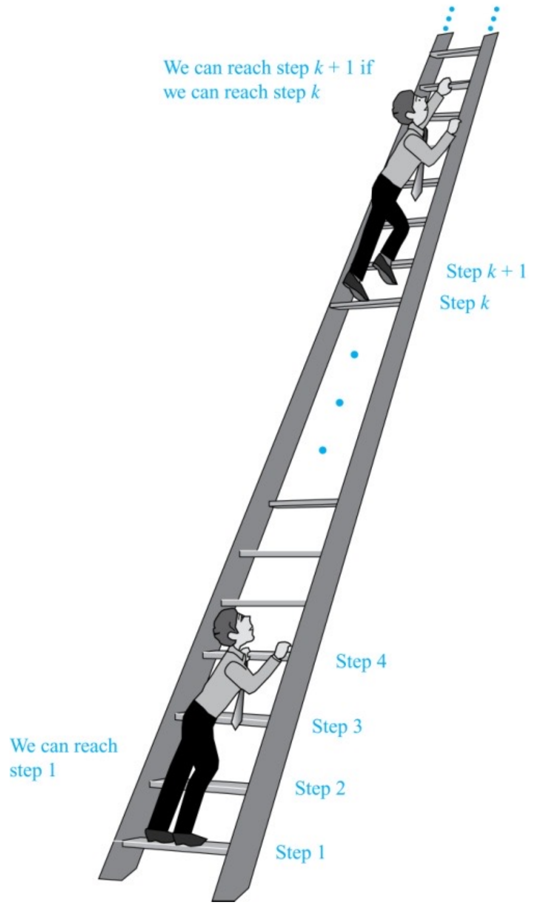
+ 數學歸納法 Mathematical Induction (5.1)

$$\Big(P(1) \wedge \forall k\,\big(P(k) \rightarrow P(k+1)\big)\Big) \rightarrow \forall n P(n)$$

- 即對於從某起始值開始的所有自然數，
  證明 $\forall n P(n)$ 為真需完成以下兩個步驟：

1. 基礎步驟: $P(1)$
2. 歸納步驟: $\forall k\,(P(k) \rightarrow P(k+1))$





例1

- 用數學歸納法(Mathematical Induction)證明:

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

例2

- 試找出序列 $1, 3, 5, 7, 9, \ldots$ 前 n 項和的公式並用數學歸納法證明。

例3

- 用數學歸納法證明:

A. $\forall n \in \mathbf{Z}^{+}, n < 2^{n}$

例3(續)

- 用數學歸納法證明:

B. $\forall n \in \boldsymbol{Z}^+, 21|(4^{n+1}+5^{2n-1})$

Proof:

Let $P(n)$: $\forall n \in \boldsymbol{Z}^+, 21|(4^{n+1}+5^{2n-1})$

$\rightarrow P(1): 4^{1+1}+5^{2(1)-1}=21, 21|4^{1+1}+5^{2(1)-1}, \therefore P(1)\ is\ true.$

$\rightarrow Assume\ P(k)\ is\ true: 21|\left(4^{k+1}+5^{2k-1}\right), k \in \boldsymbol{Z}^+.$

$\therefore 4^{k+1}+5^{2k-1}=21t, t \in \boldsymbol{Z}.$

$$\begin{aligned} Then\ P(k+1):\ 4^{(k+1)+1}+5^{2(k+1)-1} &= 4\cdot 4^{k+1}+25\cdot 5^{2k-1} \\ &= 4\left(4^{k+1}+5^{2k-1}\right)+21\cdot 5^{2k-1} \\ &= 4(21t)+21\cdot 5^{2k-1} \\ &= 21(4t+5^{2k-1}) \end{aligned}$$

$\because 4t+5^{2k-1} \in \boldsymbol{Z},$

$\therefore 21|\left[4^{(k+1)+1}+5^{2(k+1)-1}\right]$

$\therefore \forall k\left(P(k) \rightarrow P(k+1)\right)$

By PMI, $\forall n P(n)$ is true.

+ 例3(續)

- 用數學歸納法證明:

C. $\forall n \in \boldsymbol{N}, 2 - 2\cdot 7 + 2\cdot 7^2 - \cdots + 2\cdot(-7)^n = \frac{1-(-7)^{n+1}}{4}$

<u>Proof:</u>

Let $P(n)$: $\forall n \in \boldsymbol{N}, 2 - 2\cdot 7 + 2\cdot 7^2 - \cdots + 2\cdot(-7)^n = \frac{1-(-7)^{n+1}}{4}$

$\rightarrow P(0): LHS = 2\cdot(-7)^0 = 2, RHS = \dfrac{1-(-7)^{0+1}}{4} = 2, \therefore P(0)\ is\ true.$

$\rightarrow Assume\ P(k)\ is\ true: 2 - 2\cdot 7 + 2\cdot 7^2 - \cdots + 2\cdot(-7)^k = \dfrac{1-(-7)^{k+1}}{4}$

$$\begin{aligned} Then\ P(k+1): & 2 - 2\cdot 7 + 2\cdot 7^2 - \cdots + 2\cdot(-7)^k + 2\cdot(-7)^{k+1} \\ &= \frac{1-(-7)^{k+1}}{4} + 2\cdot(-7)^{k+1} \\ &= \frac{1-(-7)^{k+1}+8\cdot(-7)^{k+1}}{4} \\ &= \frac{1-(-7)\cdot(-7)^{k+1}}{4} \\ &= \frac{1-(-7)^{(k+1)+1}}{4} \end{aligned}$$

$\therefore \forall k\big(P(k) \rightarrow P(k+1)\big)$

By PMI, $\forall n P(n)$ is true.

+ 遞歸定義函數 Recursively Defined Functions (5.3)

- 以 N 作定義域的遞歸定義函數 $f(n)$ 含以下兩個步驟:
  1. 基礎步驟: 定義函數的起始值 $f(0)$
  2. 歸納步驟: 給出函數 $f(n)$ 對應較小取值之函數值所定義的規則

- 範例一:
  以下對應單元二 2.4 中的遞歸序列 1, 4, 7, 10, ⋯ 就是由非負整數集至實數集的函數:

$$a_n = a_{n-1} + 3 \qquad \text{其中 } a_0 = 1$$

- 範例二:

$$f(n+1) = \frac{f(n-1)}{f(n)}, \qquad \text{其中 } f(0) = -1, f(1) = 2$$

例4

- 給出下列序列$\{a_n\}$的遞歸定義(Recursive Definition):

A. $a_n = 4n - 2$

B. $a_n = 1 + (-1)^n$

C. $a_n = n(n+1)$

D. $a_n = n^2$

Solutions:

A. $a_n = a_{n-1} + 4,\quad a_1 = 2, n \geq 2$

B. $a_n = a_{n-2},\quad a_1 = 0, a_2 = 2, n \geq 3$

C. $a_n = a_{n-1} + 2n,\quad a_1 = 2, n \geq 2$

D. $a_n = a_{n-1} + 2n - 1,\quad a_1 = 1, n \geq 2$

遞歸算法 Recursive Algorithms (5.4)

- 遞歸算法 (Recursive Algorithms): 把原來的輸入值(Input)通過縮小為更小相同結構以解決問題的算法。
  - 算法結束點為某既定初始值。
- 例如: (n 的階乘的遞歸算法)

```
**procedure** *factorial*($n$: nonnegative integer)
**if** $n = 0$ **then return** 1
**else return** $n \cdot factorial(n - 1)$
{output is $n!$}
```

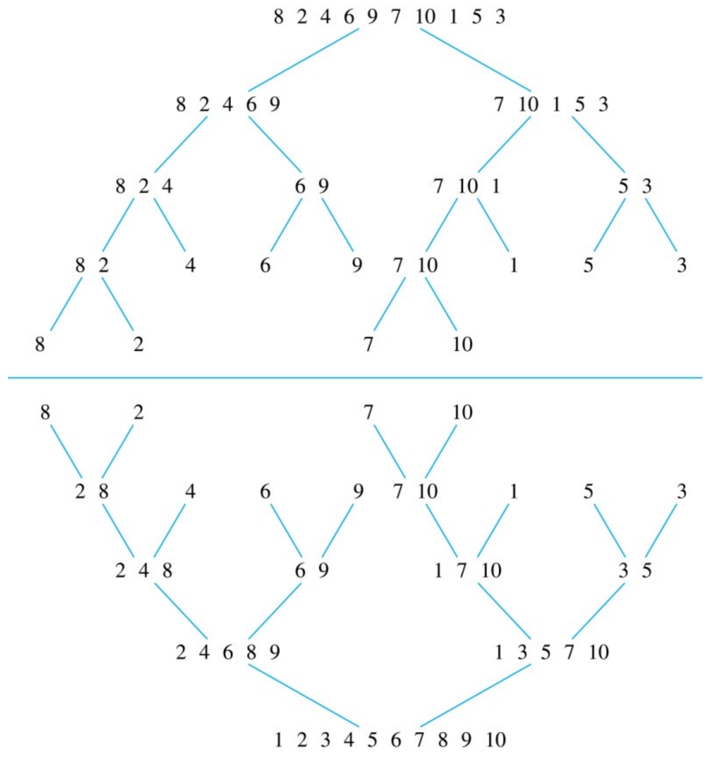
+ 例5

- 試寫一個計算正整數前 n 項和的遞歸算法。

```
**Procedure** Sum ($n$: positive integer)
**if** $n = 1$ **then return** 1
**else return** $n$ + sum_of_n($n - 1$)
{output is ($1 + \ldots + n$)}
```

歸併排序 Merge Sort

- 反覆地把列表拆分成長度相等的兩個子表直到每個子表包含一個元素為止。
  - 每些子表的序列可以表示成平衡二叉樹(balanced binary tree)。
- 在每一步中，按元素的升序排列合併成對的子表。 當所有子列表都已合併時結束。
  - 合併子表的序列可以表示成平衡二叉樹。



- 作排序的主算法mergesort的偽代碼:

```
procedure mergesort($L = a_1, a_2, \ldots, a_n$ )
if $n > 1$ then
        $m := \lfloor n/2 \rfloor$
        $L_1 := a_1, a_2, \ldots, a_m$
        $L_2 := a_{m+1}, a_{m+2}, \ldots, a_n$
        $L := merge(mergesort(L_1), mergesort(L_2))$
{$L$ is now sorted into elements in increasing order}
```

其中需應用子算法merge (下一頁)。

歸併排序 Merge Sort

- 歸併兩個有序表的子算法merge的偽代碼:

```
procedure merge($L_1$, $L_2$ :sorted lists)
$L$ := empty list
while $L_1$ and $L_2$ are both nonempty
    remove smaller of first elements of $L_1$ and $L_2$ from its list;
        put at the right end of $L$
    if this removal makes one list empty
        then remove all elements from the other list and append them to L
return $L$ {$L$ is the merged list with the elements in increasing order}
```

TABLE 1 Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.

First List	*Second List*	*Merged List*	*Comparison*
2 3 5 6	1 4		$1 < 2$
2 3 5 6	4	1	$2 < 4$
3 5 6	4	1 2	$3 < 4$
5 6	4	1 2 3	$4 < 5$
5 6		1 2 3 4	
		1 2 3 4 5 6	

教材對應閱讀章節及練習

- 閱讀章節:
  - 5.1,5.3(~Example 3), 5.4
- 對應習題: (可視個人情況定量)
  - 5.1: 3-7, 9-21, 31-34
  - 5.3: 1-4, 7-9
  - 5.4: 7, 9, 44