Chapter 5

遞　迴(Recursion)

5-1 遞迴程序(Recursive Procedure)

✧ 要點：遞迴的特點

1. 遞迴定義(recursive definitions)：一個物件的定義，是以其本身較簡單的版本來定義，此種稱爲遞迴定義。
2. 遞迴程序(recursive procedures)：一個程序或函數，若會呼叫自己本身，或者先呼叫其它程序(此程序也可以再呼叫其它程序)，最後又呼叫回自己，這種程序稱爲遞迴程序。
3. 分類：
   (1) 直接遞迴(directly recursive)：直接呼叫自己的遞迴程序。
   (2) 間接遞迴(indirectly recursive)：先呼叫其它程序，若干層之後，才又呼叫回自己。

✧ 遞迴重要範例 1：n 階乘計算(n Factorial)

➢ 定義：$n!=\begin{cases} 1 & ,n\le 1 \\ n\times(n-1)! & ,n>1 \end{cases}$

➢ 非遞迴方法

```
int fact(int n)
{    int s=1;
     for (int i=1; i<=n; i++) s*=i;
     return s;                              }
```

時間複雜度：迴圈執行 n-1 次，故時間複雜度爲 O(n)。

- 遞迴方法

```
int fact(int n)
{   if (n<=1) return 1;
    else return n*fact(n-1);        }
```

時間複雜度：若以 T(n) 代表計算 fact(n) 所需要的時間函數，有下列遞迴關係式

$$T(n)=\begin{cases}1 & ,n\le 1\\ T(n-1)+1 & ,n>1\end{cases}$$

推導：

$T(n)=T(n-1)+1$ ····· (1)

$T(n-1)=T(n-2)+1$ ····· (2)

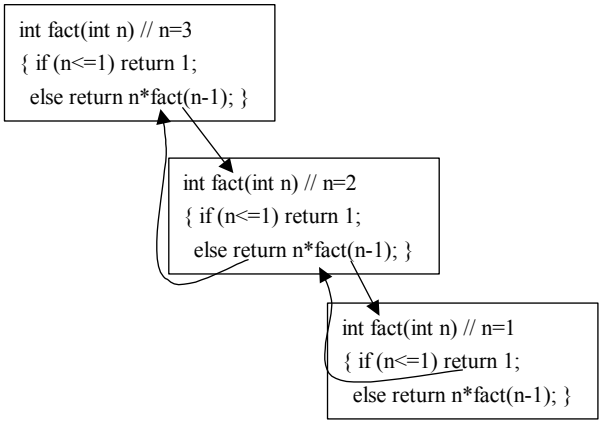
. . . . . .

$T(2)=T(1)+1$ ····· (n-1)

$T(1)=1$ ····· (n)

將(1)+(2)+ …+(n) 可以得到 $T(n)=n$ 。

執行過程



✧ 要點：遞迴程式之重要特性

1. 遞迴程序必須有一條執行路徑不再進行遞迴呼叫，否則會造成無窮遞迴。不再進行遞迴之條件，稱爲終止條件(terminating condition)。
2. 遞迴程式之時間計算，當遞迴程序中無迴路存在時，其時間複雜度將與呼叫次成正比。若有迴路時，必須列出其時間函數，再加以求解。
3. 遞迴程式之空間複雜度，一般是與（最深的）呼叫深度成正比；若遞迴程序中有陣列之局部變數或參數時，所需之空間複雜將更高。

✧ 基本數學範例 1

1. 使用遞迴方式，求 S=1+2+3+…+N。其遞迴定義如下：

$$\sum_{i=1}^{n} i = \begin{cases} 0 & ,n = 0 \\ \sum_{i=1}^{n-1} i + n & ,n > 0 \end{cases}$$

2. 遞迴程式：

```
int sum(int n)
{   if (n<=0) return 0;
    else return sum(n-1)+n;        }
```

3. 時間複雜度：O(n)

✧ 基礎數學範例 2

1. 使用遞迴方式，找出一組數目 a[1]~a[n] 中的最大值。其遞迴定義如下：

$$Max(a[1..n]) = \begin{cases} a[1] & ,n = 1 \\ Max2(Max(a[1..n-1]), a[n]) & ,n > 1 \end{cases}$$

2. 遞迴程式：

```
int max(int a[], int n)
{   if (n==1) return a[1];
```

```
    else
    {   int m=max(a, n-1);
        return a[n]>m?a[n]:m;     }
}
```

3. 時間複雜度：O(n)

✧ 基礎數學範例 3

1. 使用遞迴方式，求一組數目的總和 S=a[1]+a[2]+a[3]+…+a[n]。其遞迴定義如下：

$$\sum_{i=1}^{n} a_i = \begin{cases} \sum_{i=1}^{n-1} a_i + a_n & ,n > 0 \\ 0 & ,n = 0 \end{cases}$$

2. 遞迴程式：

```
int sum(int a[], int n)
{   if (n<=0) return 0;
    else return sum(a, n-1)+a[n];        }
```

3. 時間複雜度：O(n)

✧ 基礎數學範例 4

1. 使用下面定義的函數 pi(m,n) 來計算排列數

$$pi(m,n) = m\times(m+1)\times(m+2)\times\ldots\times n$$
$$= m\times pi(m+1,n)$$

2. 其遞迴定義如下：

$$pi(m,n) = \begin{cases} n & ,m = n \\ m\times pi(m+1,n) & ,m < n \end{cases}$$

3. 遞迴程式：

```
int pi(int m, int n)
{   if (m==n) return n;
    else return m*pi(m+1, n);          }
void main()
{   int n;
```

```
    scanf("%d", &n);
    printf("%d", pi(1,n));          }
```

4. 時間複雜度：O(n)

✧ 基礎數學範例 5

1. 使用遞迴方式，計算級數 S=1-2+3-4+5-6……± n。

   令 sum(m,n) =m-(m+1)+(m+2)-(m+3)+……± n

   =m-[(m+1)-(m+2)+(m+3)+……± n]

   =m-sum(m+1,n)

2. 其遞迴定義如下：

$$sum(m,n)=\begin{cases} n & ,m=n \\ m-sum(m+1,n) & ,m<n \end{cases}$$

3. 遞迴程式：

```
int sum(int a[], int m, int n)
{   if (m==n) return n;
    else return m-sum(m+1, n);      }
void main()
{   int n;
    scanf("%d", &n);
    printf("%d", sum(1,n));         }
```

4. 時間複雜度：O(n)

✧ 基礎數學範例 6

1. 使用遞迴方式，求整數除法 x÷y 的商。其遞迴定義如下：

$$x \div y=\begin{cases} (x-y)\div y+1 & ,x \geq y \\ 0 & ,x<y \end{cases}$$

2. 遞迴程式：

```
int div(int x, int y)
{   if (x>=y) return div(x-y, y)+1;
    else return 0;                  }
```

3. 時間複雜度：O(x/y)

✧ 基礎數學範例 7

1. 使用遞迴方式，求整數除法 x÷y 的餘數。其遞迴定義如下：

$$x \bmod y = \begin{cases} (x-y) \bmod y & , x \geq y \\ x & , x < y \end{cases}$$

2. 遞迴程式：

```
int mod(int x, int y)
{   if (x>=y) return mod(x-y, y);
    else return x;                    }
```

3. 時間複雜度：O(x/y)

✧ 基礎數學範例 8

1. 使用遞迴方式，求兩正整數的乘積 x × y。其遞迴定義如下：

$$x \times y = \begin{cases} x \times (y-1) + x & , y > 0 \\ 0 & , y = 0 \end{cases}$$

2. 遞迴程式：

```
int mul(int x, int y)
{   if (y>0) return mul(x, y-1)+x;
    else return 0;                    }
```

3. 時間複雜度：O(y)

✧ 基礎數學範例 9

1. 使用遞迴方式，求兩正整數的指數計算 x^y。其遞迴定義如下：

$$x^y = \begin{cases} x^{y-1} \times x & , y > 0 \\ 1 & , y = 0 \end{cases}$$

2. 遞迴程式：

```
int exp(int x, int y)
{   if (y>0) return exp(x, y-1)*x;
    else return 1;                    }
```