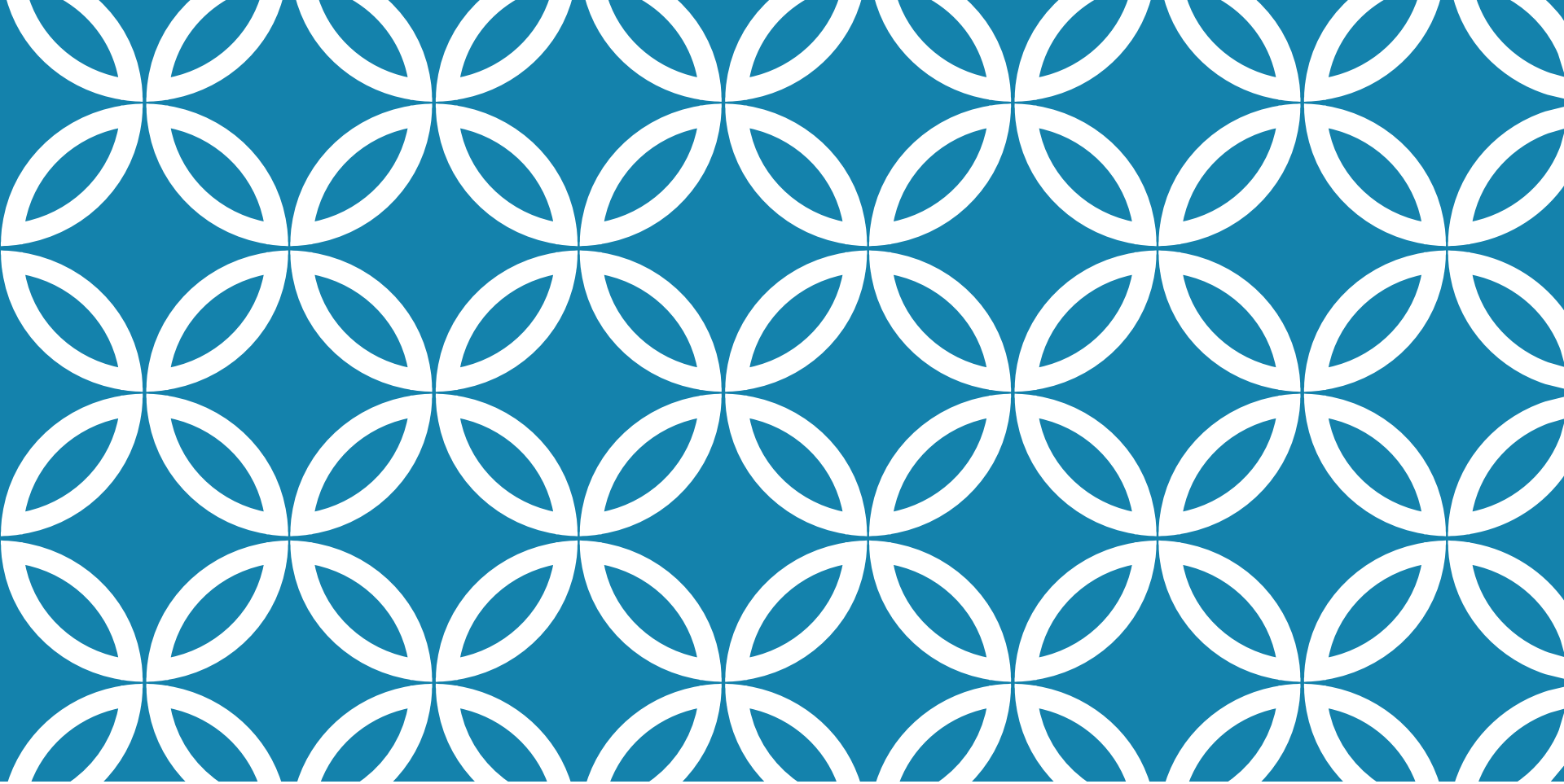


二元樹的走訪

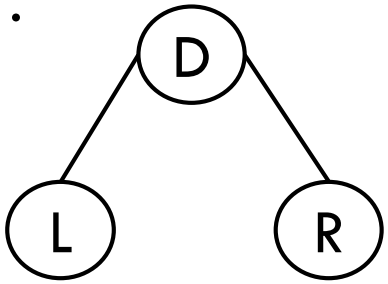
資料結構
鍾宜玲

二元樹的走訪

資料讀取一遍的結果有DLR, DRL, LDR, LRD, RDL及RLD

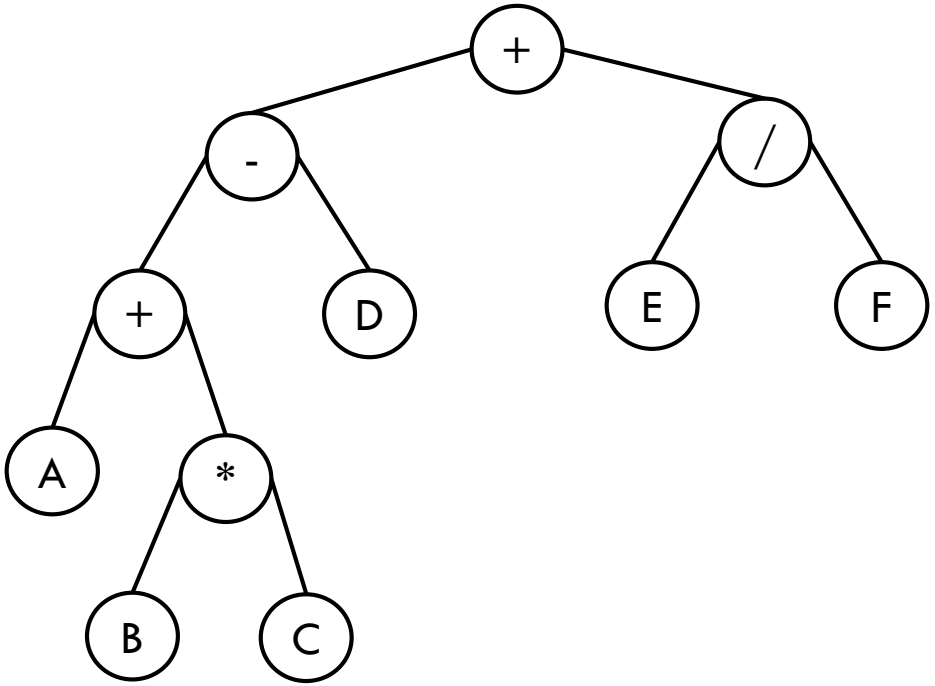
若限制節點的左子樹比右子樹先走訪：

- DLR前序 (preorder)
- LDR中序 (inorder)
- LRD後序 (postorder)



恰與運算式的前序 (prefix)、中序 (infix) 及後序 (postfix) 互相對應。

運算式的二元樹表示法



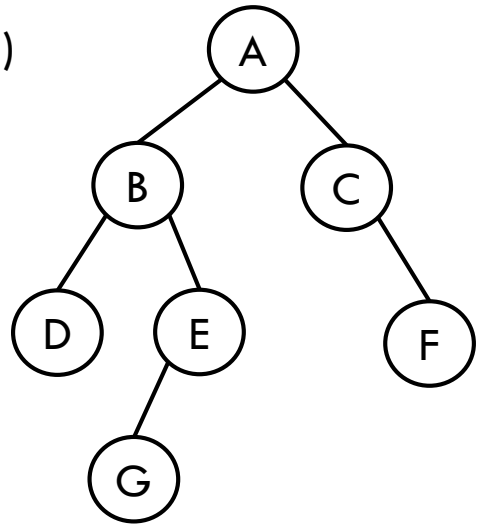
前序: + - + A * B C D / E F

中序: A + B * C – D + E / F

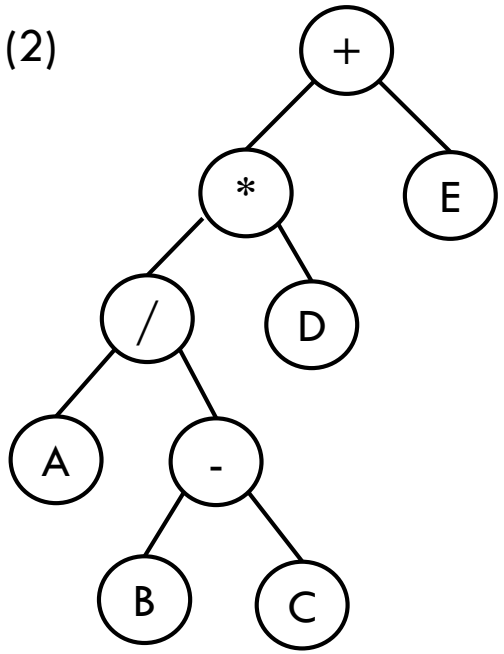
後序: A B C * + D – E F / +

例 寫出下面二棵二元樹的前序、中序、後序走訪順序。

(1)



(2)



解：

(1) 前序： ABDEGCF

中序： DBGEACF

後序： DGEBFCA

(2) 前序： +*/A-BCDE

中序： A/B-C*D+E

後序： ABC-/D*E+

後序走訪函數POST( )

若二元樹的節點定義如下：

```
typedef struct tnode{
      char data;
      struct tnode *left;
     struct tnode *right;
} TNODE;
```

則後序法的函數post( ) 可製作如下：

```
void post(TNODE  *p)
{
   if( p != NULL ) {
        post( p->left );              /*走訪左子樹*/
        post( p->right );             /*走訪右子樹*/
        printf("%c", p->data);        /*走訪樹根節點*/
   }
}
```

二元樹走訪的中序及前序的程式。

中序

```
void inorder(TNODE *p)
{
   if (p!=NULL){
       inorder(p->left);
       printf("%c", p->data);
       inorder(p->right);
    }
}
```

前序：

```
void preorder(TNODE *p)
{
   if (p!=NULL) {
       printf("%c", p->data);
       preorder(p->left);
       preorder(p->right);
    }
}
```

例　某二元樹之前序及中序走訪順序如下：

前序：ABCDEFG　中序：DCBEAFG

(1) 畫出此二元樹

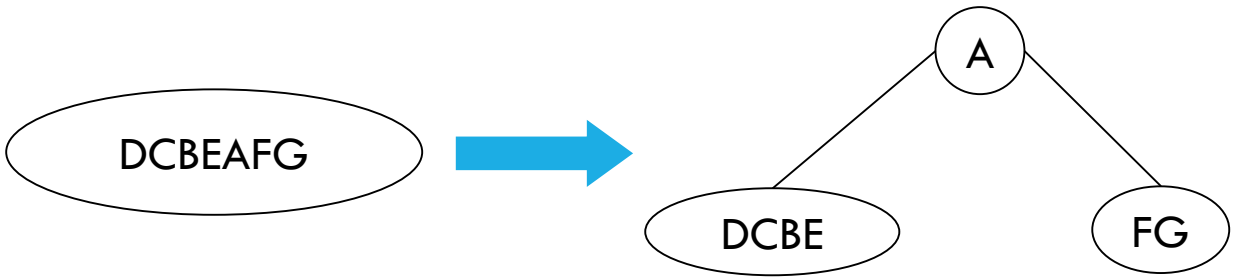
(2) 寫出後序走訪順序。

解：

(1)利用前序找出樹根，再利用中序找出左右子樹，即

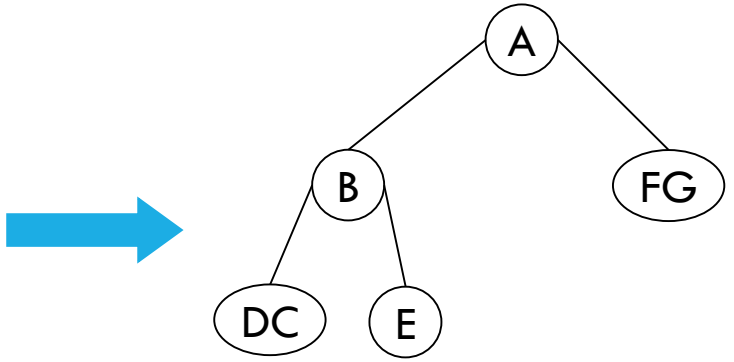
(a) 前序A BCDEF　　中序DCBE A FG

樹根　　左子樹　　右子樹



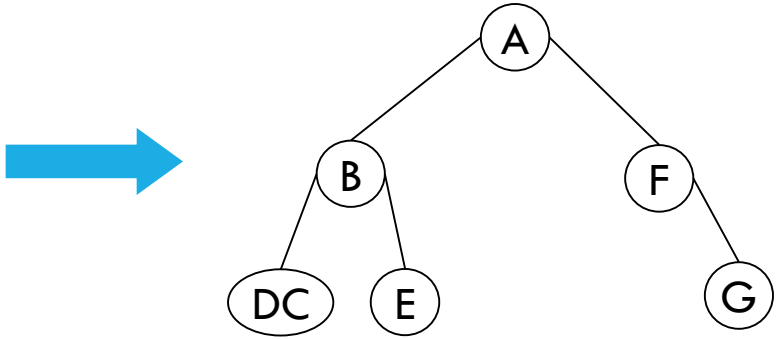
(b) 前序BCDE　　中序DC B E

樹根　　左　右



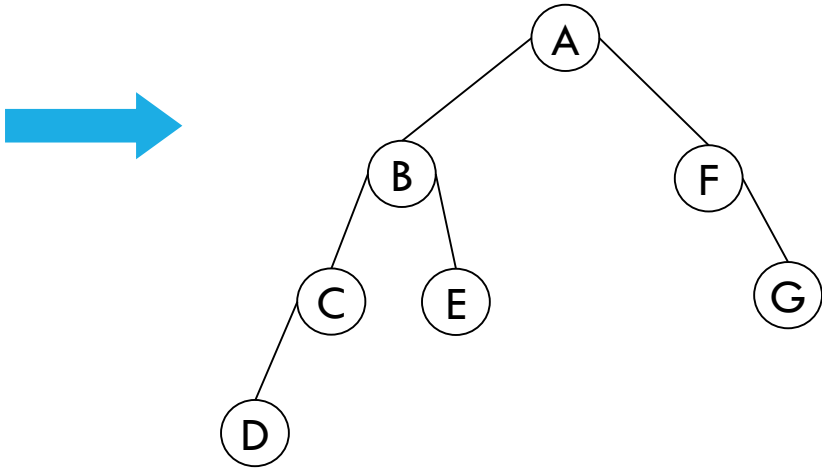
(c) 前序FG　　中序FG

樹根　　右



(d) 前序CD　　中序DC

樹根　　左



(2)後序走訪順序為 DCEBGFA