計算機程式設計
C語言 tree

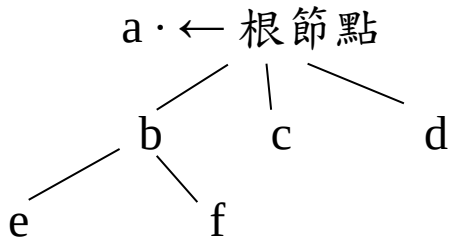
郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

二元樹

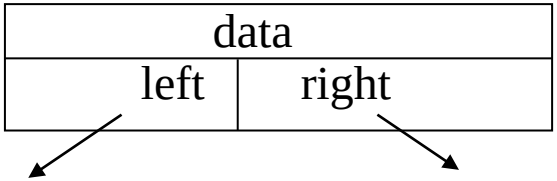
- 樹是一種資料結構
  - 每一顆樹有一個根節點(root node)，
  - 根節點下有零到n個子節點。
  - 子節點也可以擁有自己的子節點。
  - e和f是b的子節點，a是b的父節點。
  - 一個節點最多有n個子節點，則稱n元樹。
  - 某樹一個節點最多有二個子節點，則稱二元樹。
  - 某節點沒有子節點，稱葉節點。
  - 沒有父節點的節點，稱根節點。
  - a是根節點，e，f，c…d是葉節點。



二元樹的節點結構

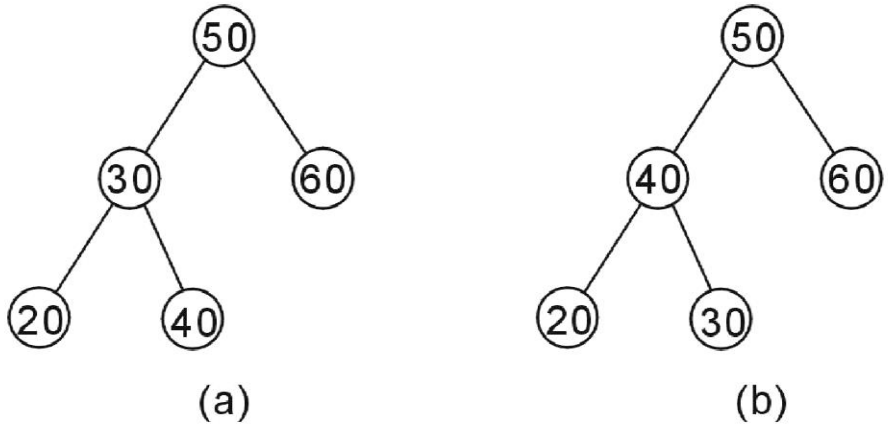
- data欄位存放二元樹節點的基本資料，right和left指向右子樹和左子樹的指標。

```
typedef struct  node_s {
        int    data；
        struct   node_s   * right, * left;
} tree_t;
typedef   tree_t  * btree;
```



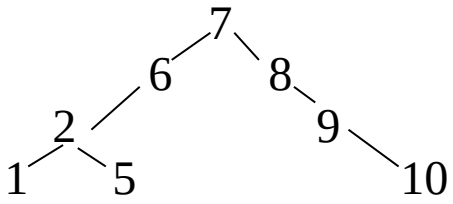
二元搜尋樹

- 二元搜尋樹可以是空集點，假使不是空集合，樹中每一節點（node）均含有一鍵值（key value），且具有下列特性：
  - 在左子樹的所有鍵值均小於樹根/父節點的鍵值。
  - 在右子樹的所有鍵值均大於樹根/父節點的鍵值。
  - 左子樹和右子樹亦是二元搜尋樹。
  - 每個鍵值都不一樣。



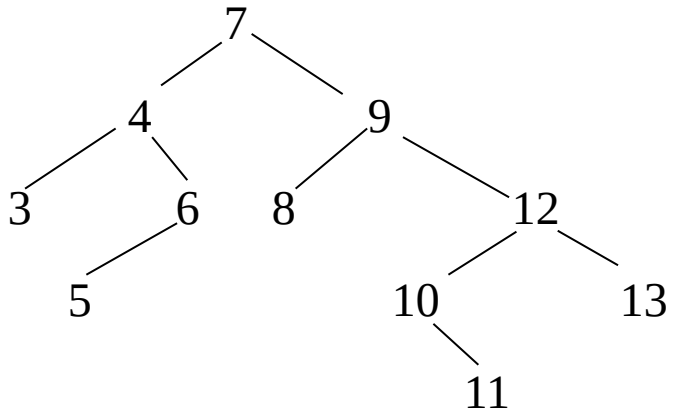
二元搜尋樹的建立

- 二元搜尋樹建立
  - 將第一個欲建的元素放在根節點
  - 比較元素值與節點值，若元素值大於節點值，則將此元素值送往右子節點，若右邊子節點不是NULL則重複比較，否則建立一個新節點存放這筆資料，然後將新節點的右邊子節點和左邊子節點設成NULL。
  - 若元素值小於節點值，將此元素值送往左子節點，若此左子節點不是NULL，則重複比較。否則建立一個新節點存放這筆資料，然後將新節點的右子節點，和左子節點設成NULL。
  - 範例：依資料順序7，6，2，8，9，10，1，5建立樹結構。



二元樹的列印

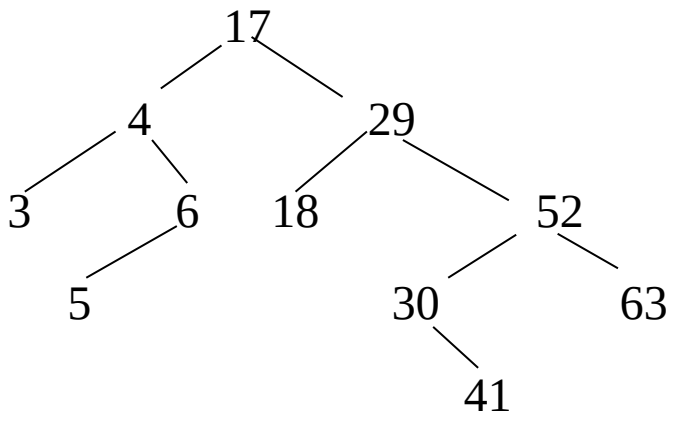
- 線性串列只有從頭到尾或從尾到頭的列印(左子樹->右子樹)
- 二元樹有三種不同列印方式：
  - (父節點)中序(inorder)列印方式。
    - 以從小到大的方式列印
    - 3 4 5 6 7 8 9 10 11 12 13
  - 前序（preorder）列印方式。
  - 後序（postorder）列印方式。




```
void inorder(btree root) {
    if (root!=NULL) {
        inorder(root->left);
        printf("\2: %d\n", root->data);
        inorder(root->right);
    }
}
```

二元樹的列印

```
btree create_btree(btree root, int val)  {
      btree newnode, current, back;
      newnode = (btree) malloc(sizeof(tree_t));
      newnode->data = val;
      newnode->left =  newnode->right = NULL;
      if ( root == NULL )    {       /* insert root node  */
        root = newnode;         return root;
      }
      else    {                    /* insert other node */
          current = root;
          while ( current != NULL )    {
              back = current;
              if ( current->data > val )    current = current->left;
              else                current = current->right;
          }
          if ( back->data > val )       back->left = newnode;
          else            back->right = newnode;
      }
      return root;
}
```



二元樹的列印

```
int main()  {
    int   arr[] = { 7, 4, 1, 5, 12, 8, 13, 11 };
    btree ptr;
    int   val, i;
    ptr = NULL;              /* initial the root pointer */
    printf("\1: Create binary tree for following value.\n");
    for ( i = 0; i < 8; i++ )    {
        ptr = create_btree(ptr,arr[i]);
        printf("\2: %d\n",arr[i]);
    }
    printf("\1: Using inorder to print the binary tree.\n");
    inorder(ptr);
}
```

```
☺: Create binary tree for following value.
☻: 7
☻: 4
☻: 1
☻: 5
☻: 12
☻: 8
☻: 13
☻: 11
☺: Using inorder to print the binary tree.
☻: 1
☻: 4
☻: 5
☻: 7
☻: 8
☻: 11
☻: 12
☻: 13
Press any key to continue
```

二元樹的列印

- 前序列印
  - 每個節點皆會比它左子節點及右子節點先列印，左子節點又比右子節點有更高優先順序。
  - 若以前序列印方式，得到結果：7，4，3，6，5，9，8，12，10，11，13
- 後序(postorder)列印
  - 在列印某節點前，先列印左節點，然後右節點，等到兩個子節點列印完後，才列印此節點。
  - 列印此資料結構，得到結果：3，5，6，4，8，11，10，13，12，9，7

```
void postorder(btree root) {
    if (root!=NULL)  {
        postorder(root->left);
        postorder(root->right);
        printf("\2: %d\n",root->data);
    }
}
```

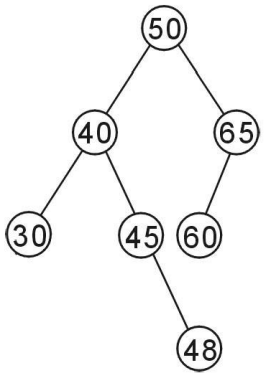
二元樹的列印

```
void preorder(btree root)   {
     if ( root != NULL )    {
          printf("\2: %d\n",root->data);
          preorder(root->left);
          preorder(root->right);
     }
}
int main()  {
     int   arr[] = { 7, 4, 1, 5, 12, 8, 13, 11 };
     btree ptr;
     int   val, i;
     ptr = NULL;              /* initial the root pointer */
     printf("\1: Create binary tree for following value.\n");
     for ( i = 0; i < 8; i++ )     {
       ptr = create_btree(ptr,arr[i]);
       printf("\2: %d\n",arr[i]);
     }
     printf("\1: Using preorder to print the binary tree.\n");
     preorder(ptr);
}
```

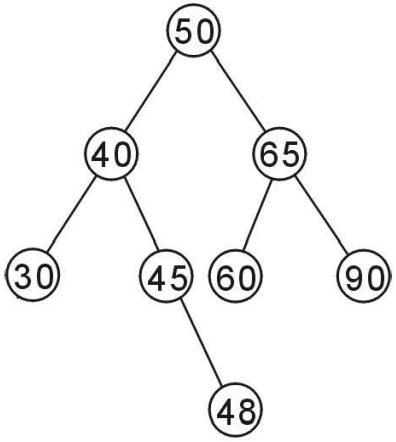
```
☺: Create binary tree for following value.
☻: 7
☻: 4
☻: 1
☻: 5
☻: 12
☻: 8
☻: 13
☻: 11
☺: Using preorder to print the binary tree.
☻: 7
☻: 4
☻: 1
☻: 5
☻: 12
☻: 8
☻: 11
☻: 13
Press any key to continue
```

二元搜尋樹加入與刪除

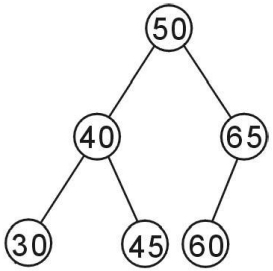
1. 今欲加入48



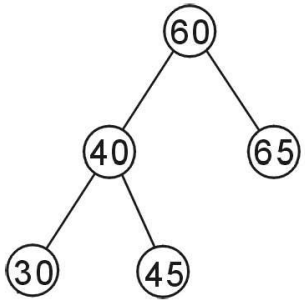
2. 繼續加入90則



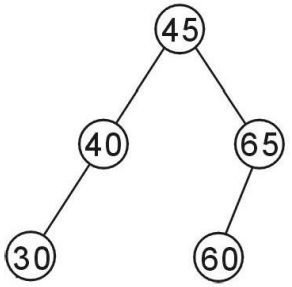
3. 刪除某一節點時，若刪除的是樹葉節點，則直接刪除之，假若刪除不是樹葉節點，則在左子樹找一最大的節點或在右子樹找一最小的節點，取代將被刪除的節點



4. 刪除50，則可用下列二種方法之一：



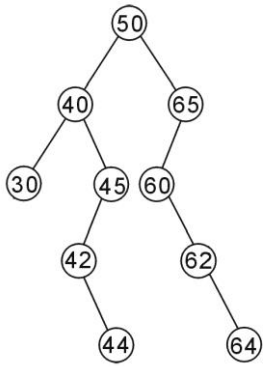
（以右子樹最小的節點取代）



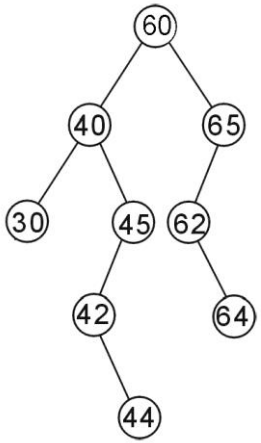
（以左子樹最大的節點取代）

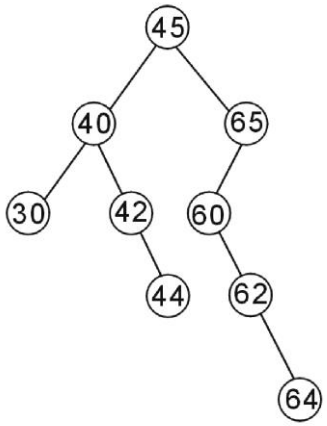
二元搜尋樹加入與刪除

5. 若取代的節點有右子樹或左子樹時，則必須加以調整其子節點



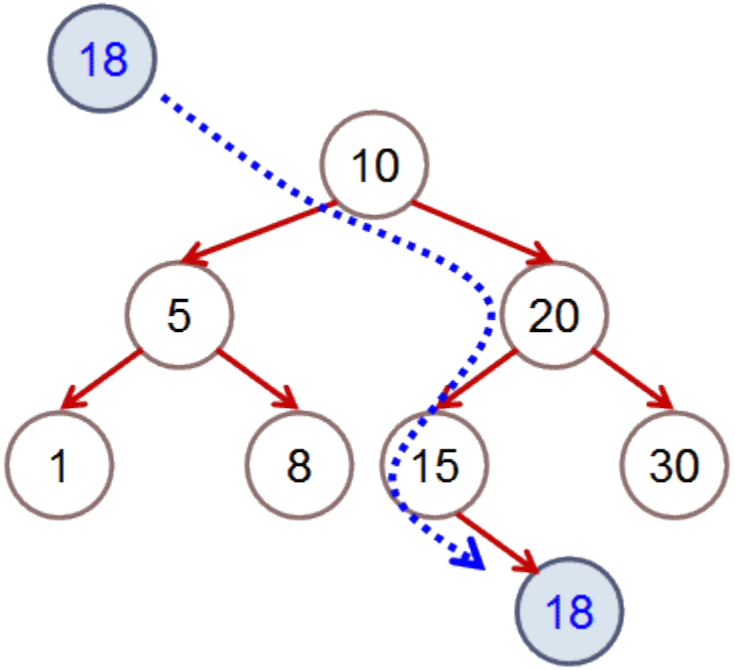
- 今欲刪除50的兩種做法





二元搜尋(Binary Search)

- 適用於已排序完成資料之搜尋
- 如欲搜尋18，藍色虛線是搜尋路徑



Homework I

- 建構唯一二元樹
  - 給定(1)中序，(2) 前序或後序，產生唯一個Binary Tree，依序印出Tree的內容，印出順序，Tree元素由上而下，由左而右印出。
  - 假設一二元樹如下圖
    - A
    - B C
    - D E F G
    - (A為根節點，B為A之左子樹，C為A之右子樹)
  - 前序(Pre-order)：
    - 1.訪問根節點
    - 2.訪問左子樹
    - 3.訪問右子樹
    - 上圖的走訪順序為：ABDECFG

Homework I

- 中序(In-order)：
  - 1.訪問左子樹
  - 2.訪問根節點
  - 3.訪問右子樹
  - 上圖的走訪順序為：DBEAFCG
- 後序(Post-order)：
  - 1.訪問左子樹
  - 2.訪問右子數
  - 3.訪問根節點
  - 上圖的走訪順序為：DEBFGCA
- 前序代碼：P
- 中序代碼：I
- 後序代碼：O

Homework I

- *每次輸入一定有一個是中序。(中序搭配前序，或是中序搭配後序，不會有前序搭配後序)。
- 輸入說明
  - 第一筆輸入前序、中序或後序代碼。
  - 第二筆輸入上一筆輸入種類尋訪的結果，大寫英文字母。
  - 第三筆輸入前序、中序或後序代碼。
  - 第四筆輸入上一筆輸入種類尋訪的結果，大寫英文字母。
- 輸出說明
  - 輸出唯一二元樹的內容，由上而下，由左而右。

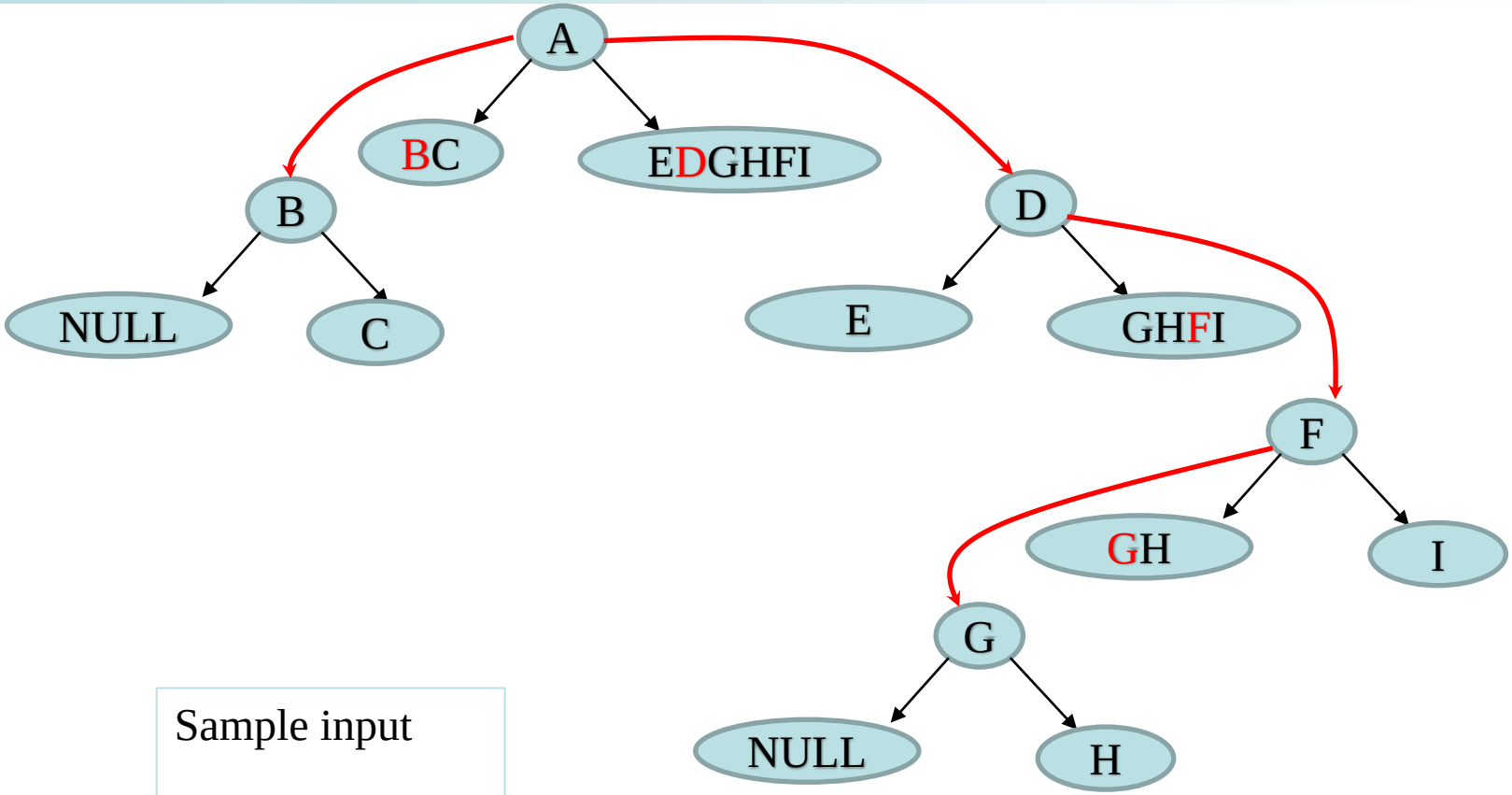
Sample input

P (前)
ABCDEFGHI
I (中)
BCAEDGHFI

Sample output

ABDCEFGIH

Homework I



Sample input

P (前 - 根左右)
ABCDEFGHI
I (中 – 左根右)
BCAEDGHFI