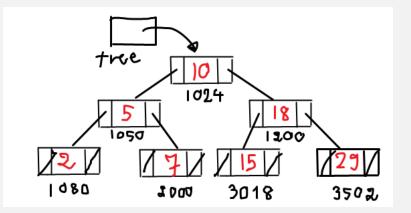


<u>ทบทวน</u>

- 1. นิยามโครงสร้างต้นไม้
- 2. วิธีการ insert ของ Binary Search Tree :
 - BigO
- 3. การ print
 - Preorder
 - Inorder
 - Postorder





<u>เนื้อหา</u>

- 1. การ findmin
- 2. Delete Tree:
 - ไม่มีลูก
 - ลูก 1 ด้าน
 - ลูก 2 ด้าน
- 3. Expression Trees

5

6

```
struct node *find_min(struct node *tree)
                                              tree
    if(tree==NULL)
                                               1024
       return NULL;
    else
       if( tree->left == NULL )
                                                 1024
         return tree;
      else
        return (find min(tree->left));
```

3

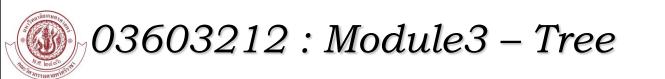
5

6

```
tree
struct node *find_min(struct node *tree)
                                             1024
    if(tree==NULL)
                                           2000
       return NULL;
    else
       if( tree->left == NULL )
                                                1024
         return tree;
      else
                                         1050
                                                       1080
         return (find _min(tree->left));
```

```
    2
    3
    4
    6
    7
```

```
2000
struct node *find_min(struct node *tree)
    if(tree==NULL)
                                                1024
       return NULL;
    else
                                         1050
                                                      1080
       if( tree->left == NULL )
         return tree;
                                     3000
      else
         return (find min(tree->left));
```



Big O ของการ find min



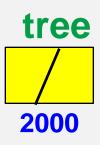
<u>Delete</u>

- 1. No Child:
- 2. One Child:
 - นำลูกที่เหลือมาแทนโหนดที่ถูกลบ
- 3. Two Childs:
 - นำลูกด้านขวาที่มีค่าน้อยที่สุดมาแทนโหนดที่ถูกลบ



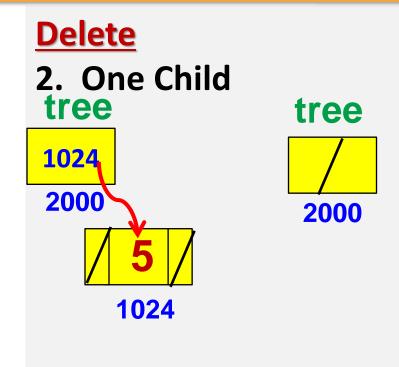
Delete

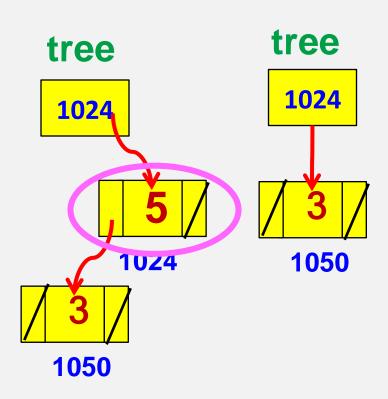
1. No Child



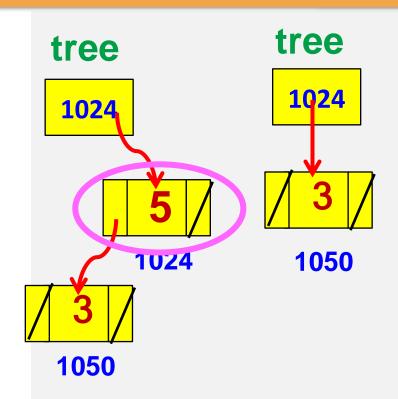
```
if (tree==NULL)
    cout << "No node";
return tree;</pre>
```











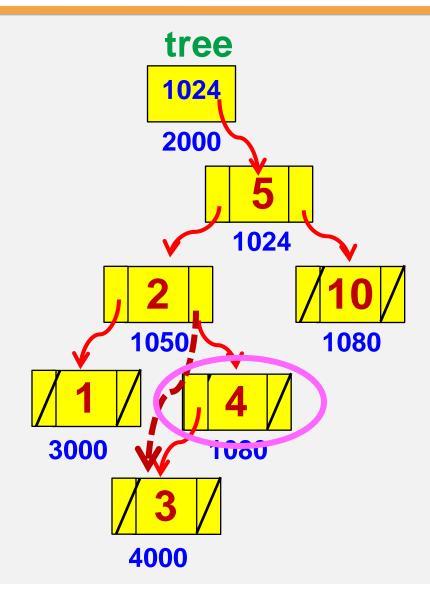
```
tmpcell=tree;
if( tree->left == NULL )
        child = tree->right;
if(tree->right == NULL)
        child = tree->left;
delete(tmpcell);

return child;
```



<u>Delete</u>

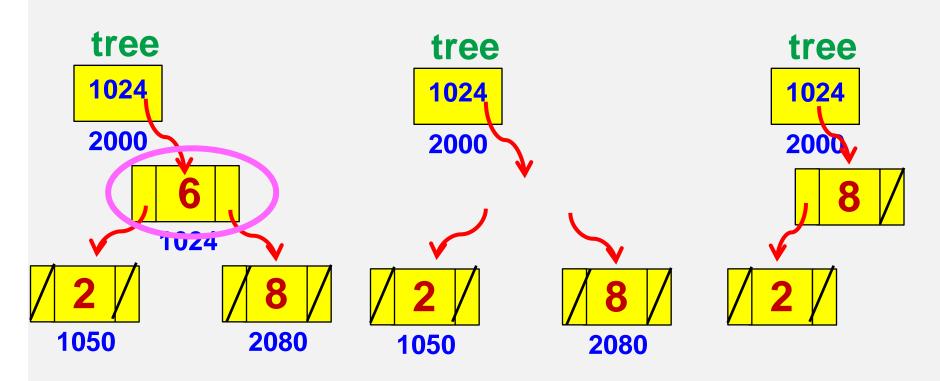
2. One Child





Delete

3. 2 Childs

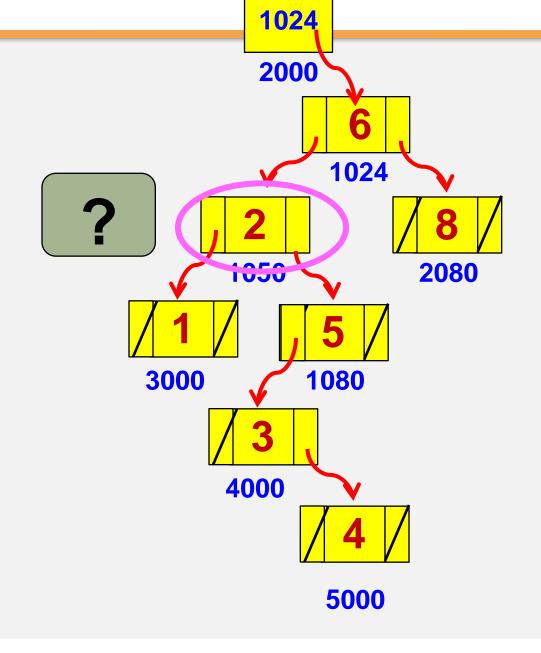




03603212 : Module3 – Tree<mark>_tree</mark>

Delete

3. 2 Childs



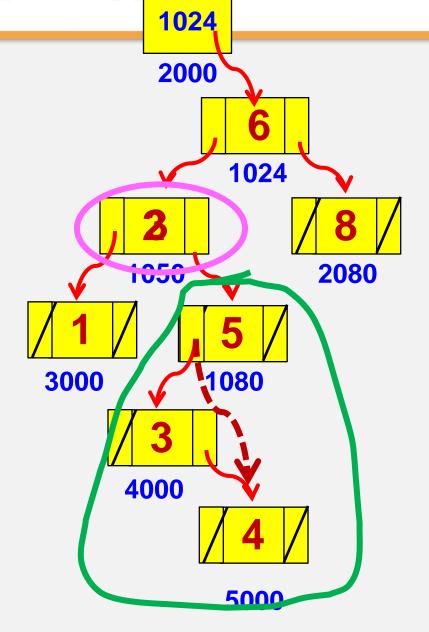


03603212 : Module3 - Tree tree

Delete

3. 2 Childs

- นำลูกทีเป็น Min right subtree มาแทนที่
 - recursive delete

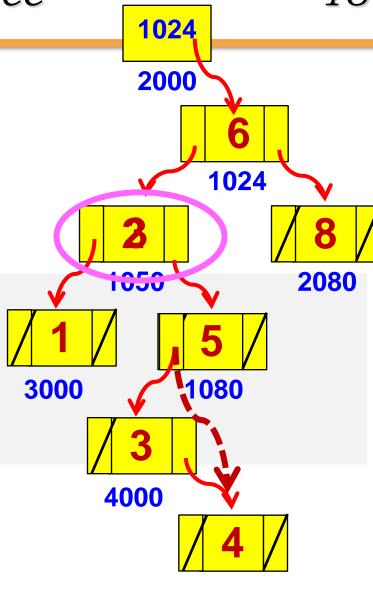


Delete

3. 2 Childs

- นำลูกทีเป็น Min right subtree มาแทนที่

```
tmpcell=find_min(tree->right);
tree->value = tmpcell->value;
tree->right =
    dTree(tree->right,tree->value);
```



5000

tree



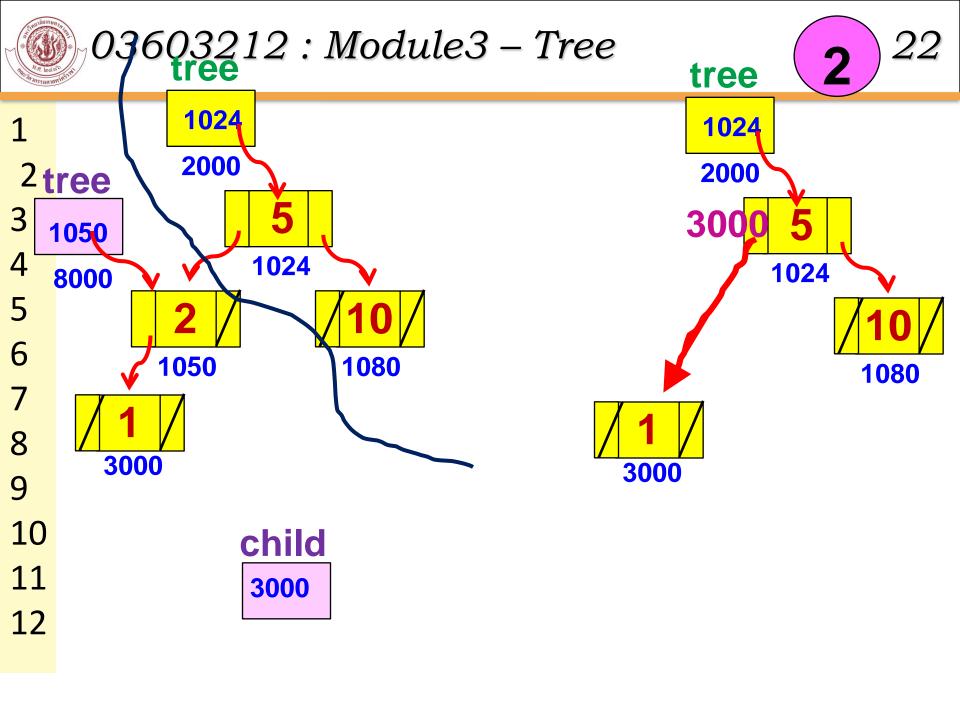
```
struct node *dTree(struct node *tree,int x)
                                                          tmpcell
     { struct node *tmpcell, *child;
                                                 tree
       if (tree==NULL)
                                                 1024
           printf("No Node\n");
                                                 2000
       else
       { if(x < tree->value)
           tree->left = dTree(tree->left, x);
                                                      1024
         else
                                                          child
           if( x > tree->value)
               tree->right=dTree(tree->right,x);
10
           else
11
               if( tree->left !=NULL&& tree->right!=NULL)
12
```

```
if( tree->left !=NULL&& tree->right!=NULL)
12
    { tmpcell=find_min(tree->right);
13
       tree->value = tmpcell->value;
14
                                                          tmpcell
15
       tree->right = dTree(tree->right,tree->value);
                                                 tree
                                                              1024
    else
                                                 1024
18
        tmpcell=tree;
                                                2000
19
        if( tree->left == NULL )
20
            child = tree->right;
21
                                                                child
        if(tree->right ==NULL)
                                                      1024
22
             child = tree->left;
23
        delete(tmpcell);
24
        return child;
    } /* end else tree is not NULL */
27
     return tree;
    }/* end function */
```

Recursiveครั้งที่1 1050 2

```
struct node *dTree(struct node *tree,int x)
   { struct node *tmpcell, *child;
                                         tree
      if (tree==NULL)
                                         1050
         printf("No Node\n");
                                         8000
      else
     { if(x < tree->value)
                                                 1050
          tree->left = dTree(tree->left, x);
         else
                                             3000
           if( x > tree->value)
10
               tree->right=dTree(tree->right,x);
11
           else
12
               if( tree->left !=NULL && tree->right !=NULL)
```

```
if( tree->left !=NULL && tree->right !=NULL )
12
    { tmpcell=find_min(tree->right);
13
       tree->value = tmpcell->value;
                                                         tmpcei
14
15
       tree->right = dTree(tree->right,tree->value);
                                                              1050
                                            tree
    else
                                            1050
18
        tmpcell=tree;
                                             8000
19
        if( tree->left == NULL )
20
            child = tree->right;
21
                                                     1050
        if(tree->right ==NULL)
22
             child = tree->left;
                                                               child
23
        delete(tmpcell);
                                                                3000
24
        return child;
    } /* end else tree is not NULL */
27
     return tree;
    } /* end function */
```





```
struct node *dTree(struct node *tree,int xtree
  { struct node *tmpcell, *child;
                                                 1024
      if (tree==NULL)
3
                                                 2000
        printf("No Node\n");
      else
                                                      1024
     { if(x < tree->value)
          tree->left = dTree(tree->left, x);
        else
                                               1050
                                                             1080
           if( x > tree->value)
10
               tree->right=dTree(tree->right,x);
11
           else
12
               if( tree->left !=NULL && tree->right !=NULL )
```

```
if( tree->left && tree->right)
12
    { tmpcell=find_min(tree->right);
13
       tree->value = tmpcell->value;
14
15
       tree->right = dTree(tree->right,tree->value);
                                               tree
                                                                tmpcell
    else
18
                                               1024
        tmpcell=tree;
                                                                  1080
19
        if( tree->left == NULL )
                                               2000
20
            child = tree->right;
21
        if(tree->right ==NULL)
                                                    1024
22
             child = tree->left;
23
        delete(tmpcell);
24
        return child;
                                             1050
                                                            1080
    } /* end else tree is not NULL */
27
     return tree;
    } /* end function */
```

Application

Expression Tree

- Leaves are operand
- Nonleaves are operator

a + b

