Al-Mustaqbal University
College of Science

جامـــــعة المـــــستـقبـل
AL MUSTAQBAL UNIVERSITY

كلية العلـوم

قسـم علوم الذكاء الاصطناعي

المحاضرة السادسة

Binary Search Tree(Deletion Operation & Printing the tree)

المادة: Searching and Sorting Algorithms
المرحلة: الثانية
اسم الاستاذ: م.م اية محمد حسين محمد علي

3.3 Deletion Operation in BST

Deleting a node from Binary search tree has following three cases...

Deleting a node from Binary search tree has following three cases...

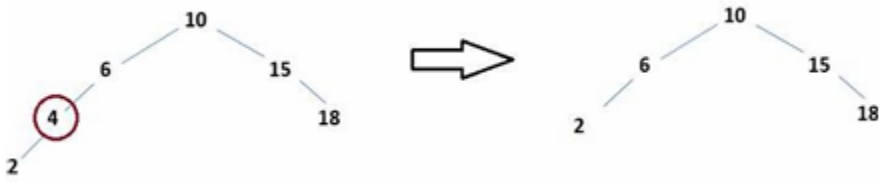
Case 1: Deleting a Leaf node (A node with no children)

Case 2: Deleting a node with one child

Case 3: Deleting a node with two children

Case 1: Deleting a leaf node Step 1: Find the node to be deleted. Step 2: Delete the node and make the father node point to null.

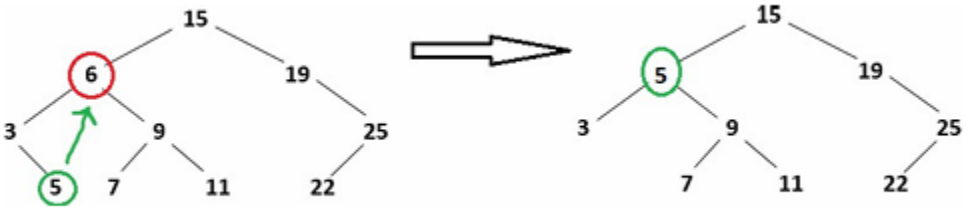
Case 2: Deleting a node with one child: Step 1: Find the node to be deleted. Step 2: Create a link between its parent and child node.



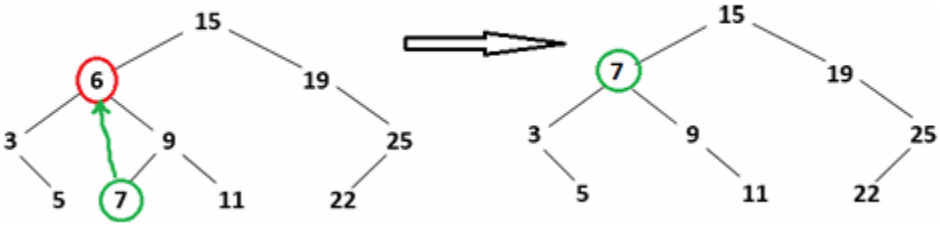
Case 3: Deleting a node with two children Step 1: Find the node to be deleted.

Step 2: find the max node in its left subtree, OR the min node in its right subtree.

Example: Delete node 6



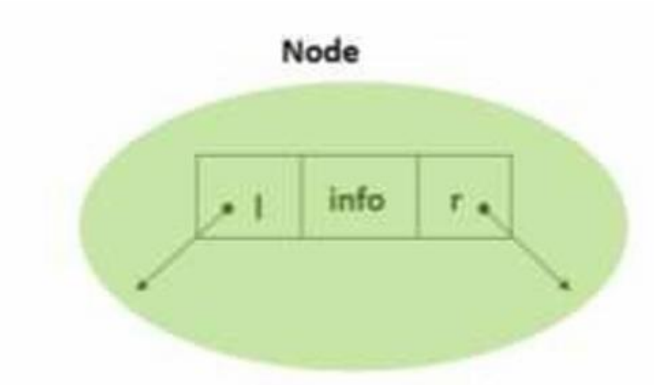
Or



Exercise: Delete the node 15.

Declaration of Tree by using linked list

نفس طريقة تعريف القائمة الموصولة الفرق اضافة حقل جديد ليصبح حقل للفرع الايمن وحقل للفرع الايسر من نوع مؤشر




```
struct node
{
   int info;
   struct node *l, *r;
};

typedef struct node *nodeptr;
```

Create the BST

```
void creat(nodeptr& t)
{
   char ch;
```

```
    if(t == 0)
    {
       t = new node();//                    عملية خلق اول عقدة في الشجرة(الجذر)
       cin >> t->info;

       t->l = 0;
       t->r = 0;

       cout << "Do you want to add from left (" << t->info << ") (Y,N): ";        //اضافه عقده في جهه اليسرى
       cin >> ch;

       if(ch == 'y')
          creat(t->l);

       cout << "Do you want to add from right (" << t->info << ") (Y,N): ";       //اضافه عقده في جهه اليمنى
       cin >> ch;

       if(ch == 'y')
          creat(t->r);
    }
}
```

Printing the tree by using of the following:

1. In-Order Traversal

```
void inorder(nodeptr t)

{

   if(t != 0)

   {

      inorder(t->l);

      cout << t->info << " ";

      inorder(t->r);

   }

}
```

2. Pre-Order Traversal

```
void preorder(nodeptr &t)
{
     if(t != 0)
     {
          cout << t->info << " ";
          preorder(t->l);
          preorder(t->r);
     }
}
```

3. Post-Order Traversal

```
void postorder(nodeptr t)
{
     if(t != 0)
     {
          postorder(t->l);
          postorder(t->r);
          cout << t->info << " ";
     }
}
```