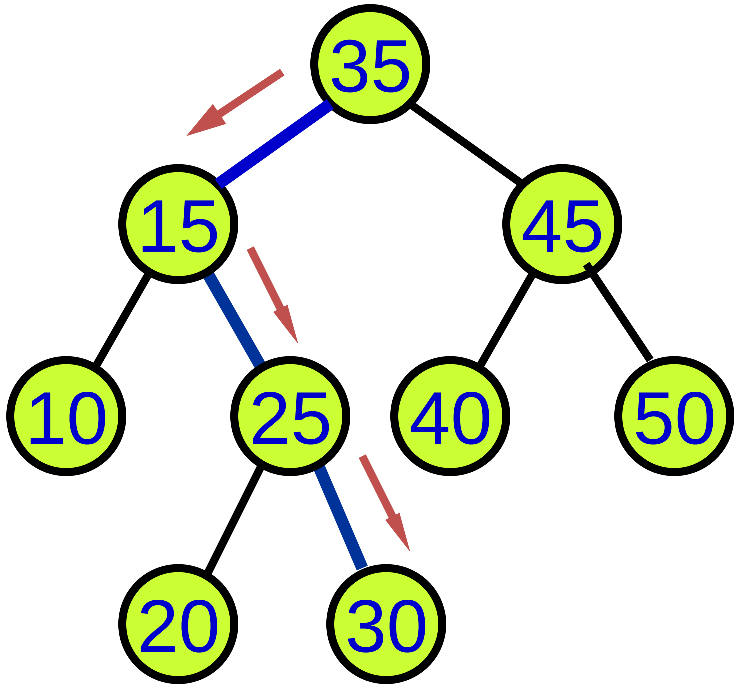
二叉查找树的插入

二叉查找树的插入

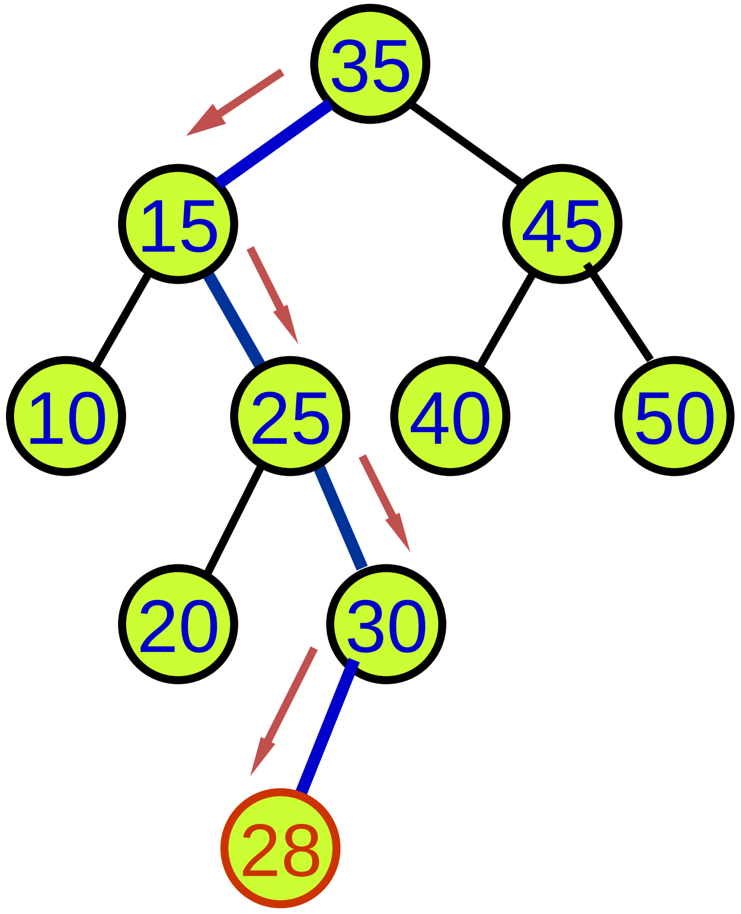
- 每次结点的插入，都要从根结点出发搜索插入位置，如果已经存在该结点，不应插入。



插入结点30，**已经存在，不应插入**

二叉查找树的插入

- 新结点在原二叉树中不存在，搜索插入位置，把新结点作为叶结点插入。



插入新结点28

Binary Search Trees

- implementations
  - **Insertion into a Binary Search Tree**

```
template <class Record>
Error_code Search_tree<Record> :: insert(const
   Record &new_data)
{
   return search_and_insert(root, new_data);
}
```

Binary Search Trees

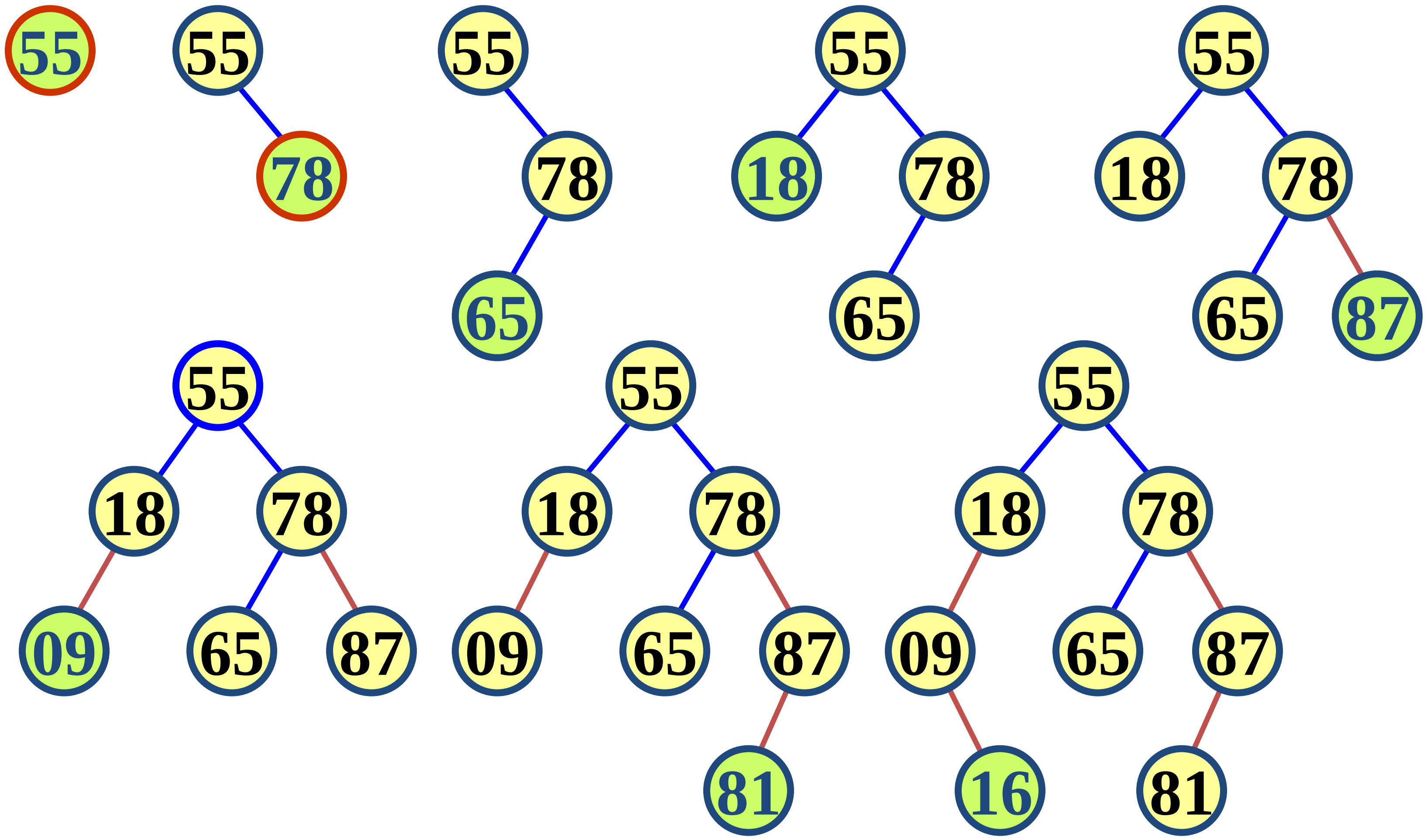
- implementations:
  - **Insertion into a Binary Search Tree**

```
template <class Record>
Error_code Search_tree<Record> :: search_and_insert(
Binary_node<Record> * &sub_root, const Record &new_data)
{
    if (sub_root == NULL) {
    sub_root = new Binary_node<Record>(new_data);
    return success;
    }
    else if (new_data < sub_root->data)
      return search_and_insert(sub_root->left, new_data);
    else if (new_data > sub_root->data)
      return search_and_insert(sub_root->right, new_data);
    else return duplicate_error;
}
```

Binary Search Trees

- implementations:
  - **Insertion into a Binary Search Tree**
    - Insert方法在将新结点插入在一棵具有n个结点的随机二叉查找树时，算法的时间复杂度为 *O*(log *n*)
    - 但是也可能在极端的情况下，这些二叉查找树退化为单枝树等极端情况，这时候插入算法时间复杂度为 O(n).

输入数据 { 55, 78, 65, 18, 87, 09, 81, 16 }



- If the keys are inserted in sorted order(有序次序） into an empty tree, however, this degenerate case will occur.