Binary Search Trees (BSTs)

CSC212: Data Structures

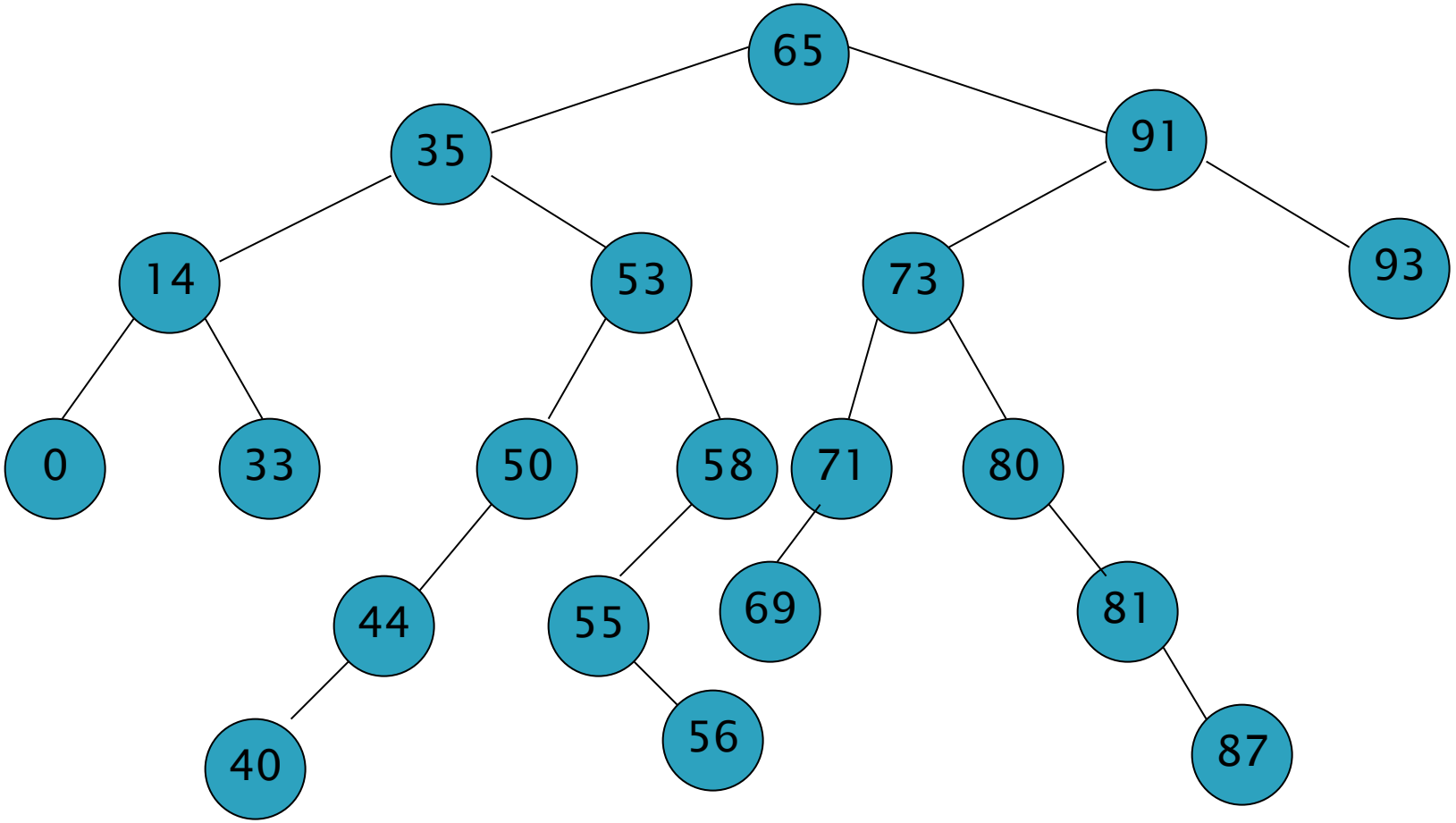
Binary Search Trees (BSTs)

- Consider the search operation **FindKey**: find an element of a particular key value in a binary tree.
  - **In binary tree this operation is O(n).**
    - In a binary tree of 10^6 nodes → 10^6 steps required.
  - **In a Binary Search Tree (BST) this operation can be performed very efficiently: $O(\log_2 n)$.**
    - A binary search tree of 10^6 nodes → $\log_2(10^6) \cong 20$ steps only are required.

Binary Search Trees (BSTs)

- A Binary Search Tree (BST) is a binary tree such that for each node, say N, the following statements are true:
  1. If L is any node in the left subtree of N, then L is less than N.
  2. If R is any node in the right subtree of N, then R is greater than N.

Binary Search Tree (BST): Example



ADT Binary Search Tree

Elements: the elements are nodes (BSTNode), each node contains the following data type: Type.

Structure: hierarchical structure; each node can have two children: left or right child; there is a root node and a current node. If N is any node in the tree, nodes in the left subtree $<$ N and nodes in the right subtree $>$ N.

Domain: the number of nodes in a BST is bounded; type/class name is BST

ADT Binary Search Tree

Operations:

1. **Method** FindKey (int tkey, boolean found).
   requires: none.
   input: tkey.
   results: If bst contains a node whose key value is tkey, then that node is made the current node and found is set to true; otherwise found is set to false and either the tree is empty or the current node is the node to which the node with key = tkey would be attached as a child if it were added to the BST.
   output: found.

ADT Binary Search Tree

2. **Method** Insert (int k, Type e, boolean inserted)
   requires: Full (bst) is false. **input**: key, e.
   results: if bst does not contain k then inserted is set to true and node with k and e is inserted and made the current element; otherwise inserted is set to false and current value does not change.
   output: inserted.
3. **Method** Remove_Key (int tkey, boolean removed)
   input: tkey
   results: Node with key value tkey is removed from the bst and removed set to true. If BST is not empty then root is made the current. **output**: removed

ADT Binary Search Tree

4. **Method** Update(int key, Type e, boolean updated)

   requires: Empty(bst) is false. **input**: key, e. **results**: current node's element is replaced with e. **Output**: updated.

These operations have the same specification as ADT Binary Tree.

5. **Method** Traverse (Order ord)
6. **Method** DeleteSub ( )
7. **Method** Retrieve (Type e)
8. **Method** Empty (boolean empty).

ADT Binary Search Tree

9. **Method** Full (boolean full)

ADT Binary Search Tree: Element

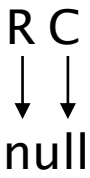
```
public class BSTNode <T> {
   public int key;
   public T data;
   public BSTNode<T> left, right;

   /** Creates a new instance of BSTNode */
   public BSTNode(int k, T val) {
         key = k;
         data = val;
         left = right = null;
   }

   public BSTNode(int k, T val, BSTNode<T> l, BSTNode<T> r) {
         key = k;
         data = val;
         left = l;
         right = r;
   }
}
```

ADT Binary Search Tree: Implementation

```
public class BST <T> {
   BSTNode<T> root, current;

   /** Creates a new instance of BST */
   public BST() {
        root = current = null;
   }

   public boolean empty() {
        return root == null;
   }

   public boolean full() {
        return false;
   }

   public T retrieve () {
        return current.data;
   }
```

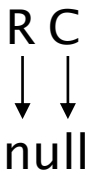
ADT Binary Search Tree: Implementation

```
public class BST <T> {
   BSTNode<T> root, current;

   /** Creates a new instance of BST */
   public BST() {
        root = current = null;
   }

   public boolean empty() {
        return root == null;
   }

   public boolean full() {
        return false;
   }

   public T retrieve () {
        return current.data;
   }
```

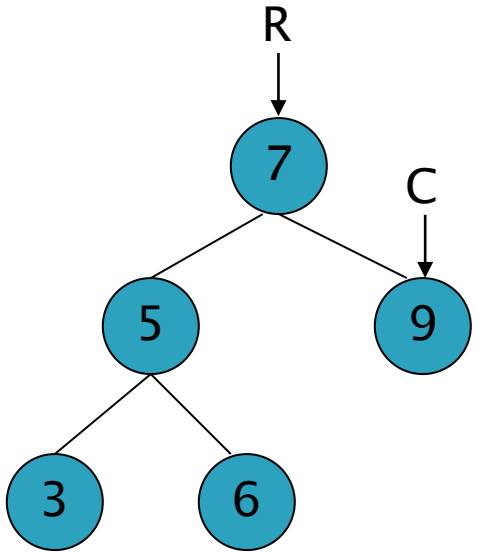


ADT Binary Search Tree: Implementation

```
public class BST <T> {
   BSTNode<T> root, current;

   /** Creates a new instance of BST */
   public BST() {
        root = current = null;
   }

   public boolean empty() {
        return root == null;
   }

   public boolean full() {
        return false;
   }

   public T retrieve () {
        return current.data;
   }
```



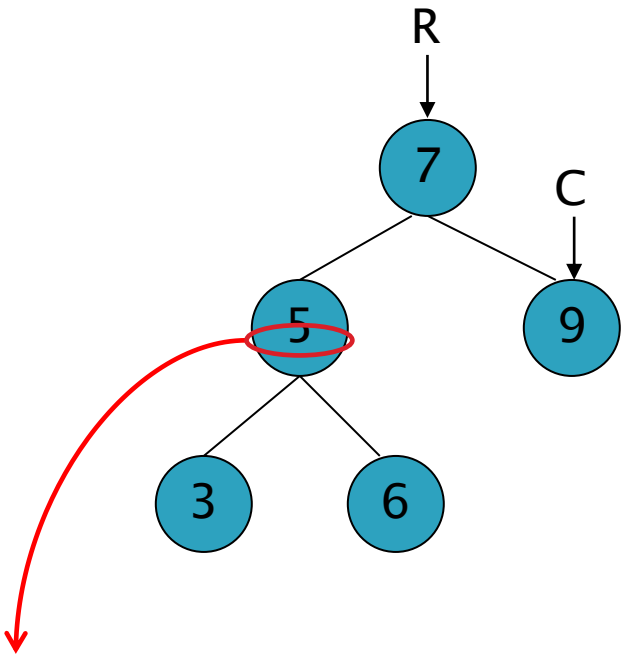
true



false

ADT Binary Search Tree: Implementation

```
public class BST <T> {
   BSTNode<T> root, current;

   /** Creates a new instance of BST */
   public BST() {
        root = current = null;
   }

   public boolean empty() {
        return root == null;
   }

   public boolean full() {
        return false;
   }

   public T retrieve () {
        return current.data;
   }
```



BST: Searching

- The search operation in a binary search tree can be carried out as:

  While (the target element is not found and there is more tree to search) do
  if the target element is "less than" the current element then search the left subtree else search the right subtree.

ADT Binary Search Tree: Implementation

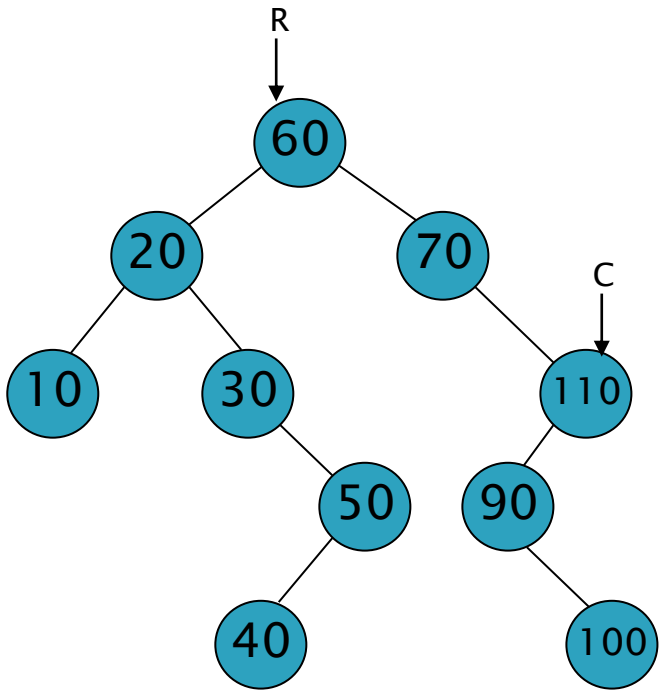
```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

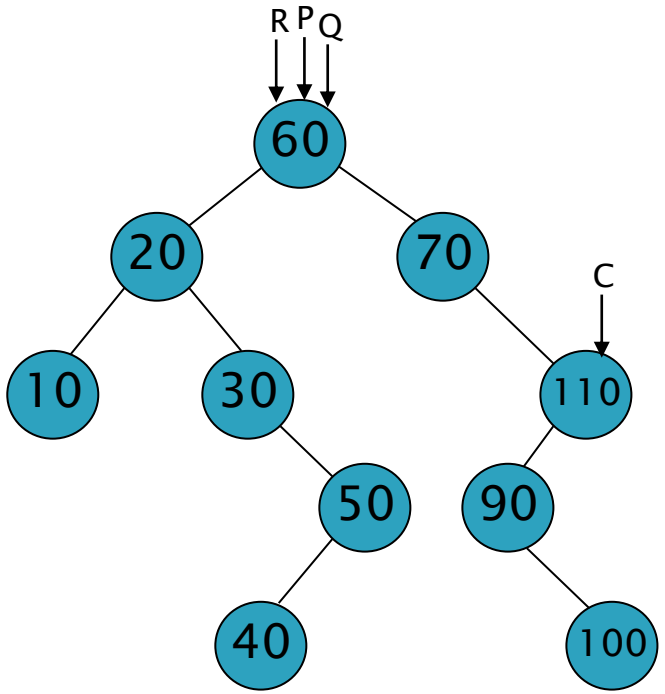


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

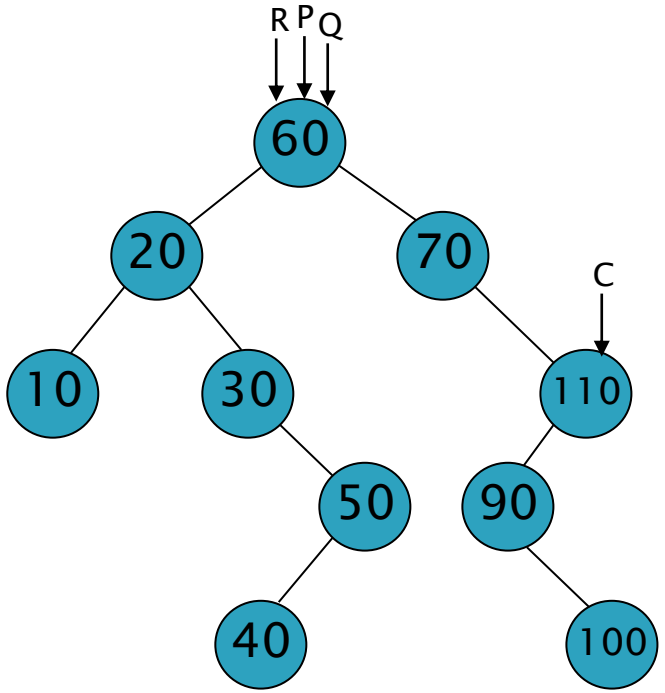


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

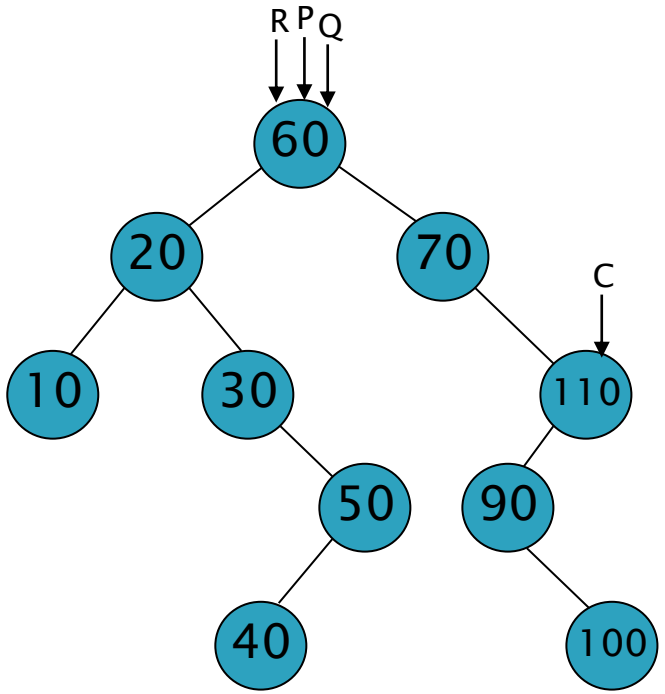


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

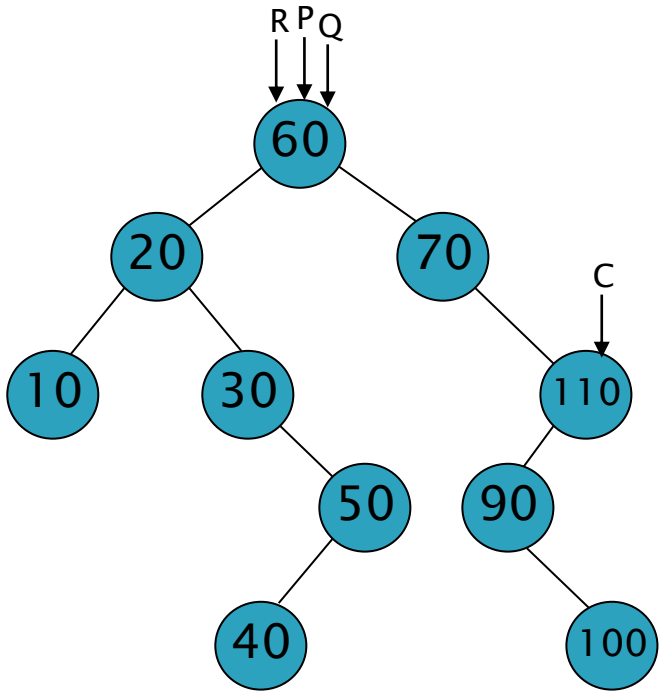


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

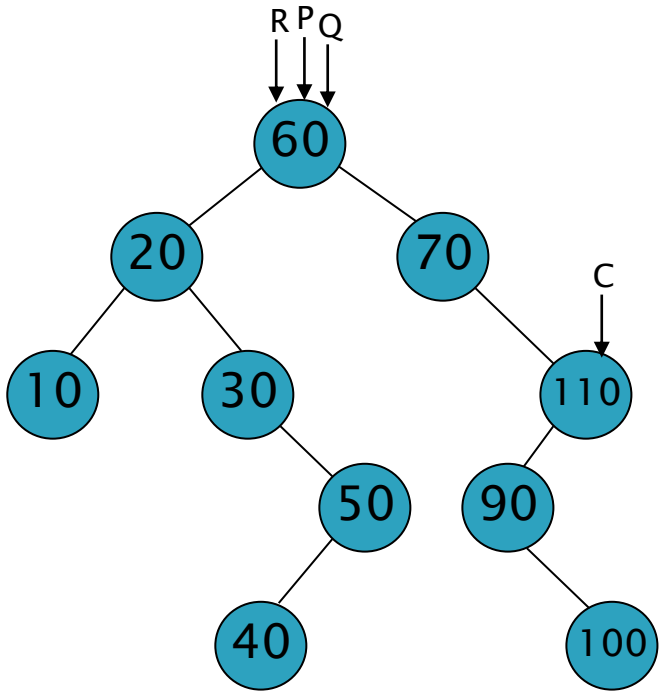


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

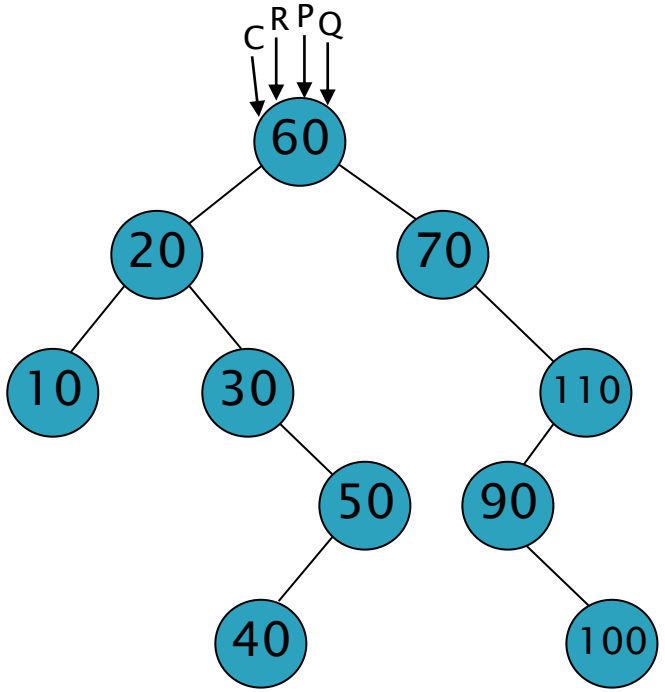


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

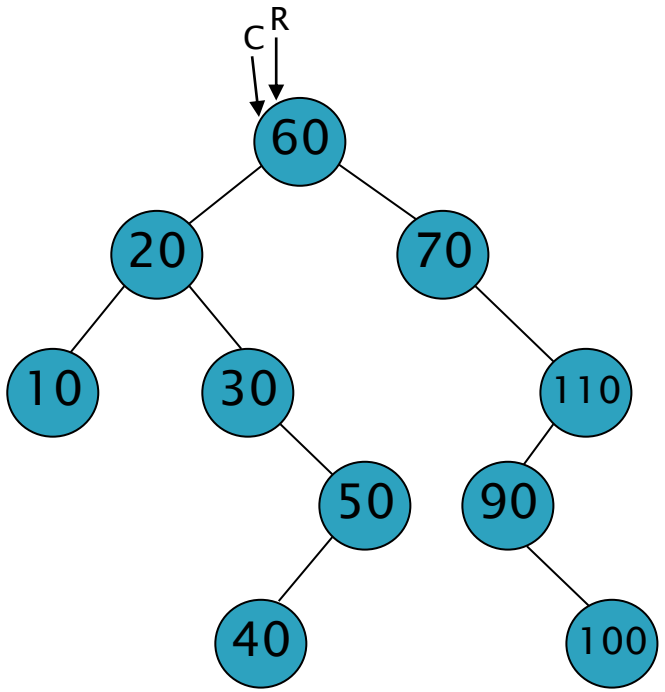


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #1

tkey = 60

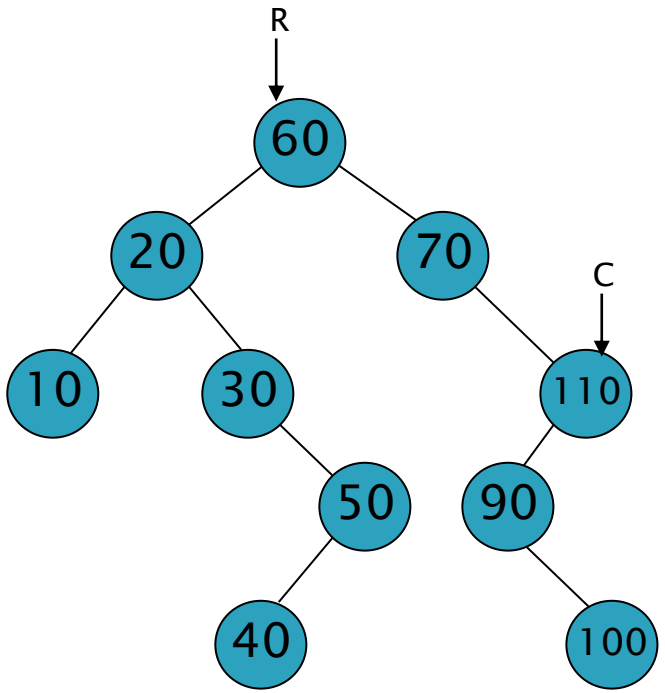


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

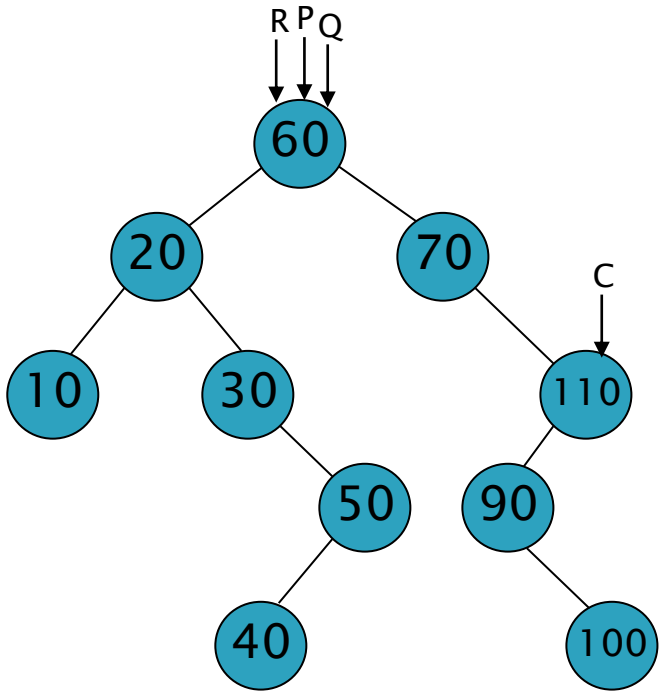


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

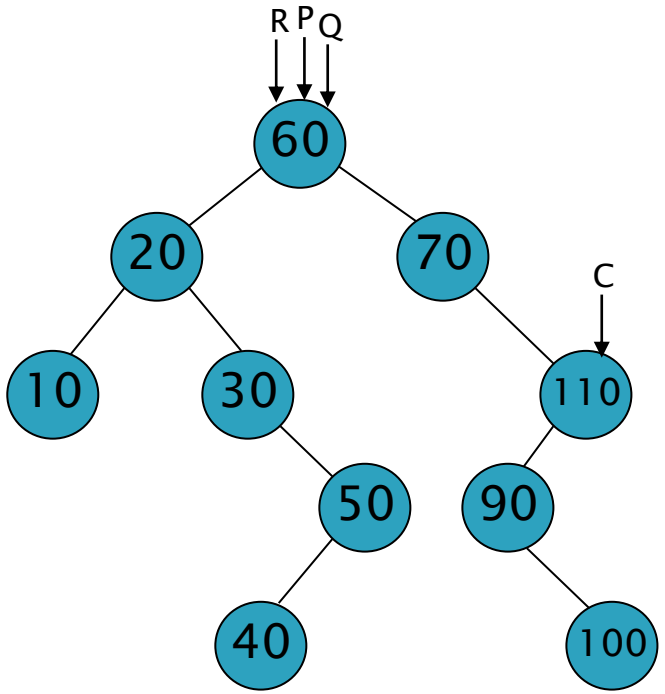


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

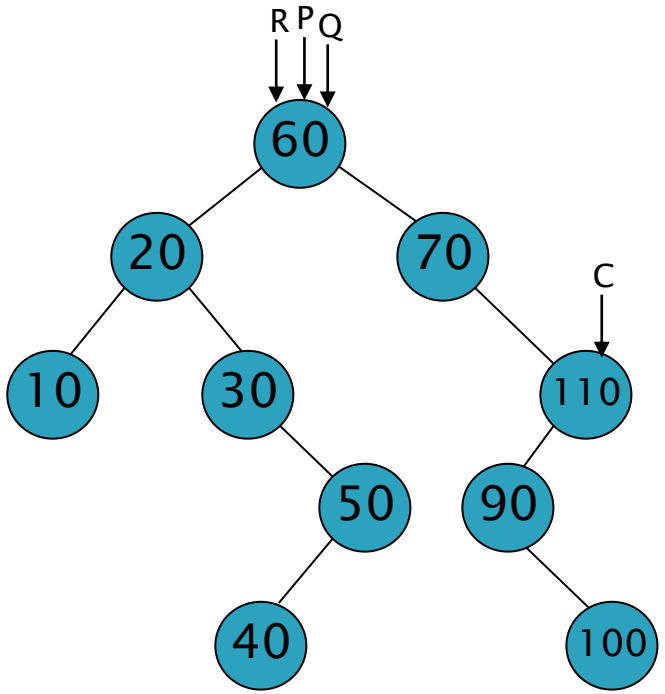


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

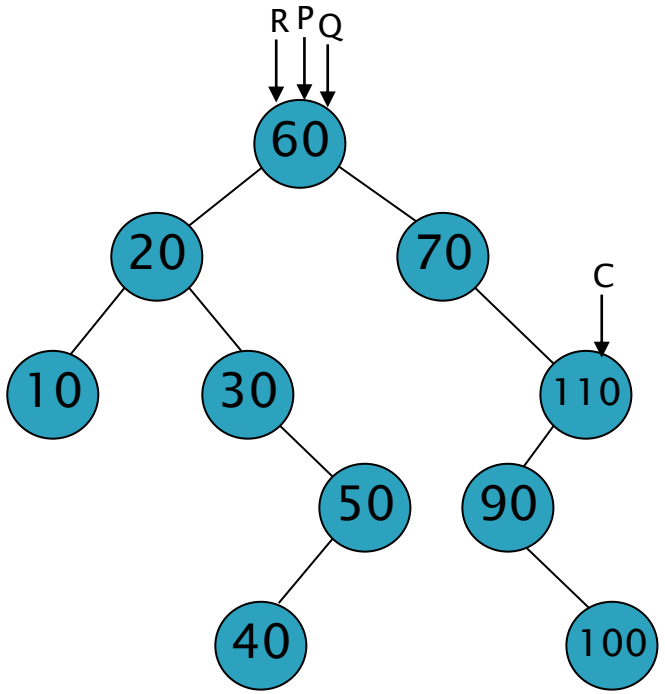


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

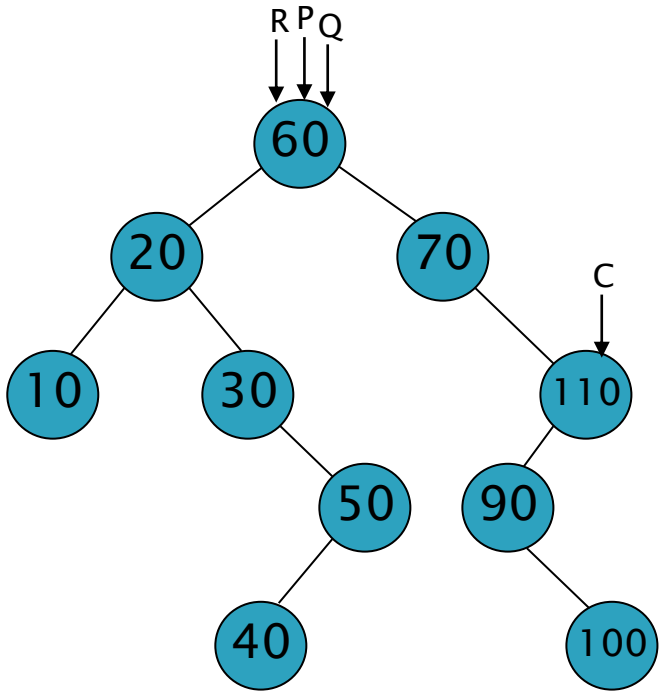


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

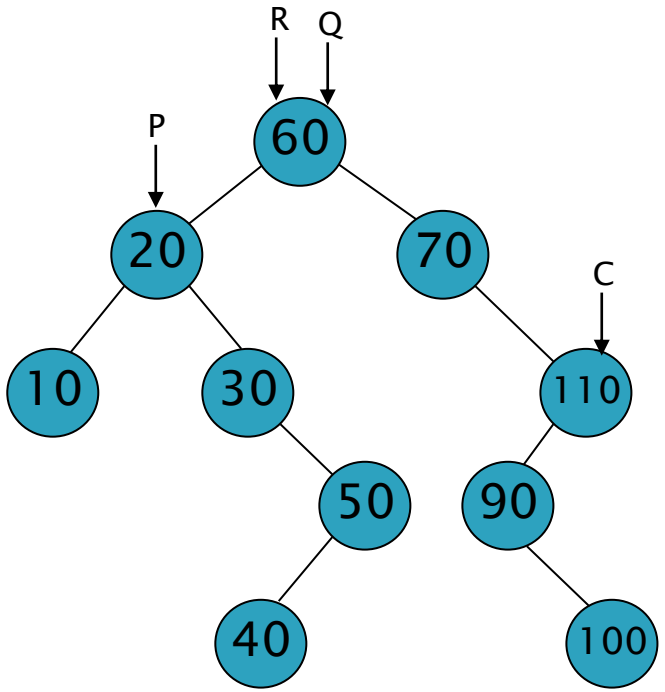


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

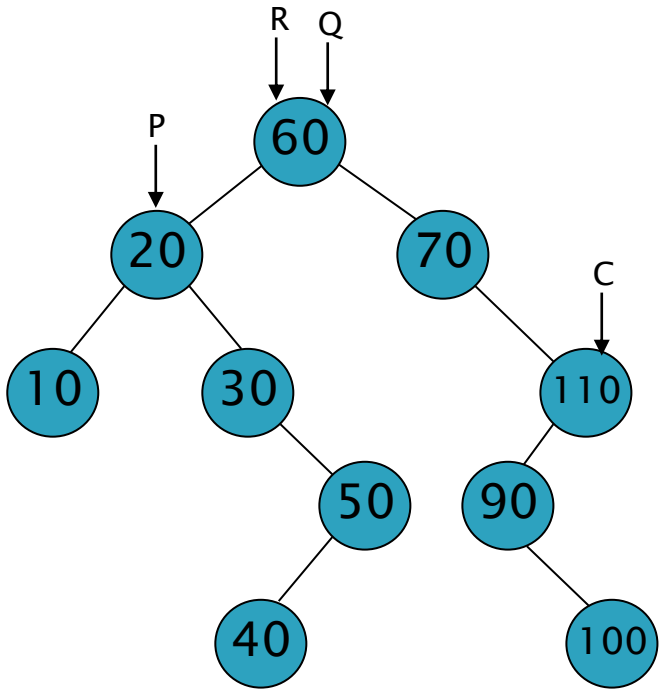


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

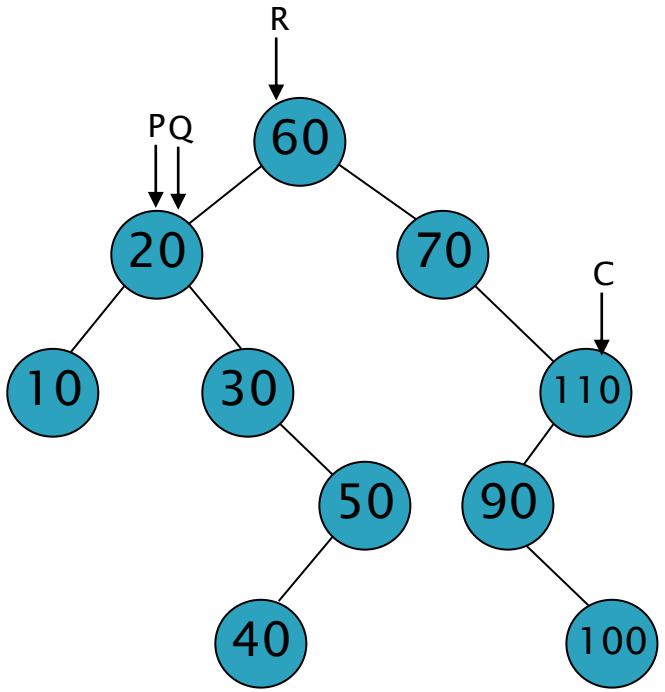


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

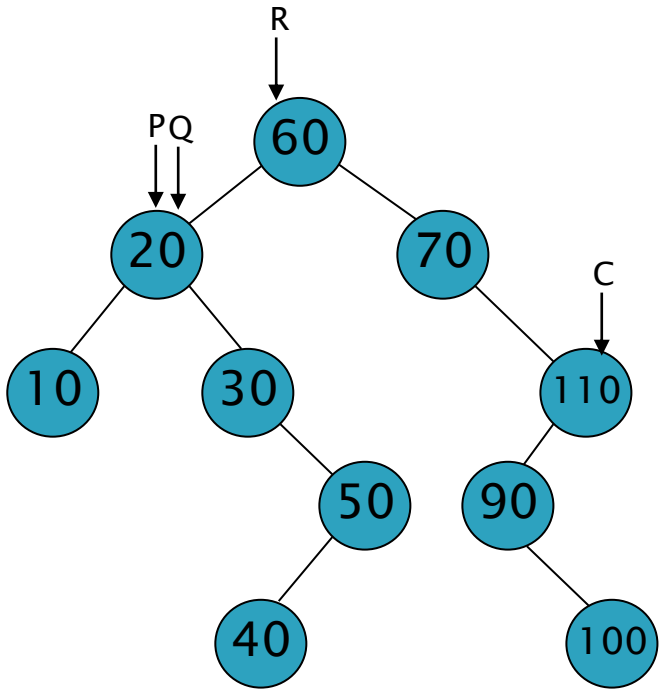


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

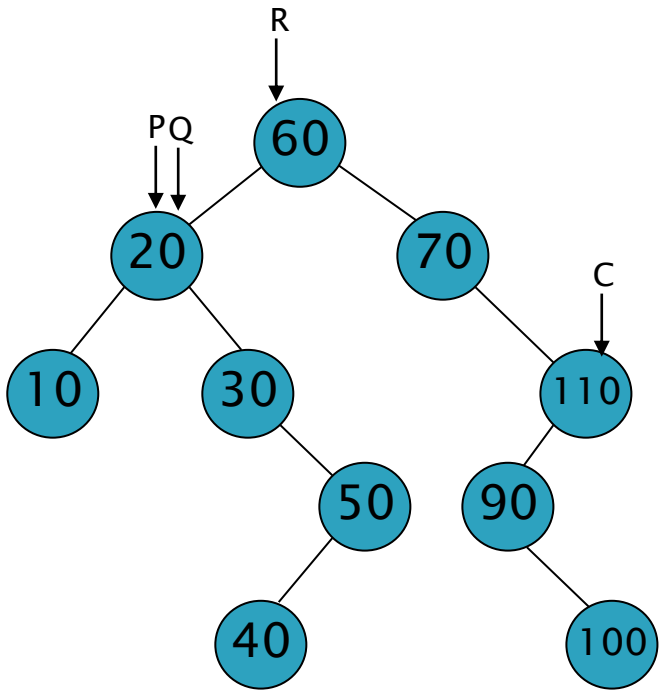


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

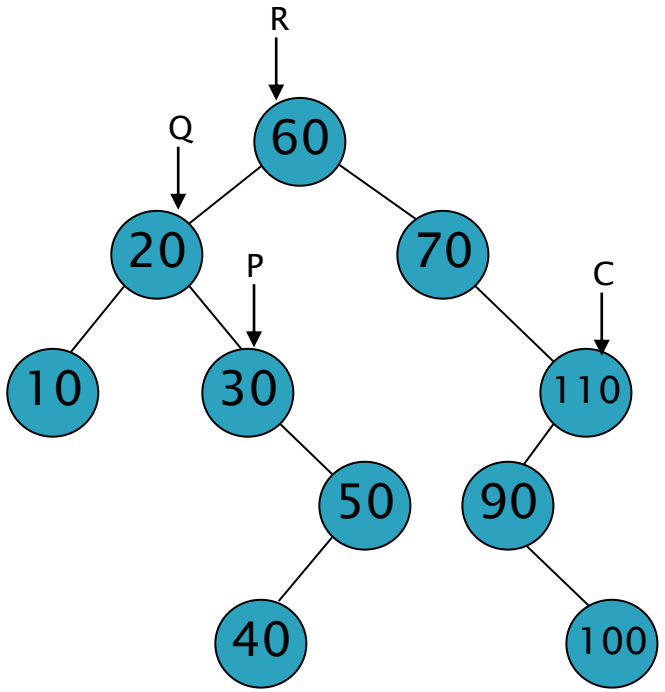


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

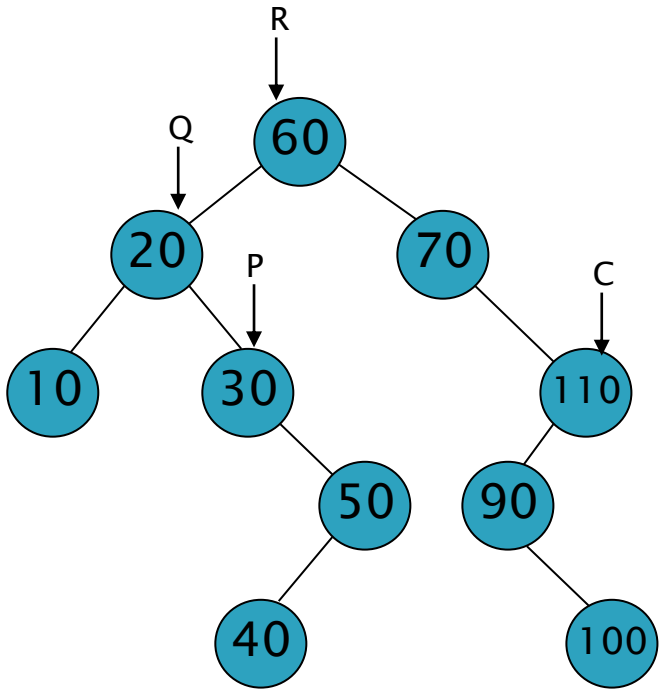


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

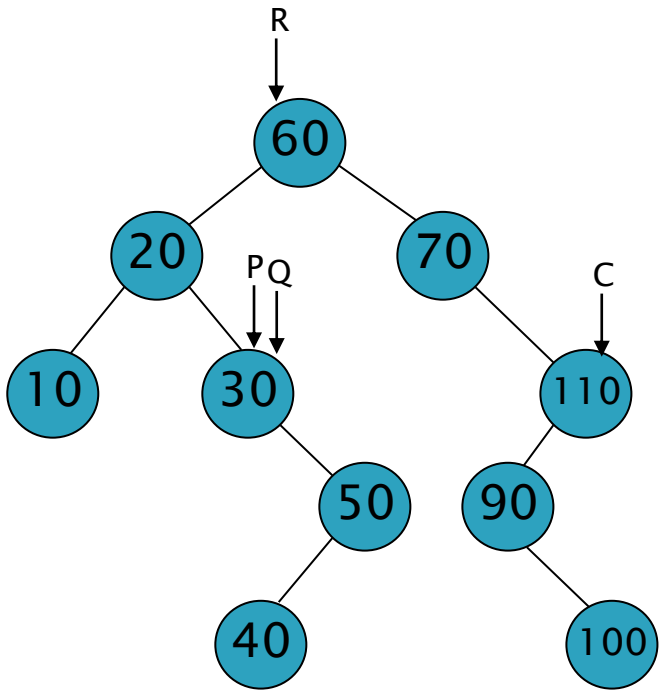


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

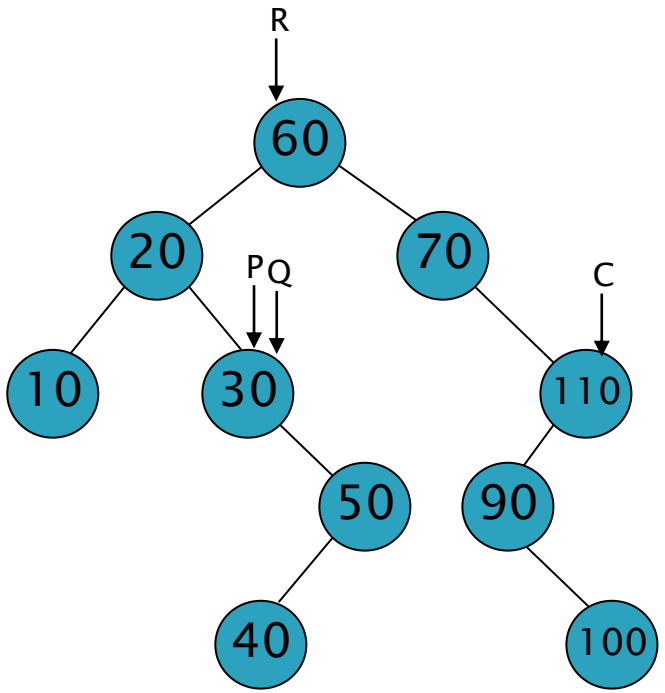


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

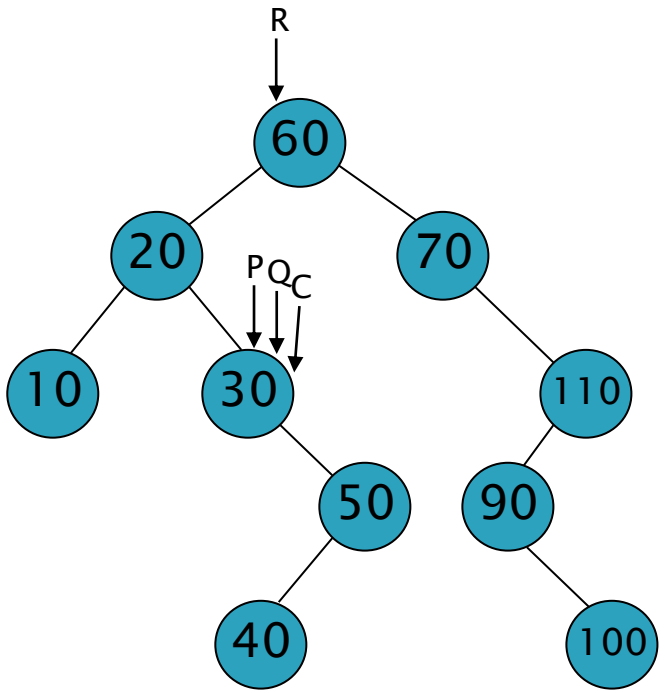


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

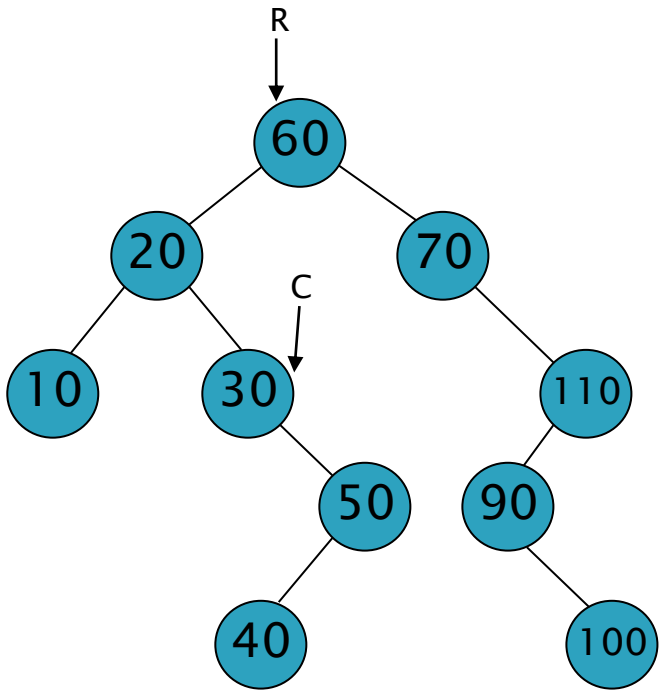


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #2

tkey = 30

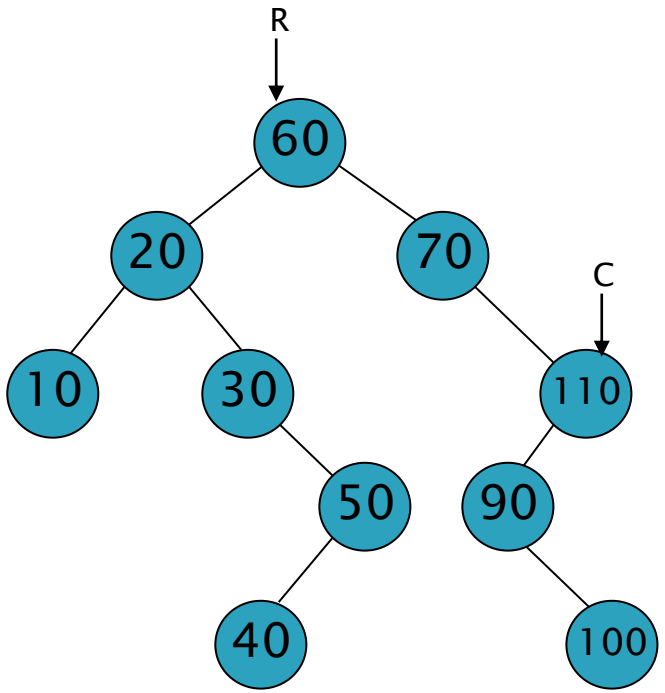


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

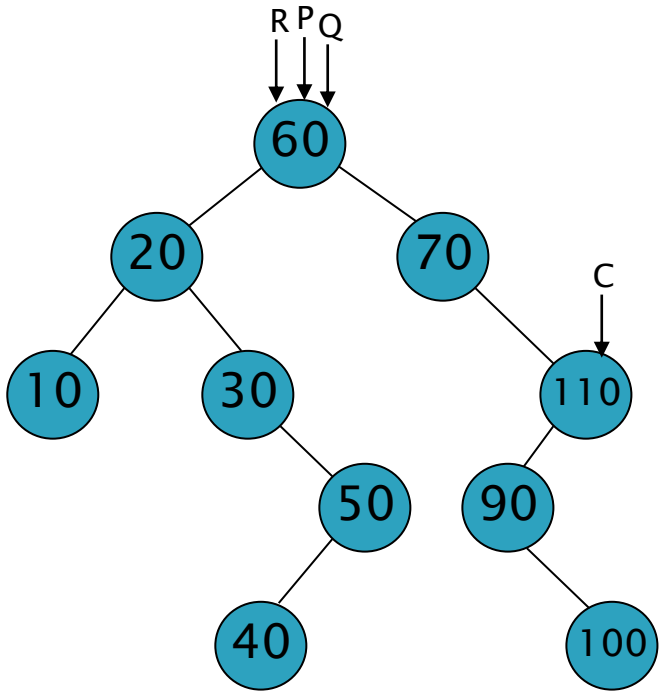


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

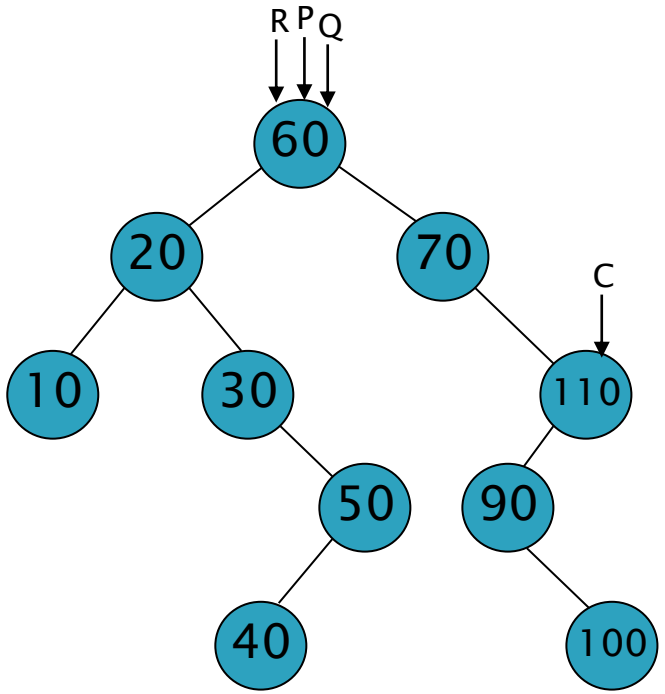


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

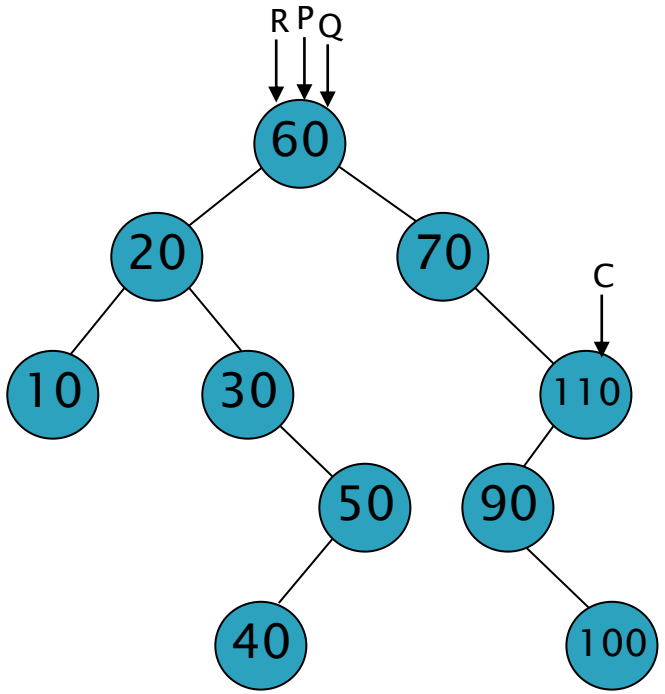


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

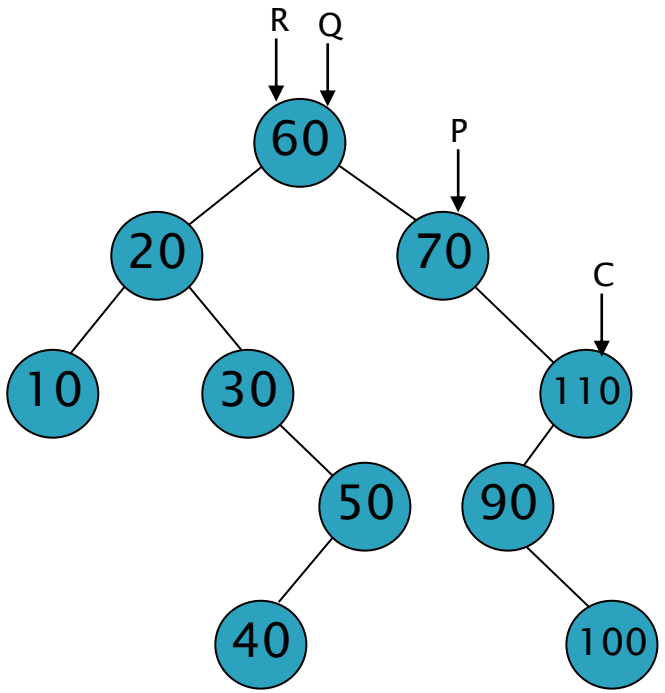


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

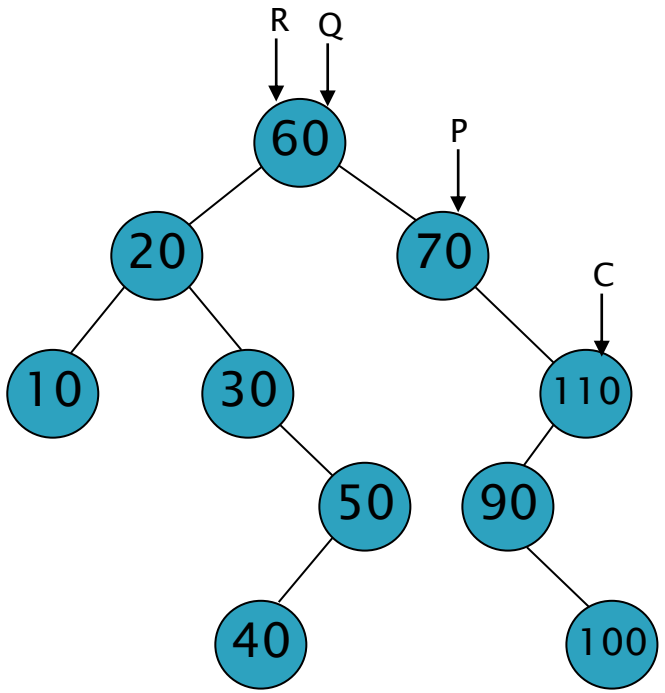


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

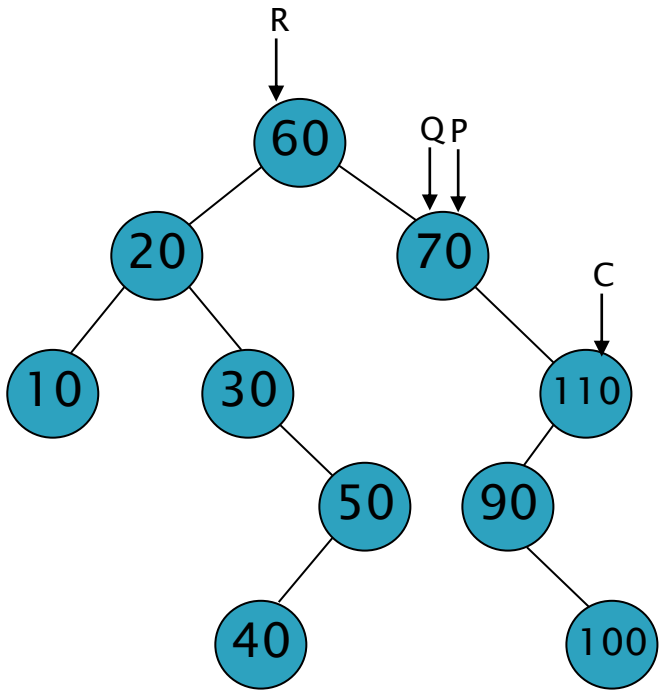


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

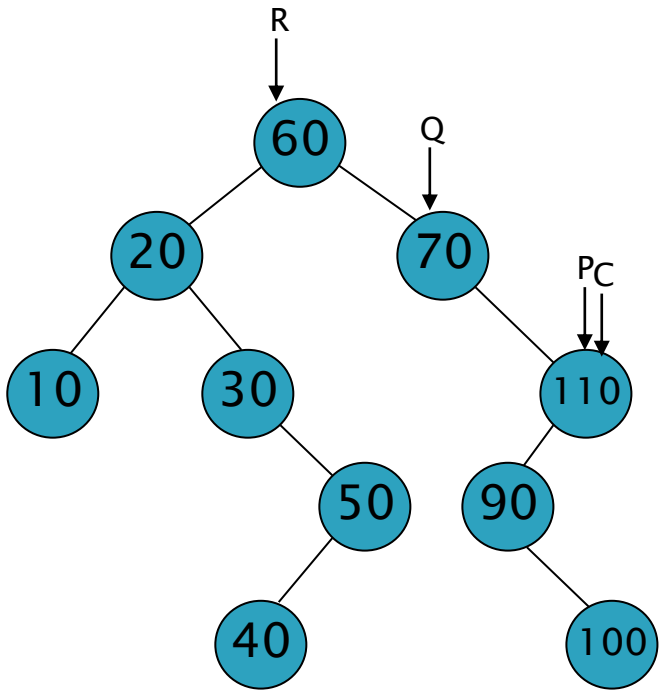


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

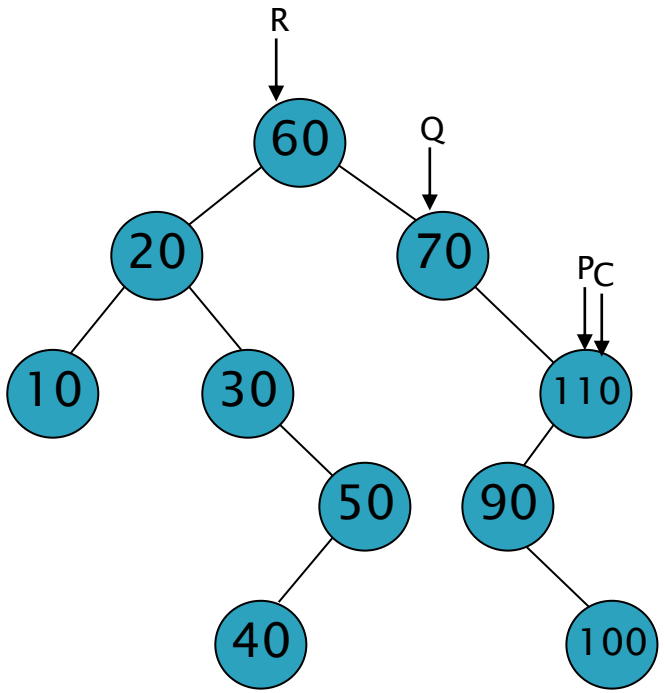


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

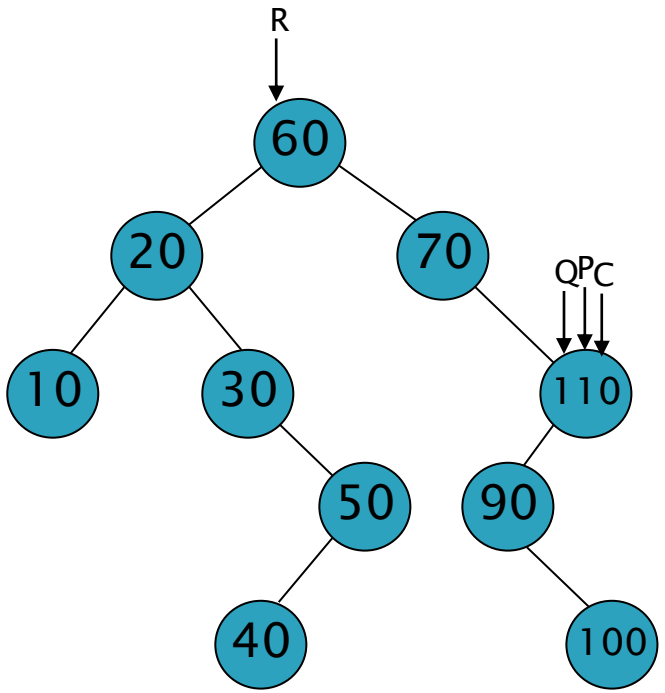


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

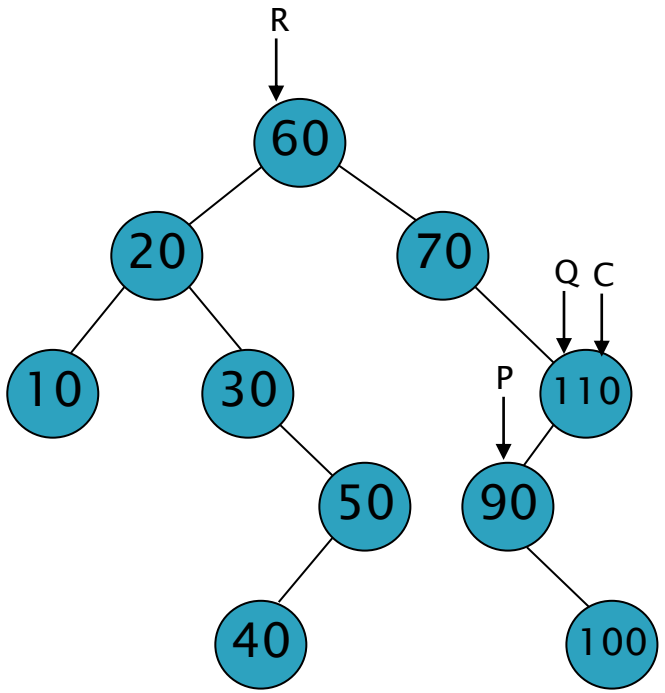


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

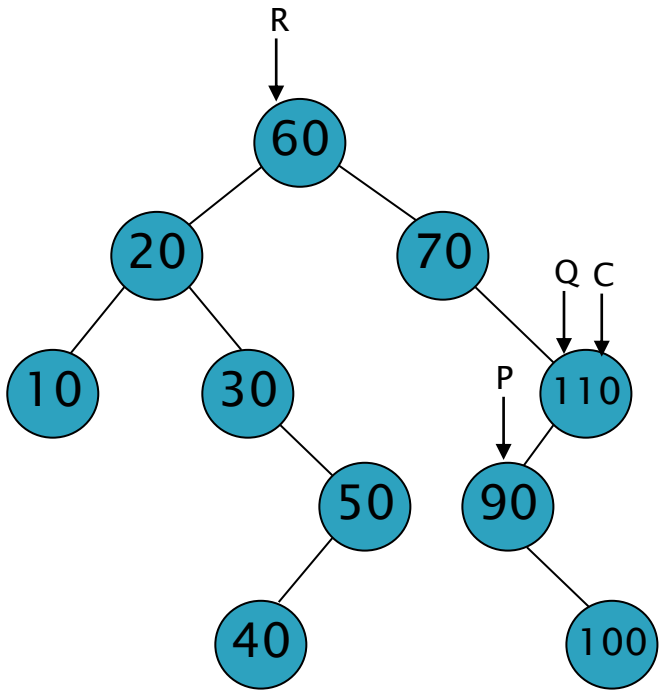


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

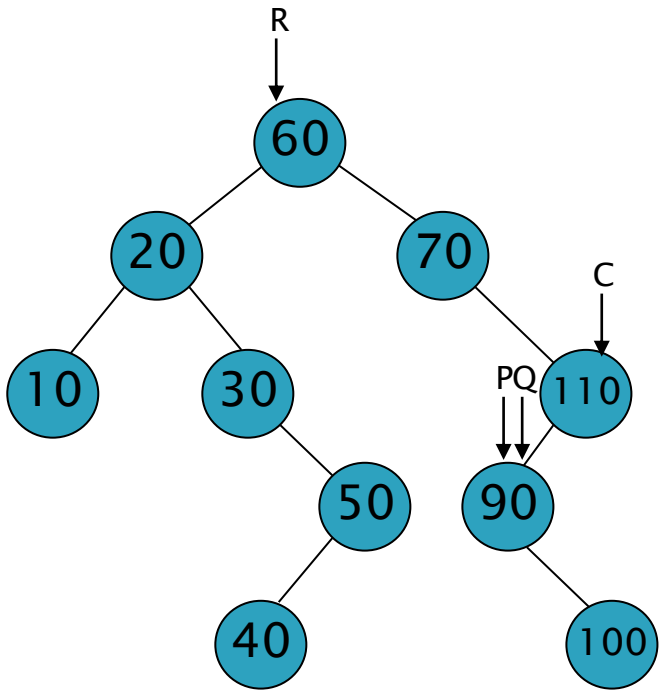


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

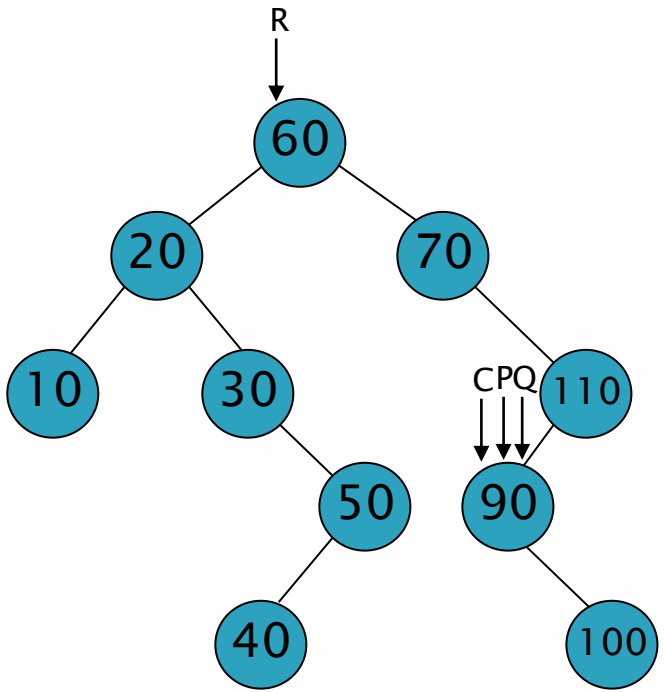


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #3

tkey = 90

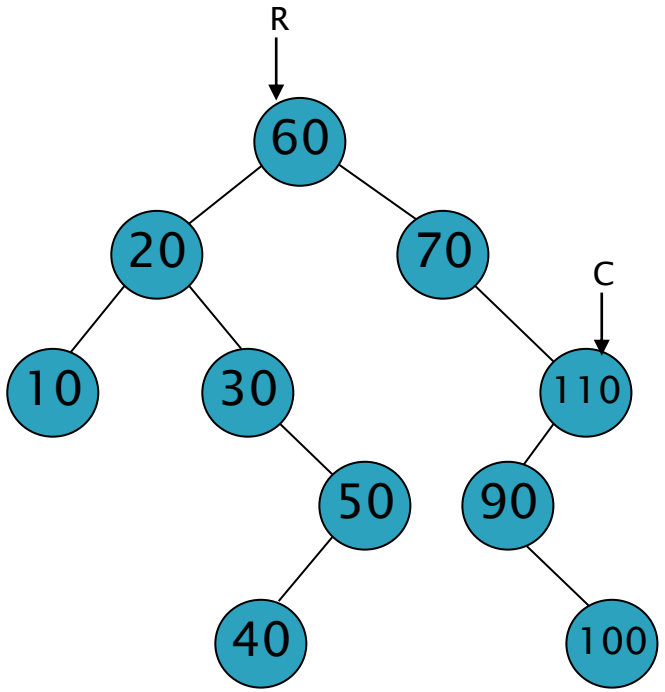


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

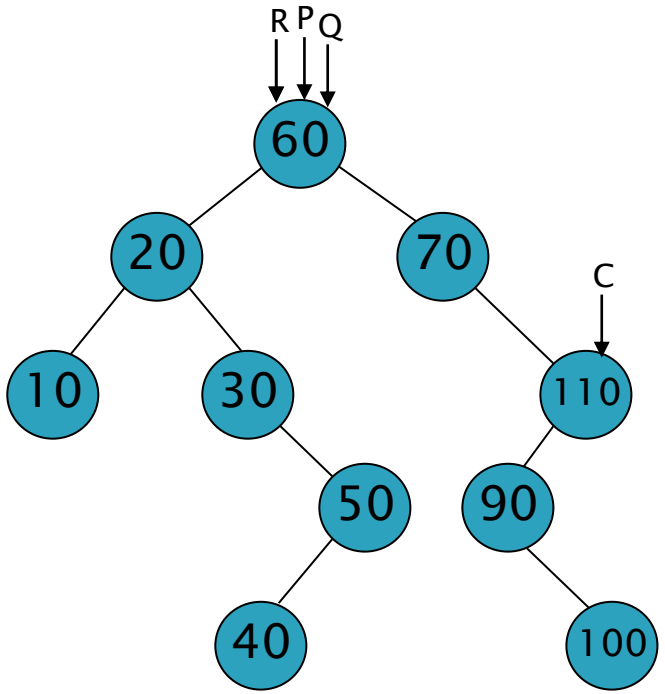


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

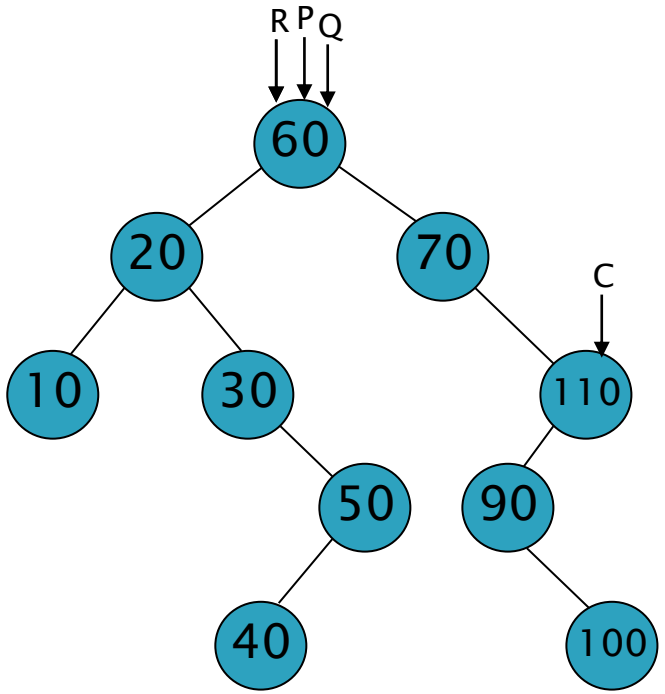


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

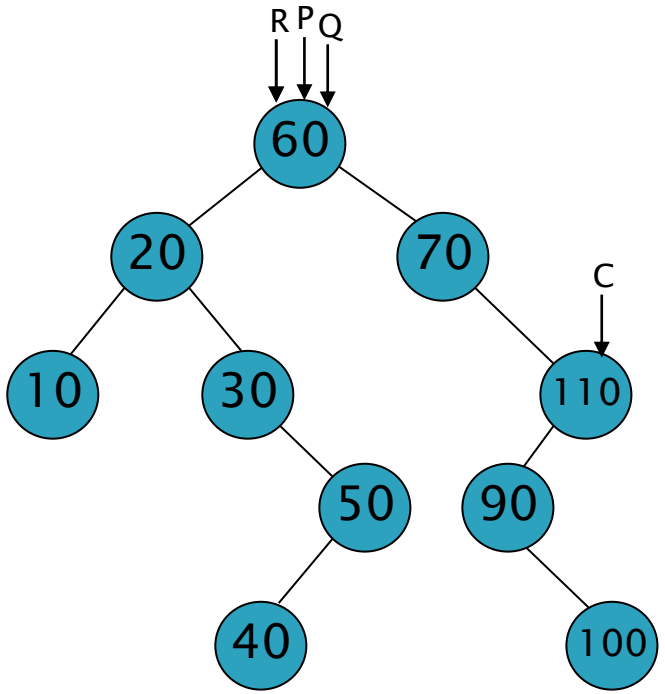


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

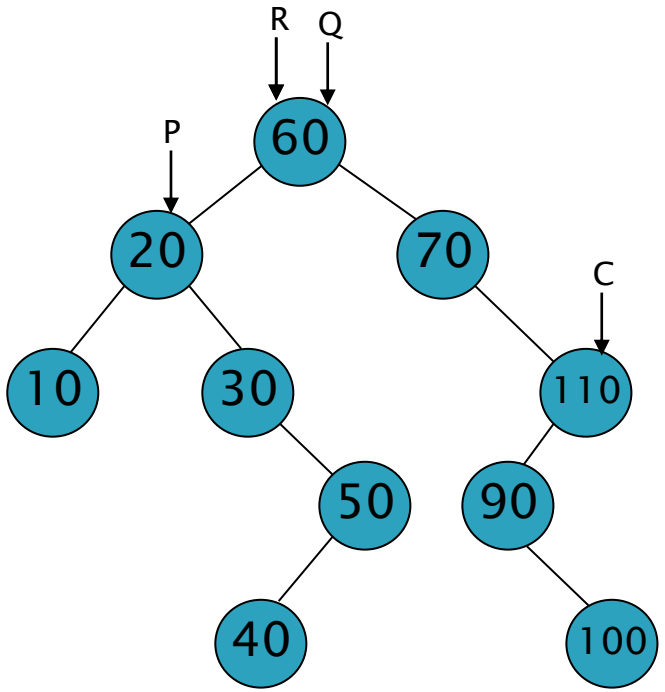


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

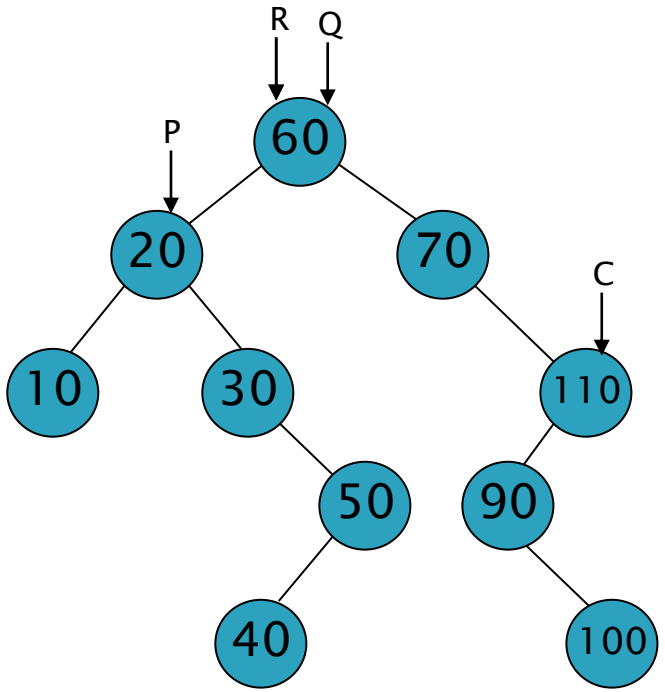


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

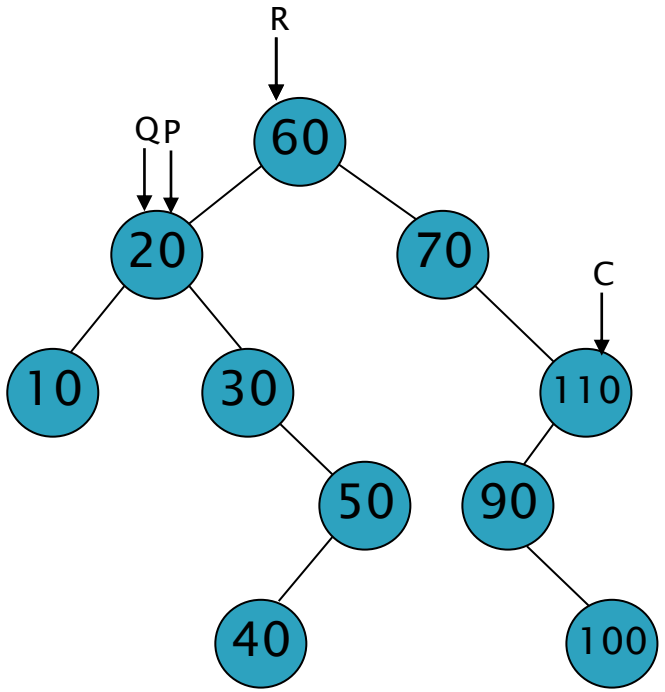


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

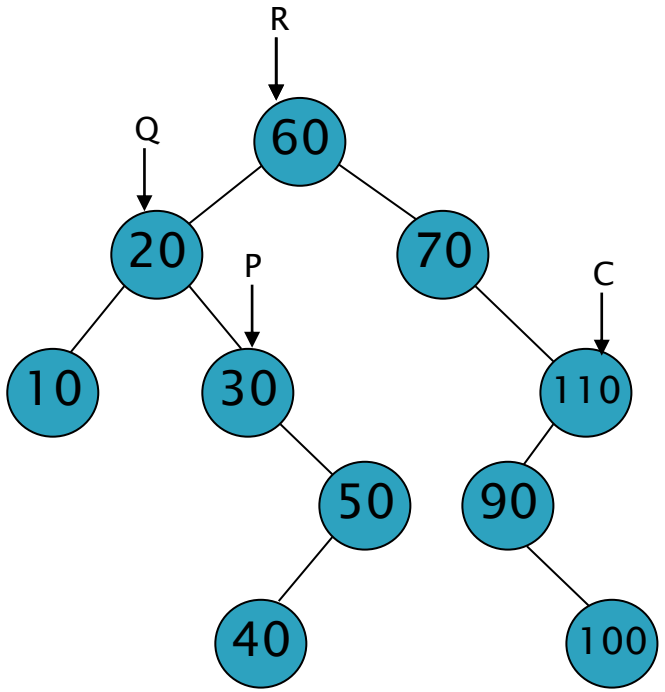


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

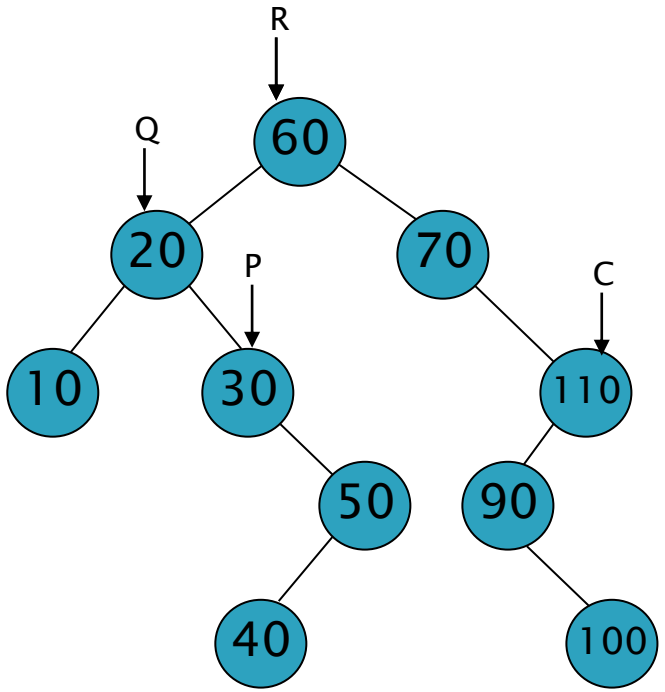


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

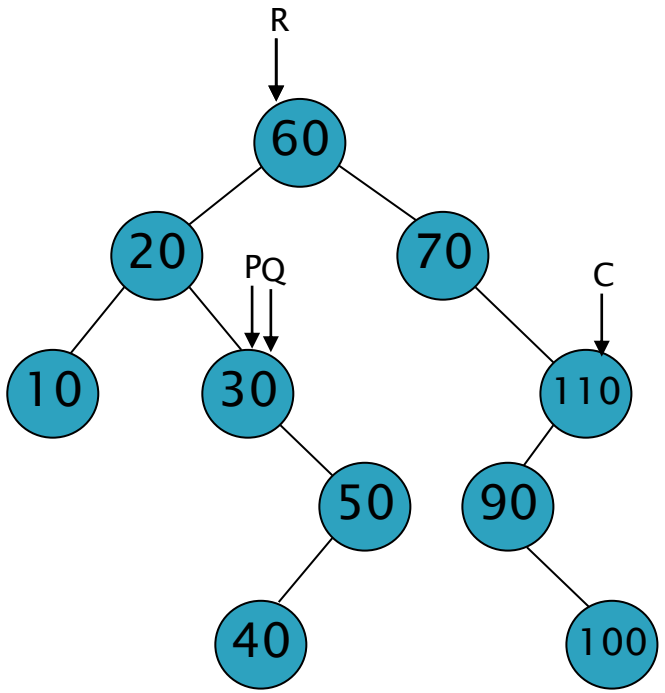


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

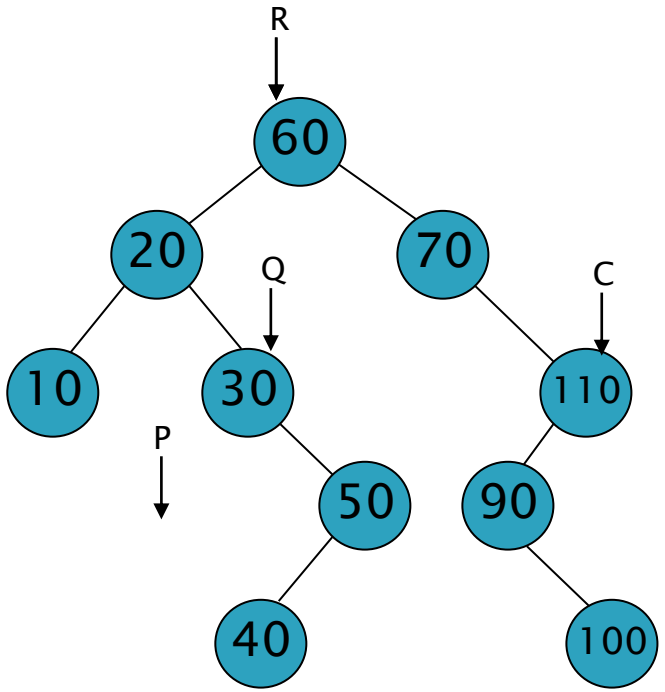


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

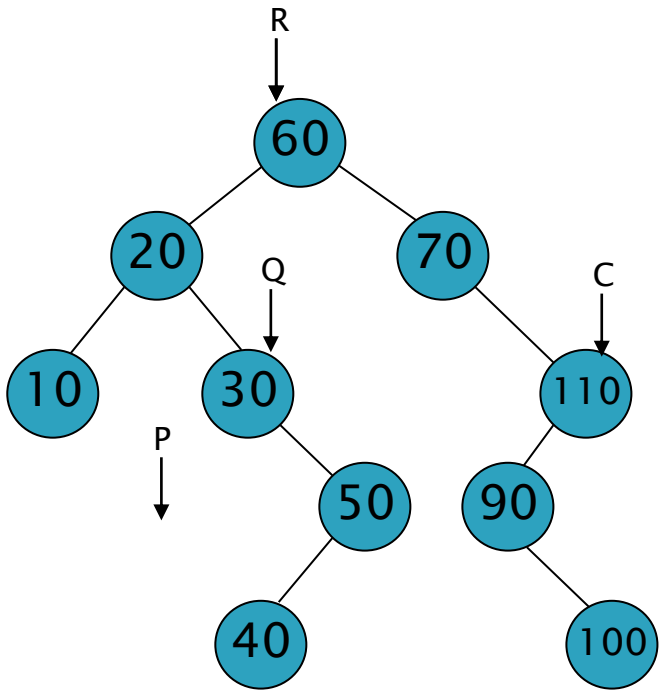


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

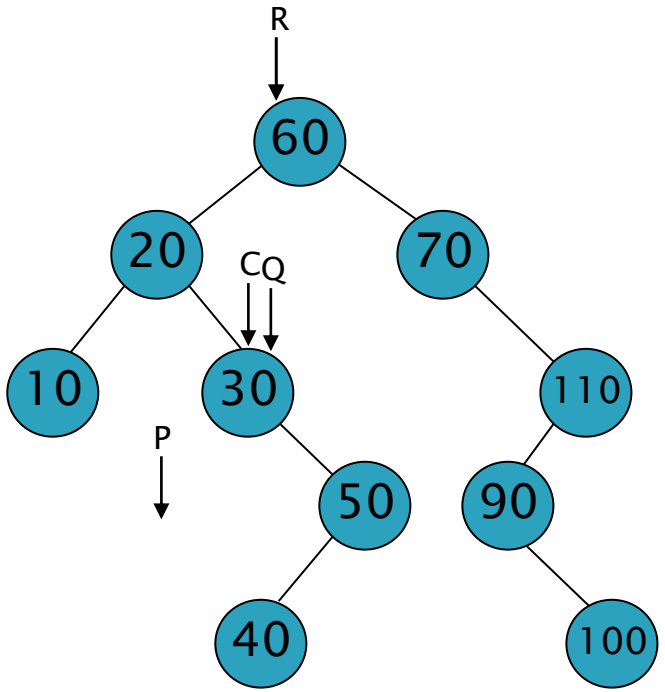


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25

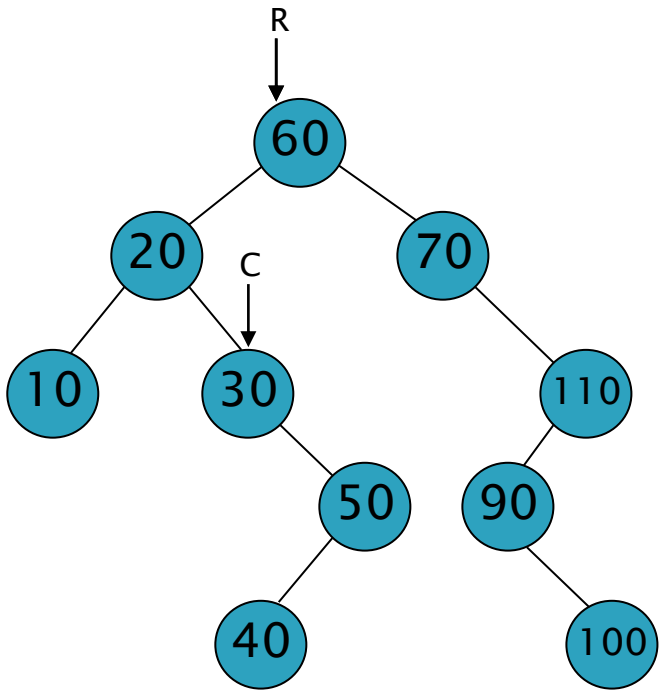


ADT Binary Search Tree: Implementation

```
public boolean findkey(int tkey) {
      BSTNode<T> p = root,q = root;

      if(empty())
                return false;

      while(p != null) {
                q = p;
                if(p.key == tkey) {
                          current = p;
                          return true;
                }
                else if(tkey < p.key)
                          p = p.left;
                else
                          p = p.right;
      }

      current = q;
      return false;
}
```

Example #4

tkey = 25



ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

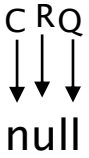
k = 60

ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

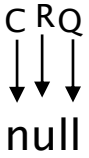


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

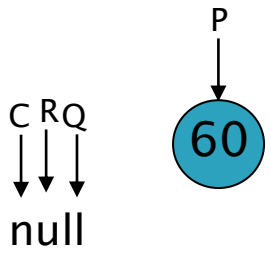


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

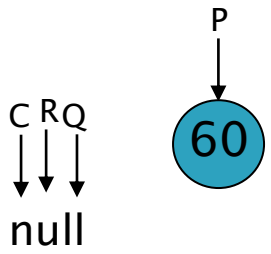


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

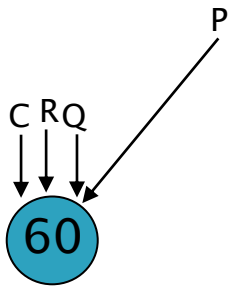


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

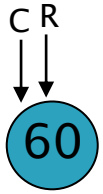


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #1

k = 60

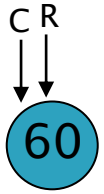


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

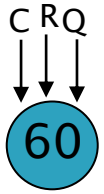


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

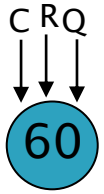


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

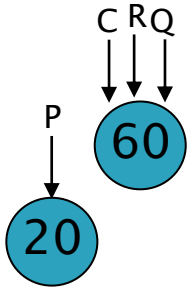


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

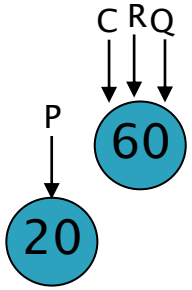


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

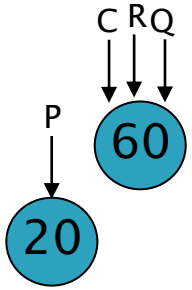


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

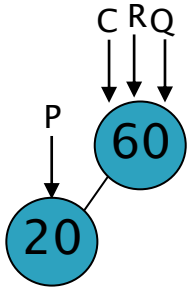


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

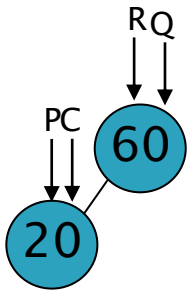


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

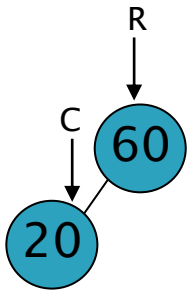


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #2

k = 20

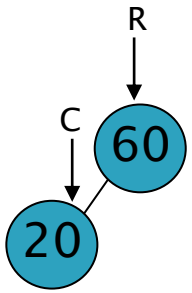


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

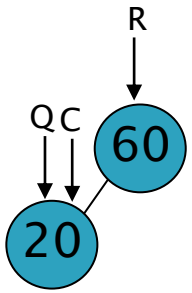


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

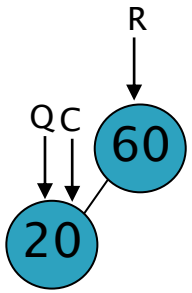


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

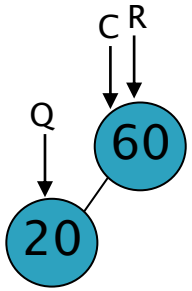


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

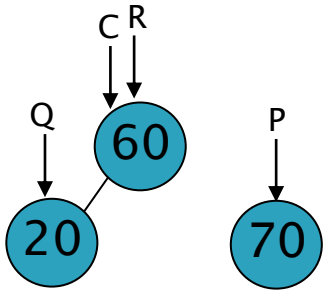


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

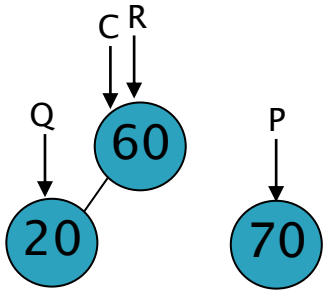


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

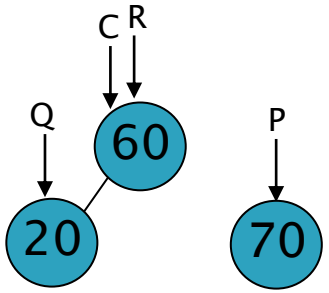


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

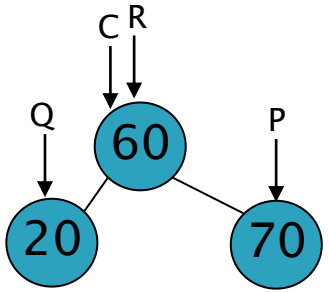


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

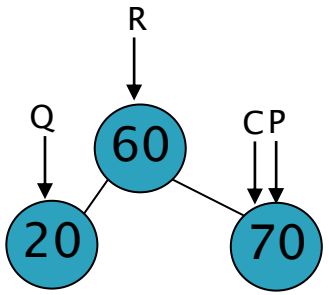


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

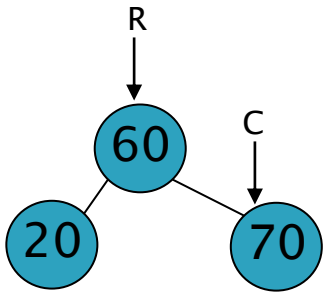


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #3

k = 70

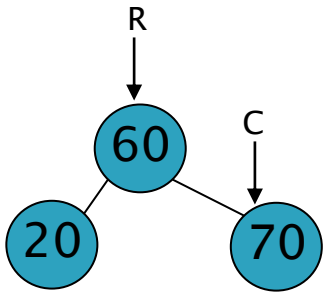


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

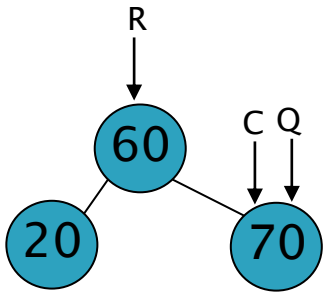


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

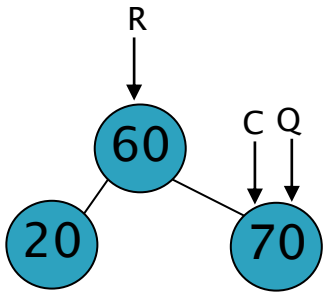


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

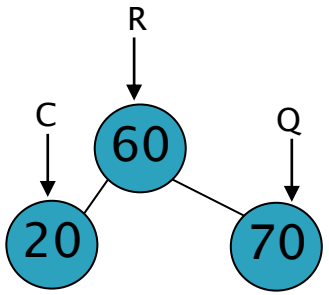


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

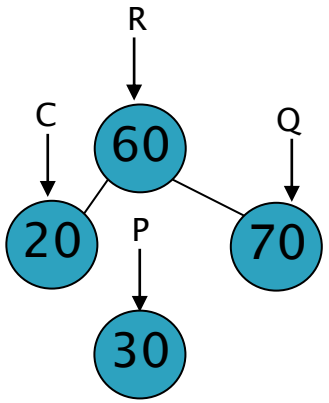


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

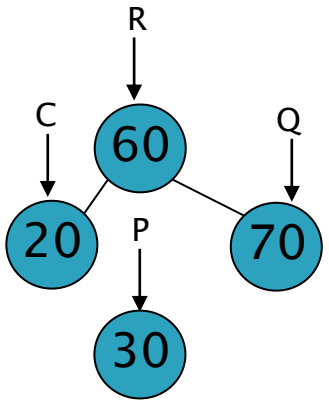


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

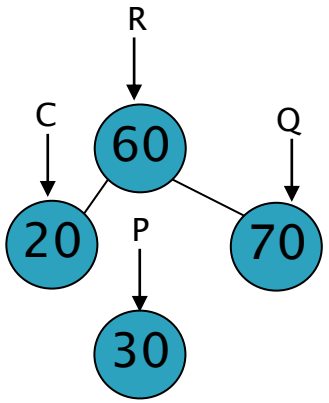


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

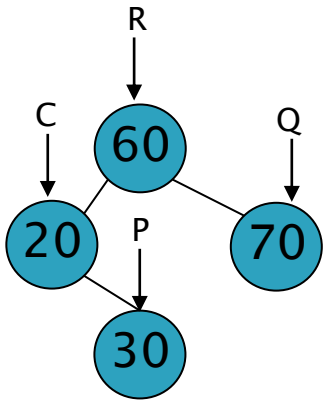


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

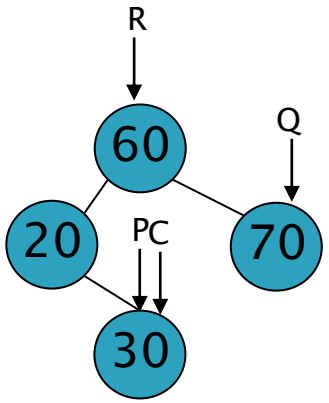


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

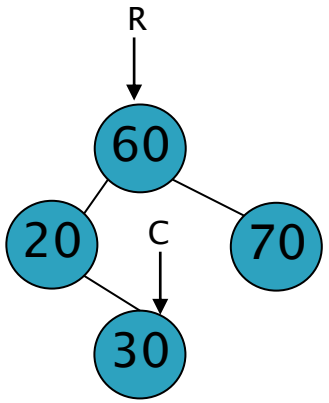


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 30

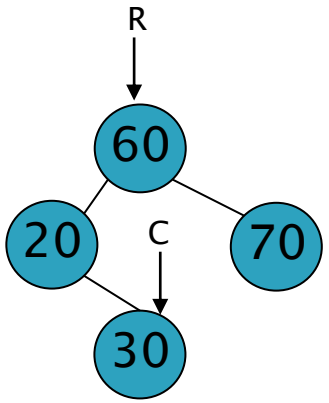


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

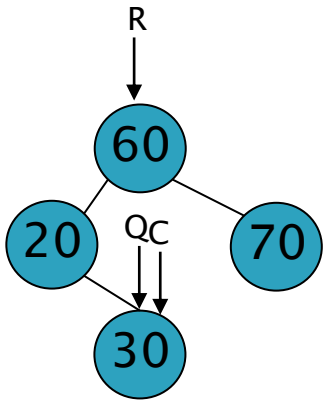


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

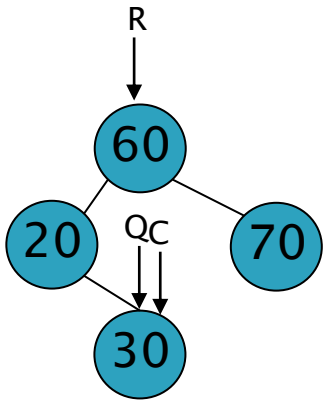


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

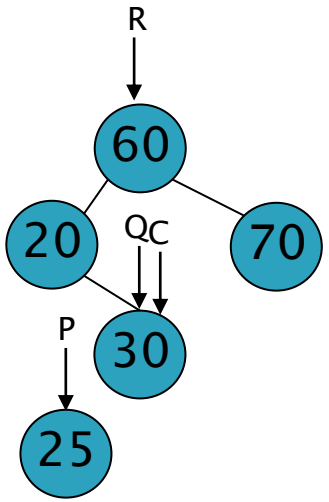


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

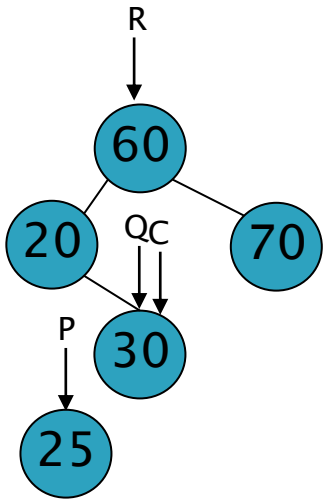


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

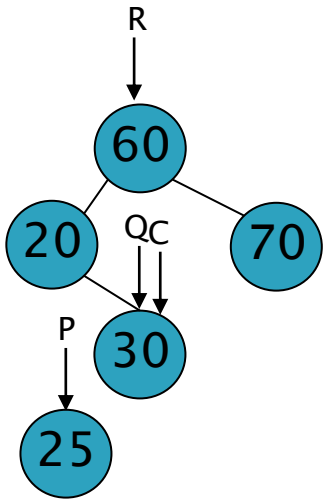


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

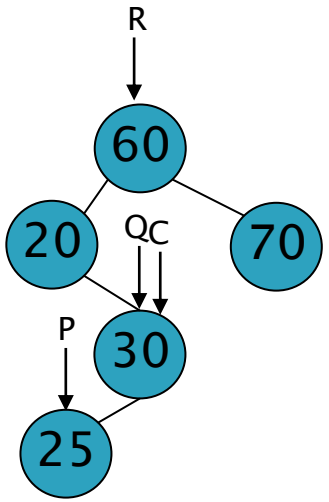


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

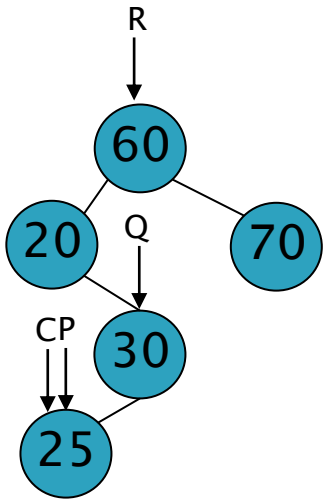


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

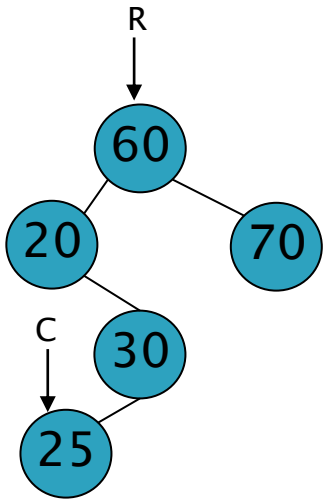


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #4

k = 25

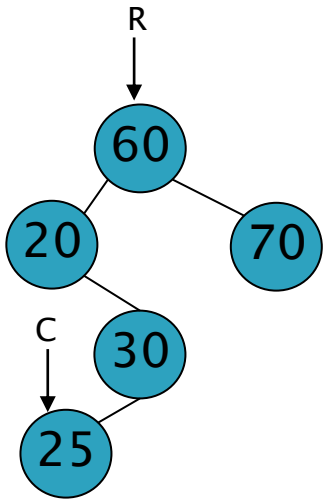


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

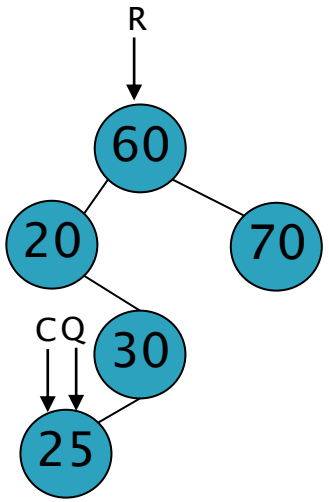


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

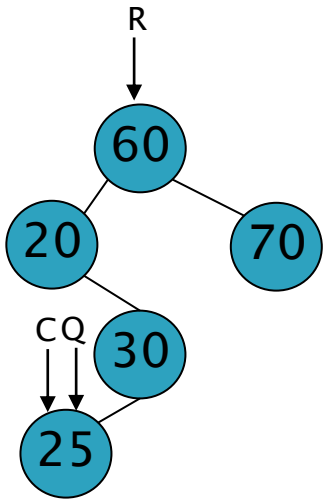


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

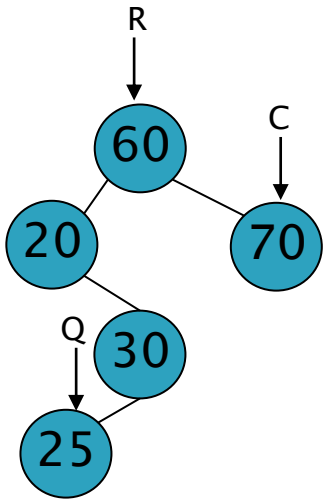


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

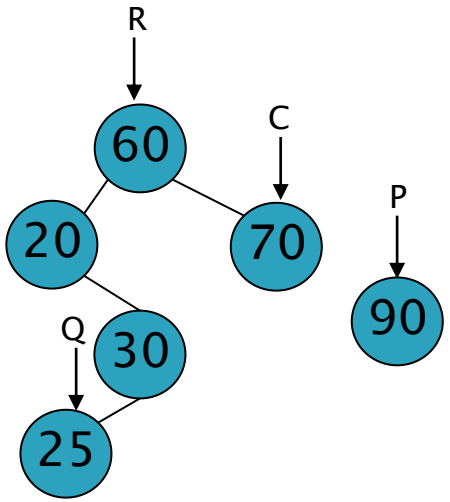


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

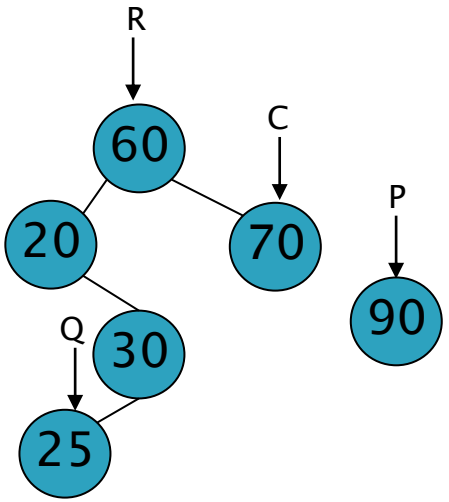


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

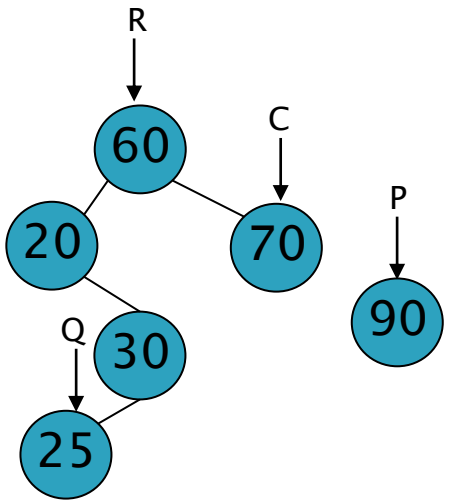


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

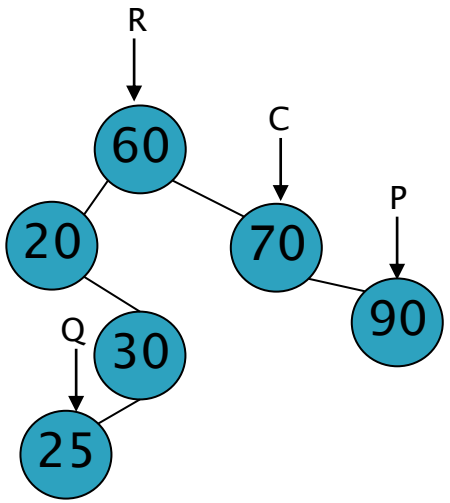


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

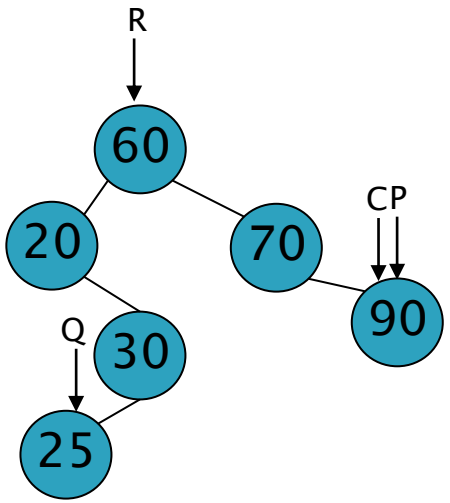


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

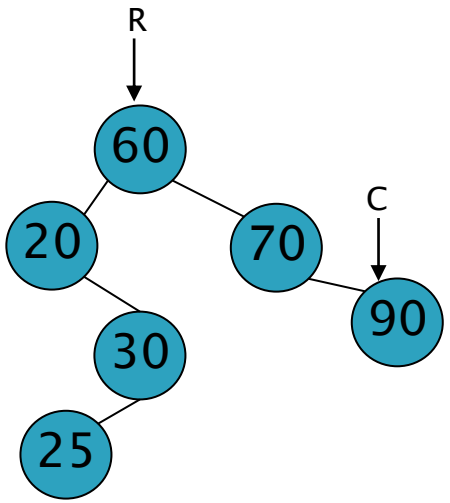


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #5

k = 90

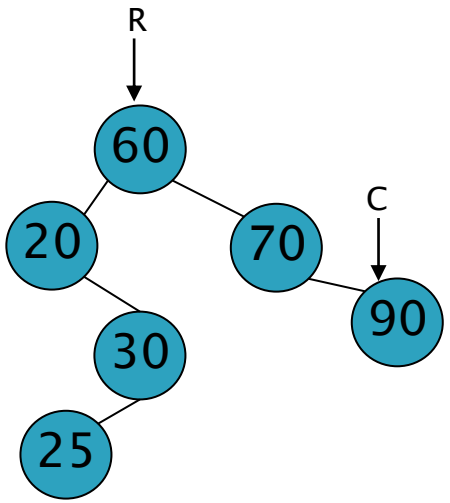


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

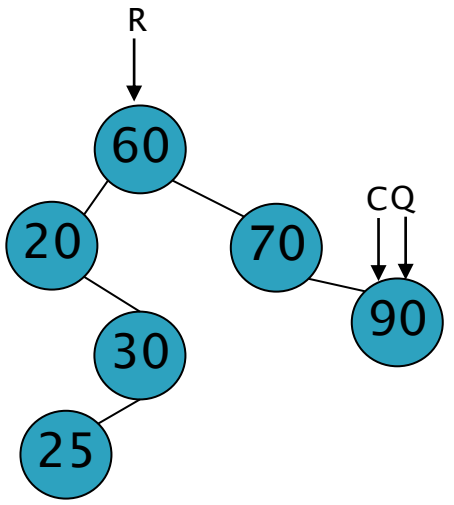


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

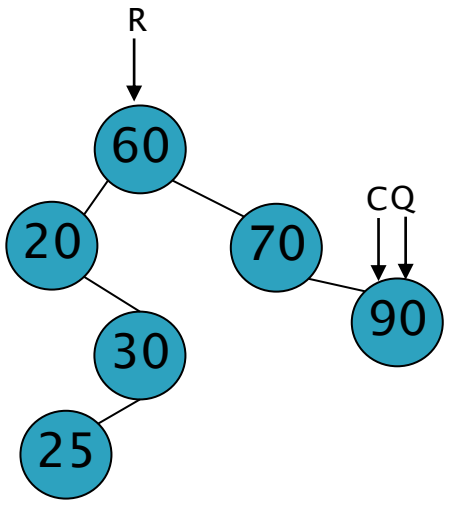


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

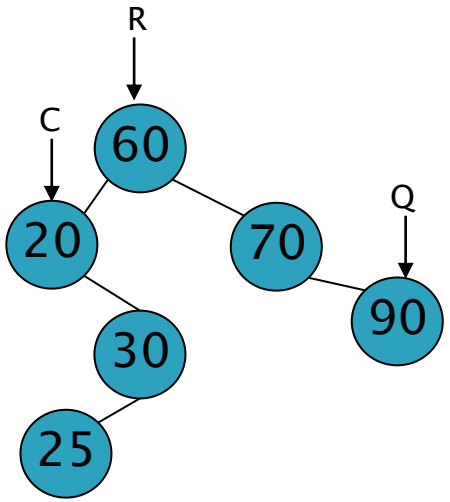


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

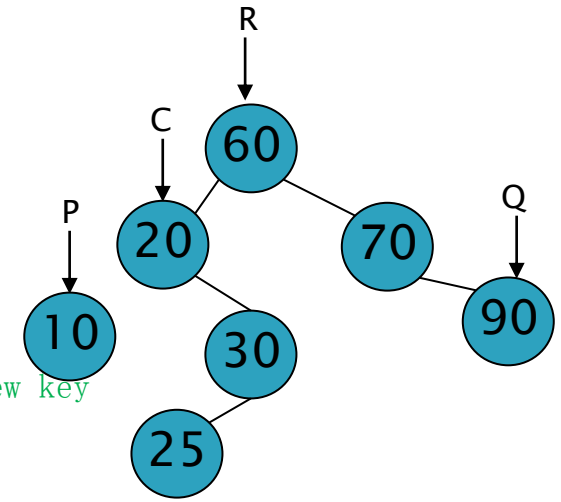


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

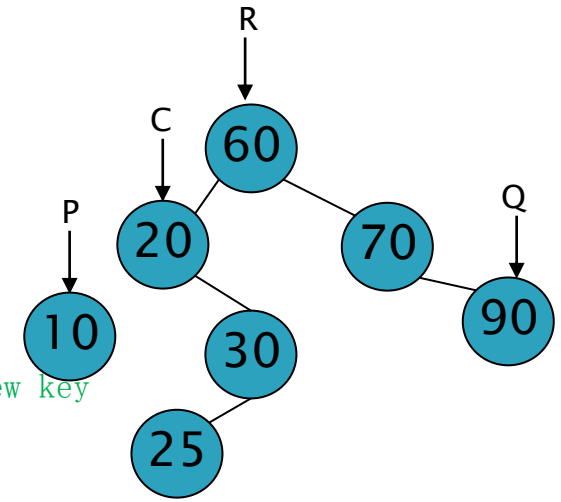


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

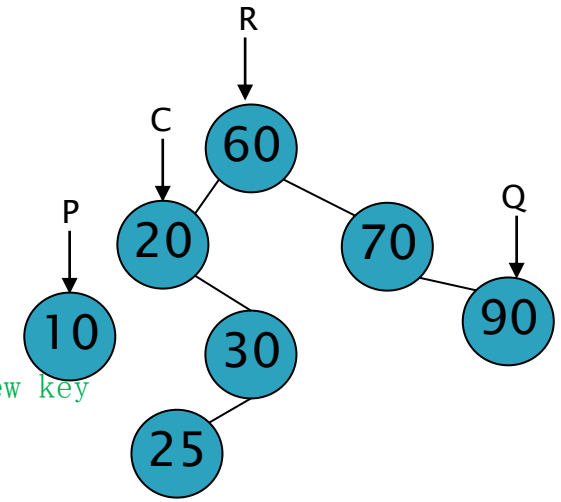


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

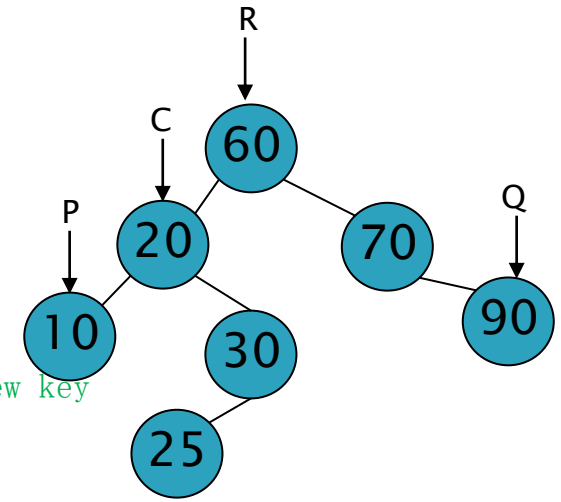


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

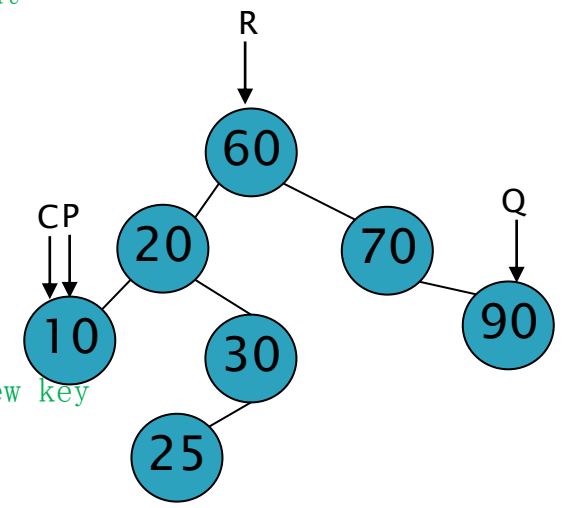


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

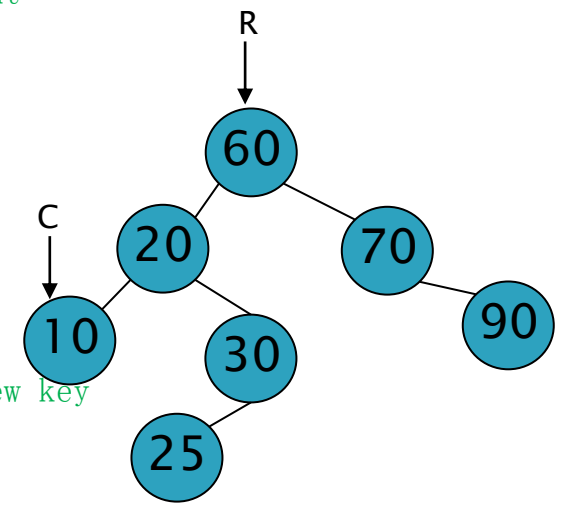


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #6

k = 10

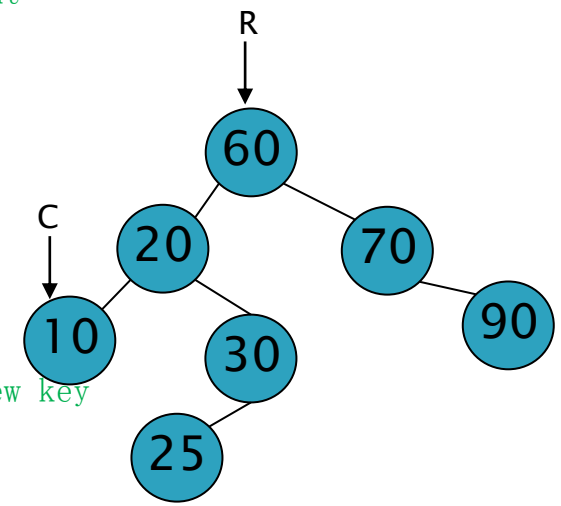


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30

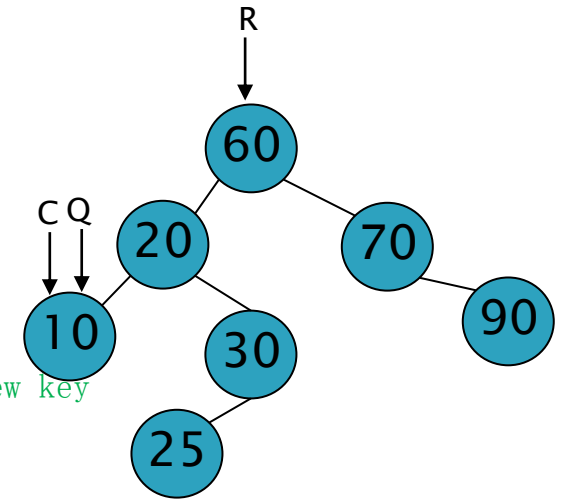


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30

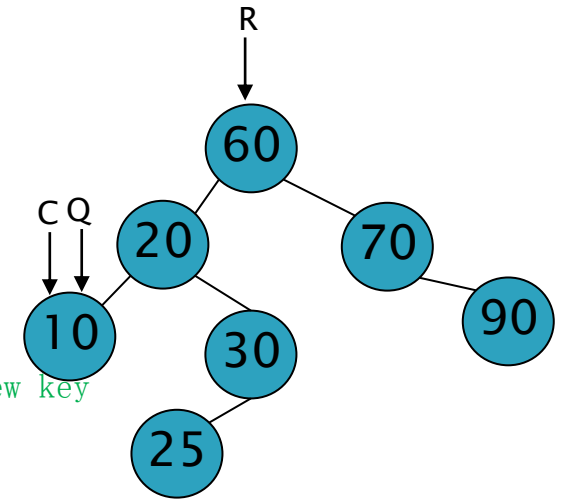


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30

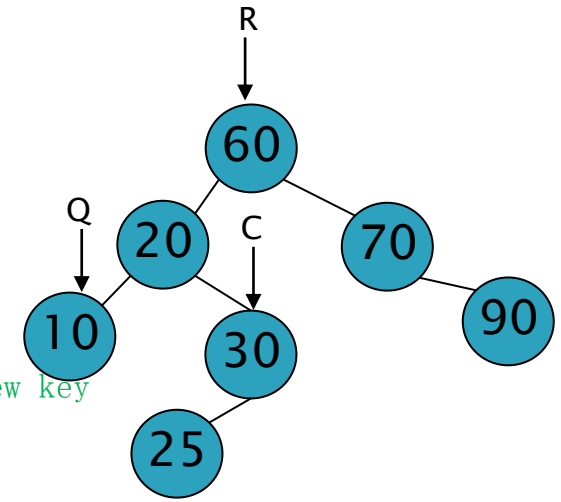


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30

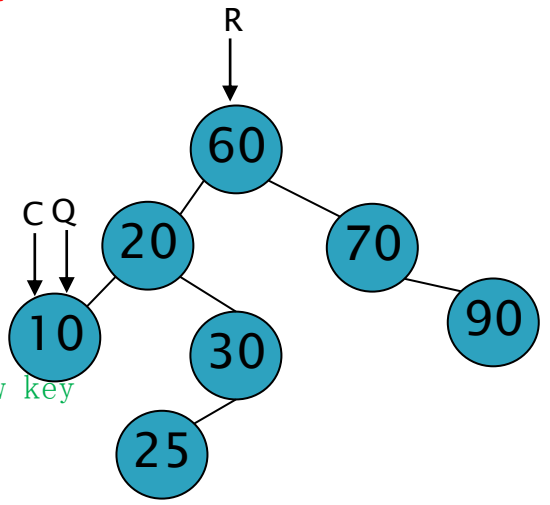


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30

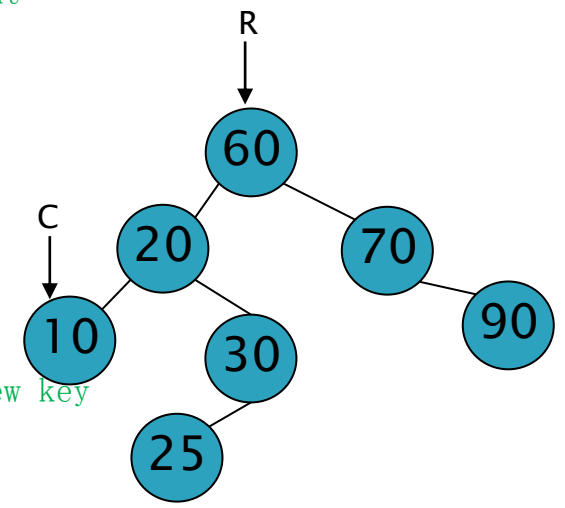


ADT Binary Search Tree: Implementation

```
public boolean insert(int k, T val) {
        BSTNode<T> p, q = current;

        if(findkey(k)) {
                current = q;  // findkey() modified current
                return false; // key already in the BST
        }

        p = new BSTNode<T>(k, val);
        if (empty()) {
                root = current = p;
                return true;
        }
        else {
                // current is pointing to parent of the new key
                if (k < current.key)
                        current.left = p;
                else
                        current.right = p;
                current = p;
                return true;
        }
}
```

Example #7

k = 30



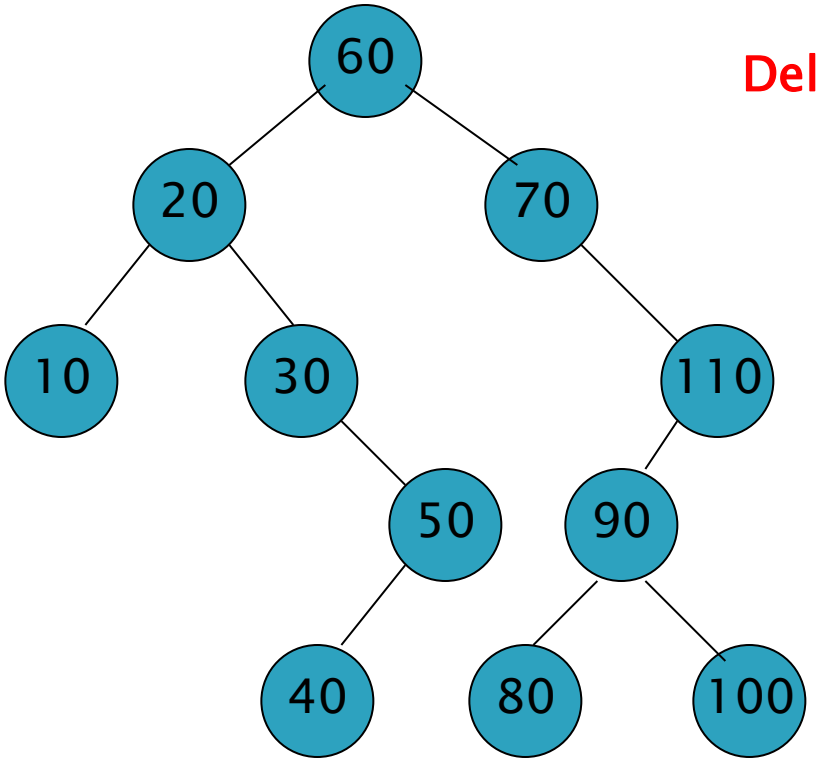
BST Node Deletion

- There are three cases:
  - Case 1: Node to be deleted has no children.
  - Case 2: Node to be deleted has one child.
  - Case 3: Node to be deleted has two children.
- In all these case it is always a leaf node that gets deleted.

BST Deletion: Case 1

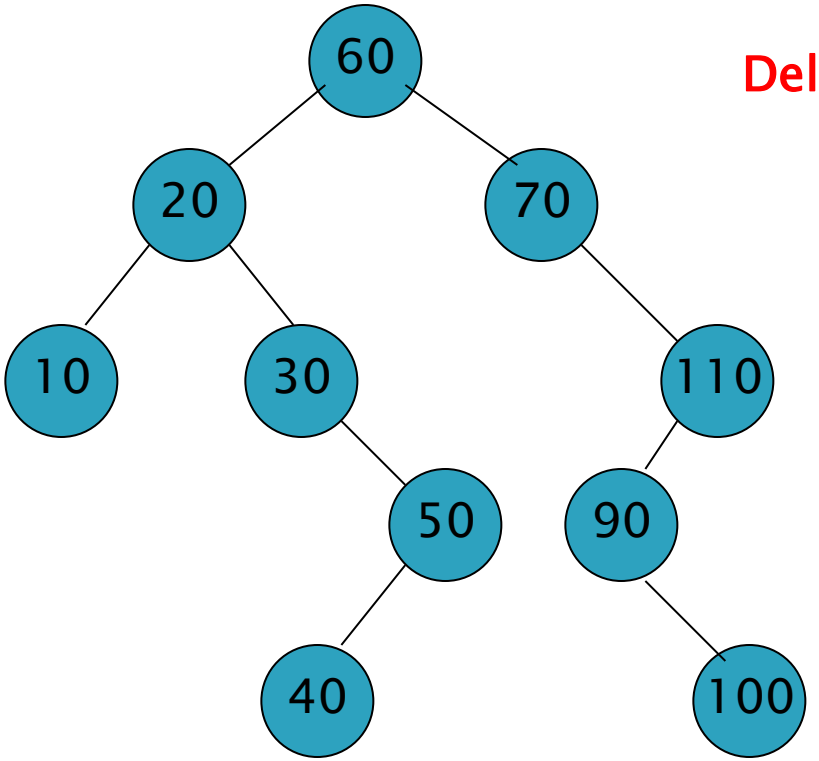
- Node to be deleted has no children.
- Simplest case. Unlink the node from its parent.
- The parent will be linked with null in the place of the deleted node.

BST Deletion: Case 1



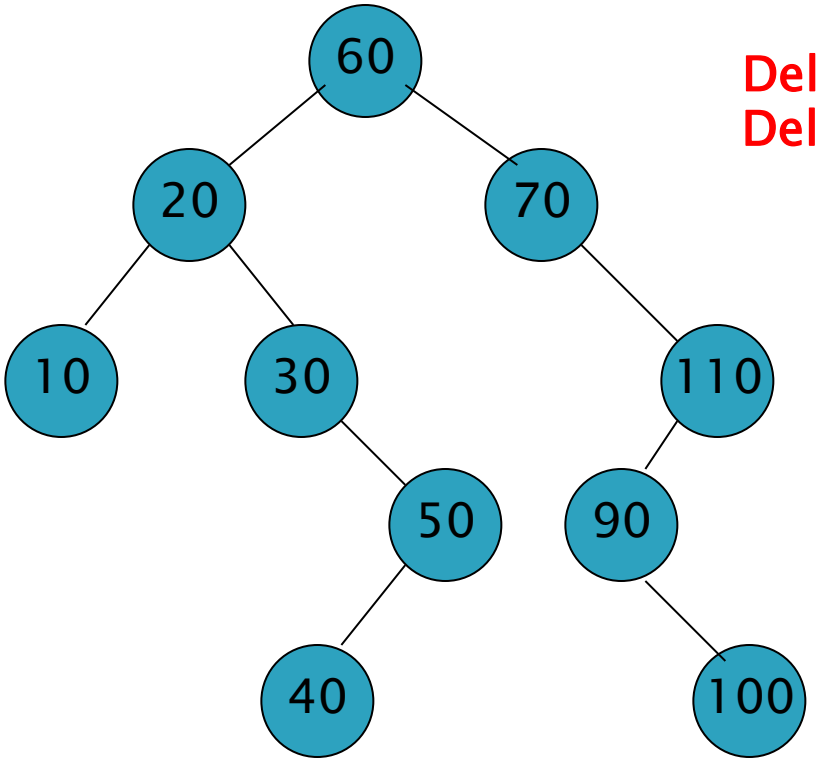
Delete 80

BST Deletion: Case 1



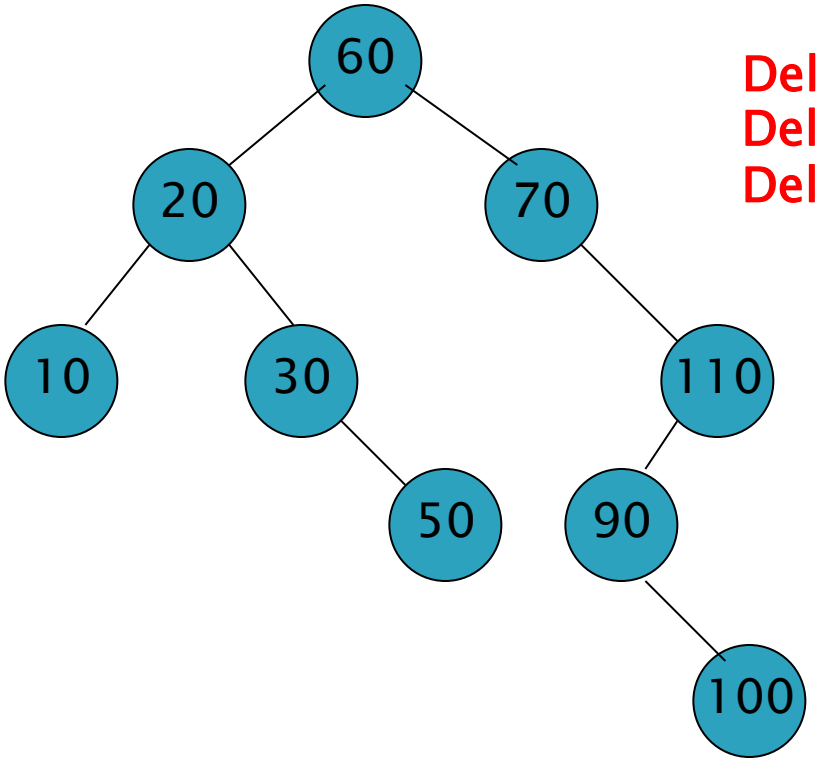
Delete 80

BST Deletion: Case 1



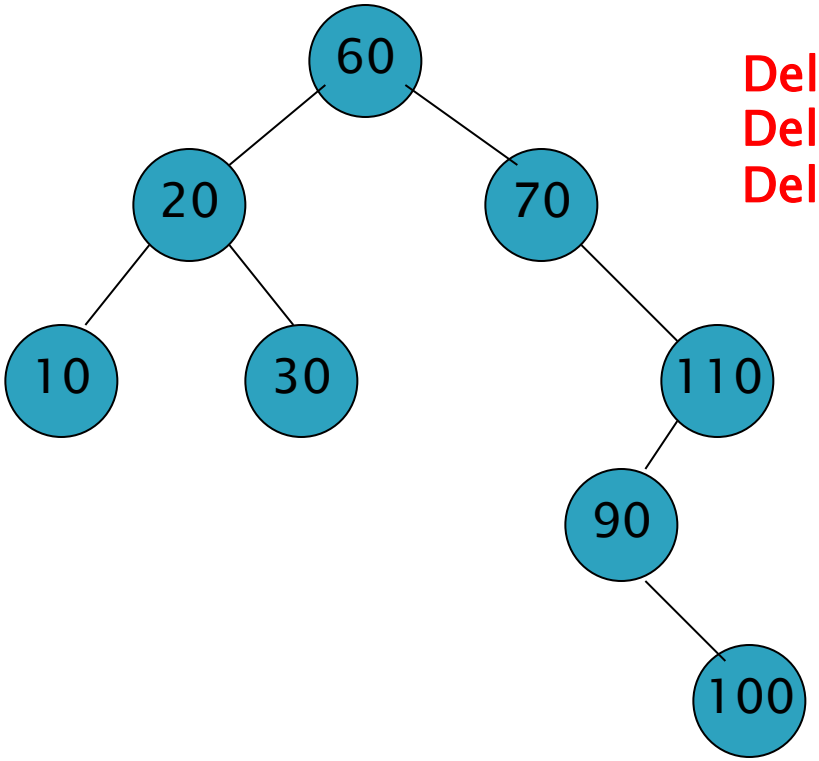
Delete 80
Delete 40

BST Deletion: Case 1



Delete 80
Delete 40
Delete 50

BST Deletion: Case 1

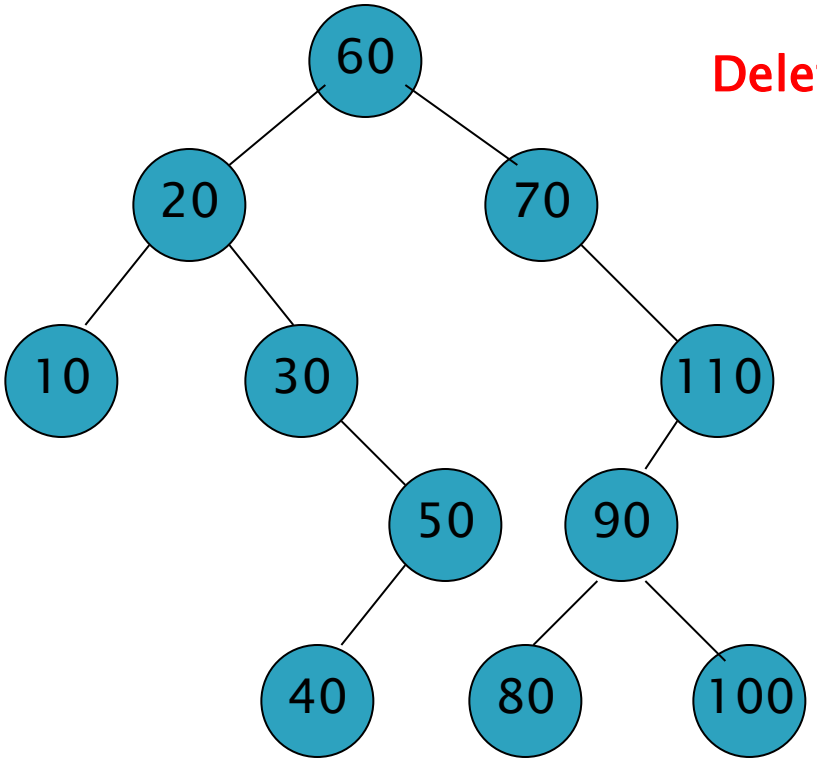


Delete 80
Delete 40
Delete 50

BST Deletion: Case 2

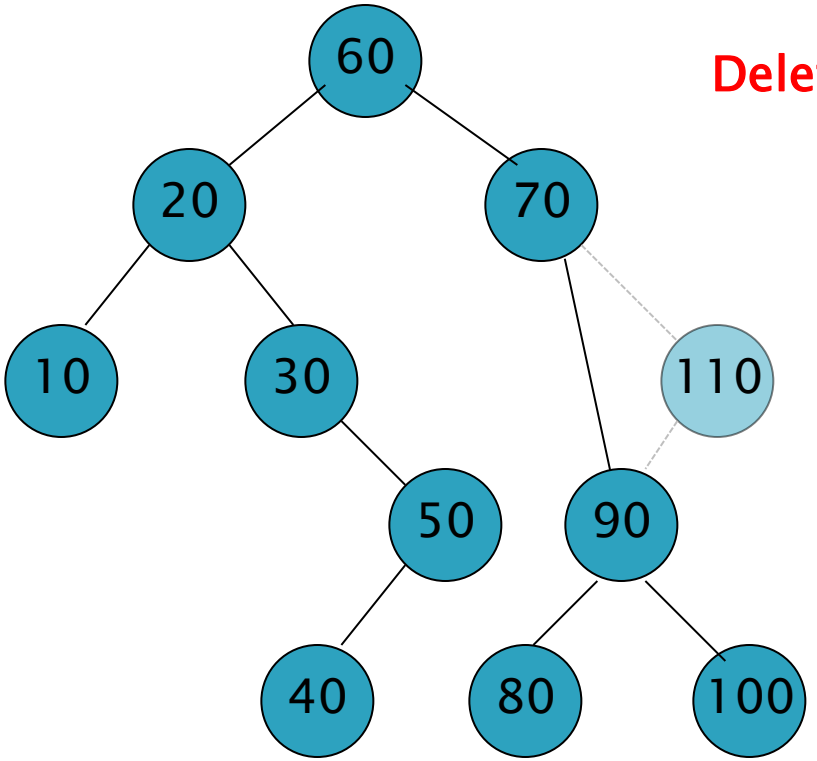
- Node to be deleted has one child.
- Remove the node, and place its child (along with its subtree) in its place.
- The parent will be linked with the child of the deleted node.

BST Deletion: Case 2



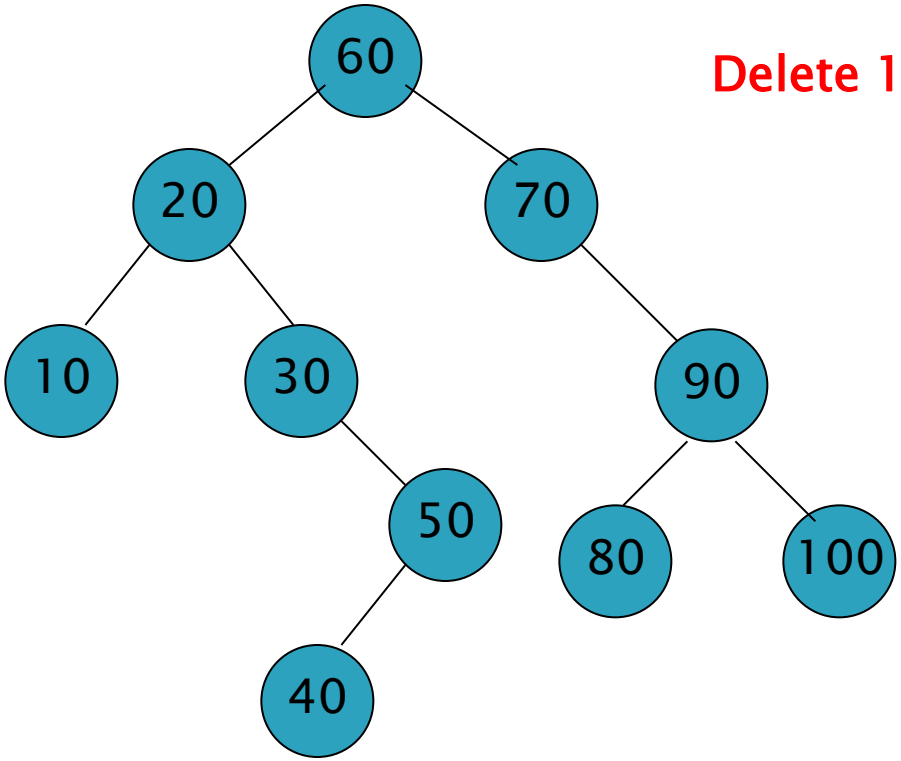
Delete 110

BST Deletion: Case 2



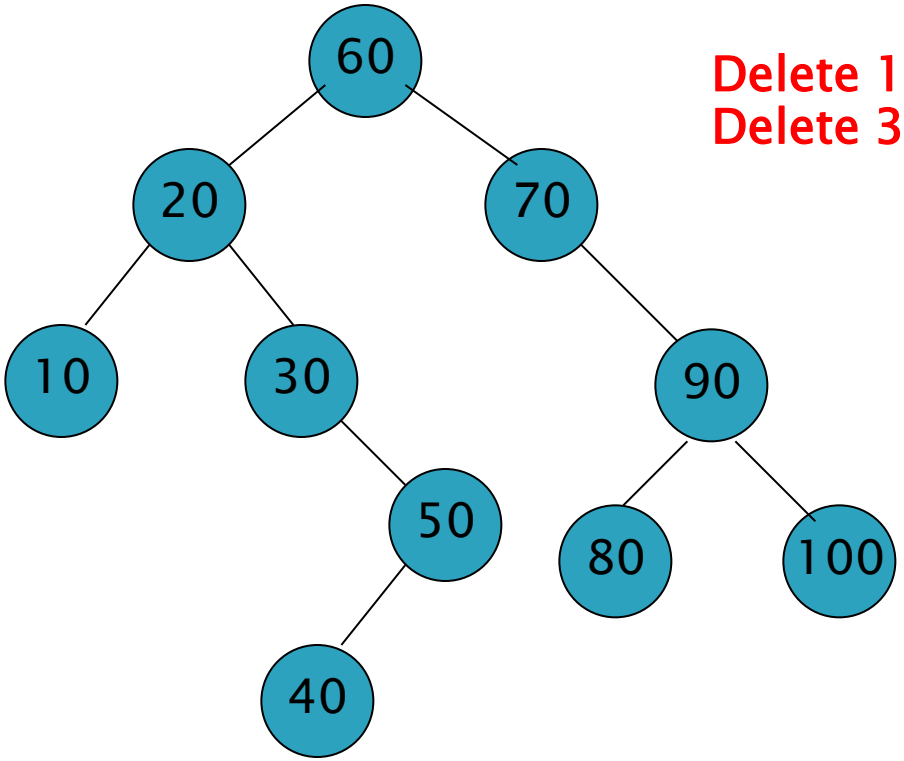
Delete 110

BST Deletion: Case 2



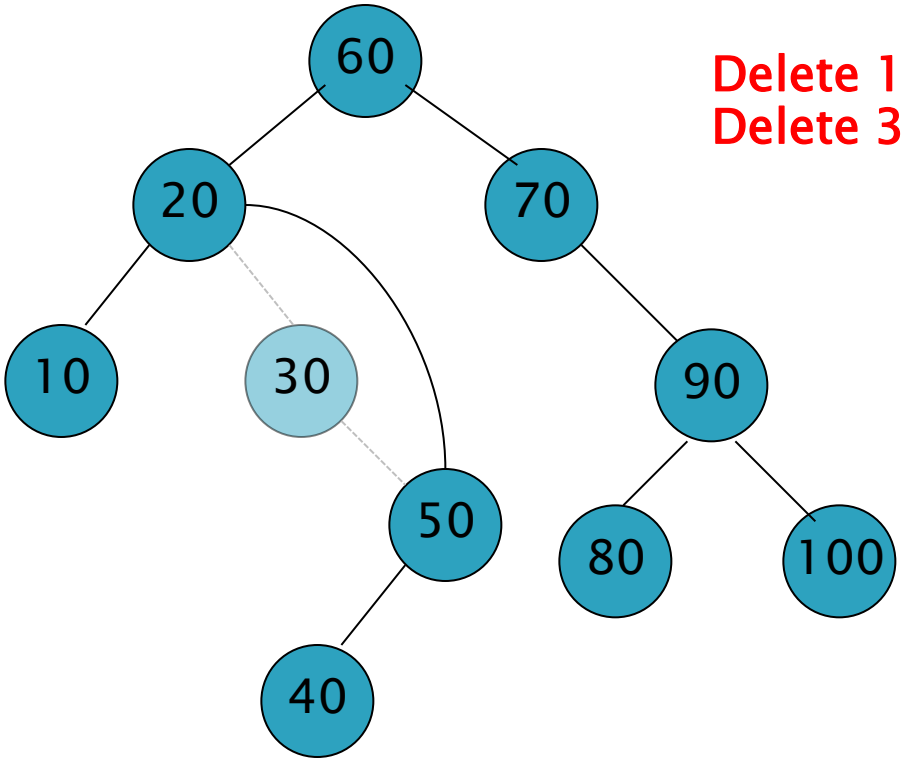
Delete 110

BST Deletion: Case 2



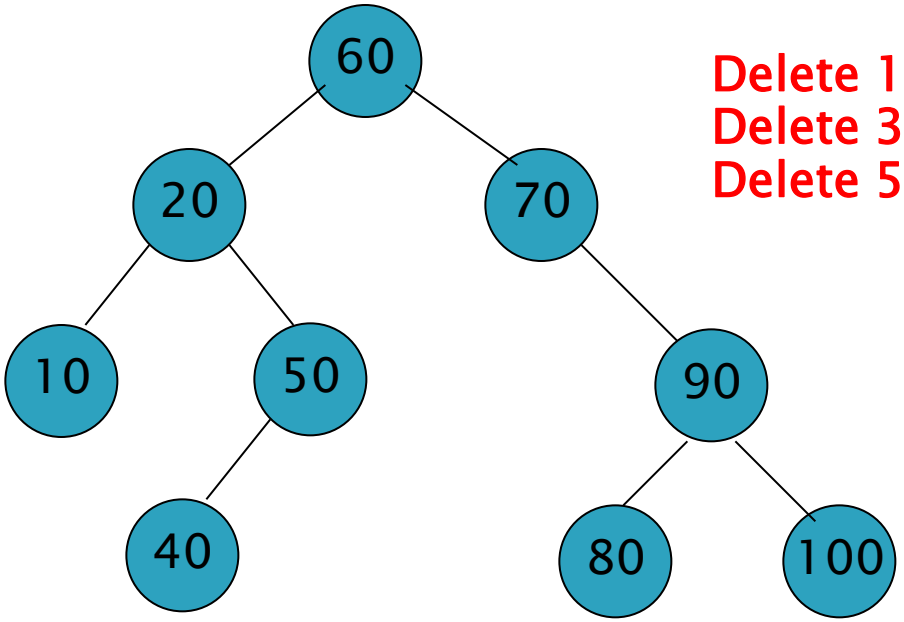
Delete 110
Delete 30

BST Deletion: Case 2



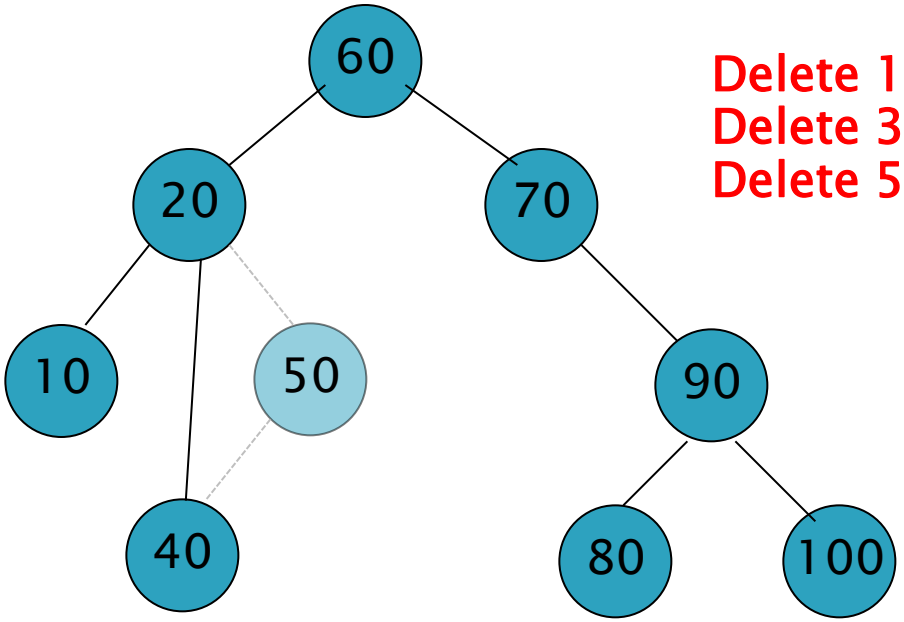
Delete 110
Delete 30

BST Deletion: Case 2



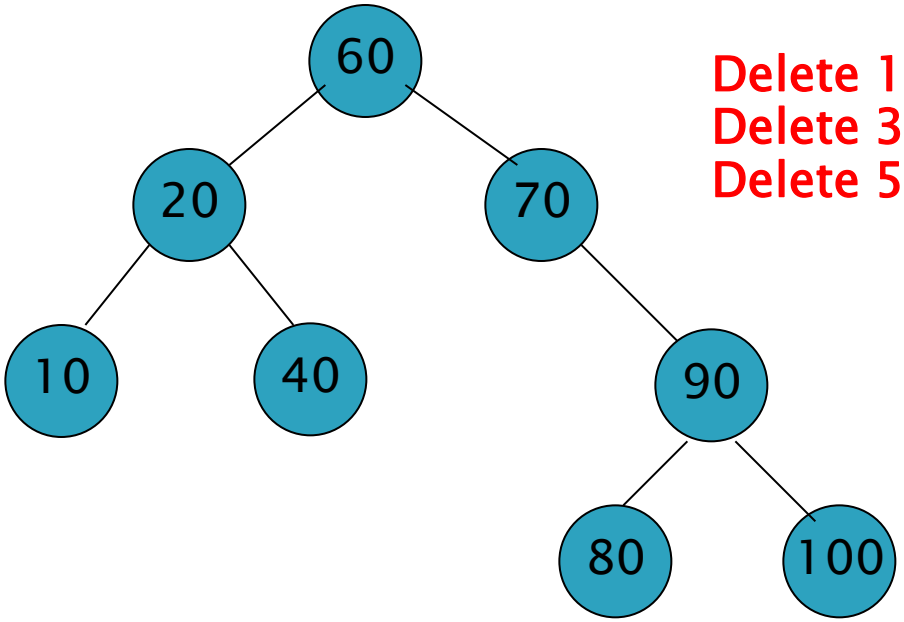
Delete 110
Delete 30
Delete 50

BST Deletion: Case 2



Delete 110
Delete 30
Delete 50

BST Deletion: Case 2

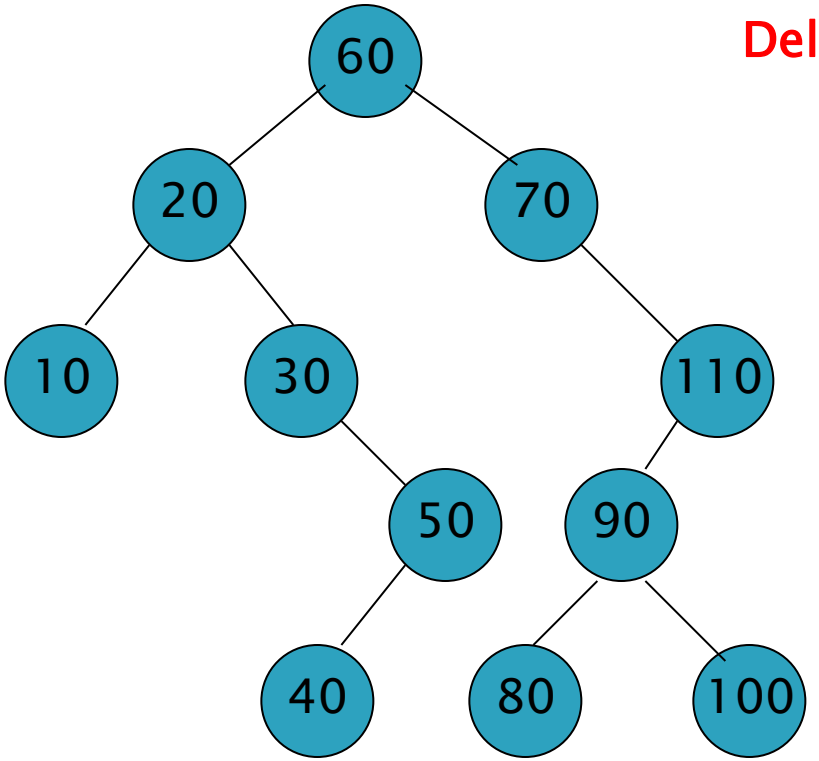


Delete 110
Delete 30
Delete 50

BST Deletion: Case 3

- Node to be deleted has two children.
- Complex case:
  - Find the node with the minimum key in the right subtree (left-most node in the right subtree).
  - Copy its key/data over the node to be deleted.
  - Delete the duplicate node (using either Case 1 or 2)
- The node will be overwritten by the minimum node in the right subtree. Then that duplicate node will be deleted.

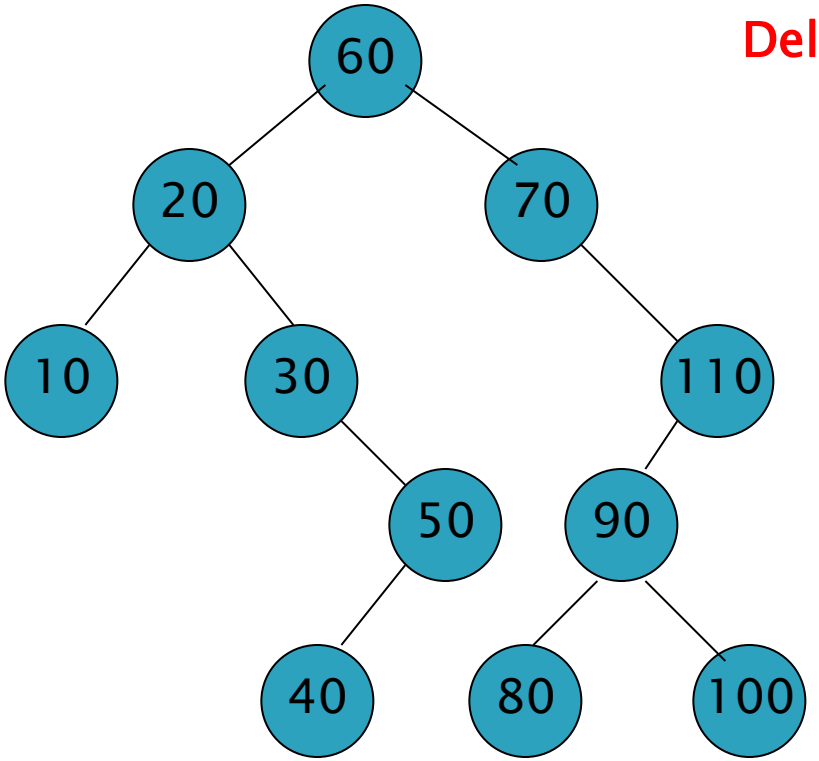
BST Deletion: Case 3



Delete 60

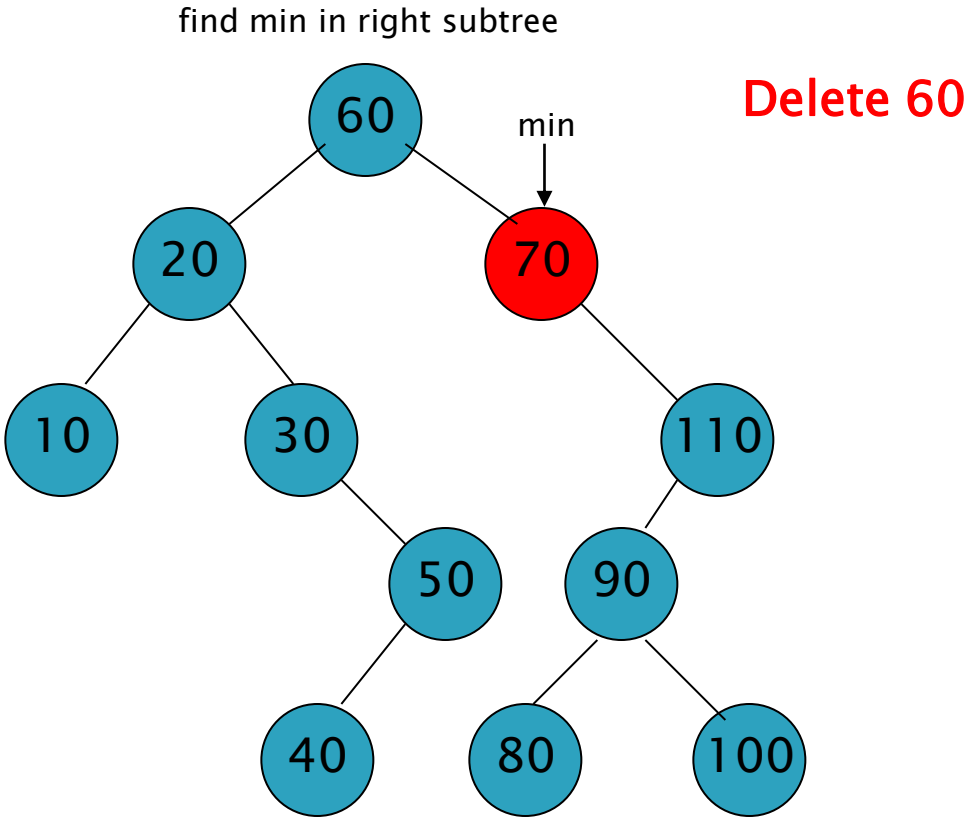
BST Deletion: Case 3

find min in right subtree

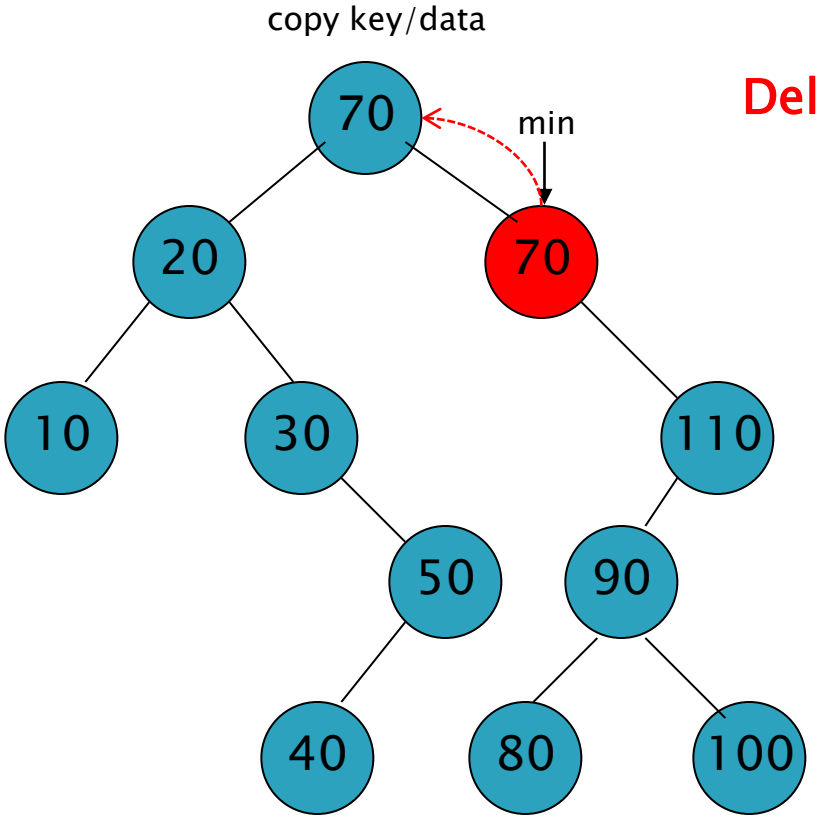


Delete 60

BST Deletion: Case 3

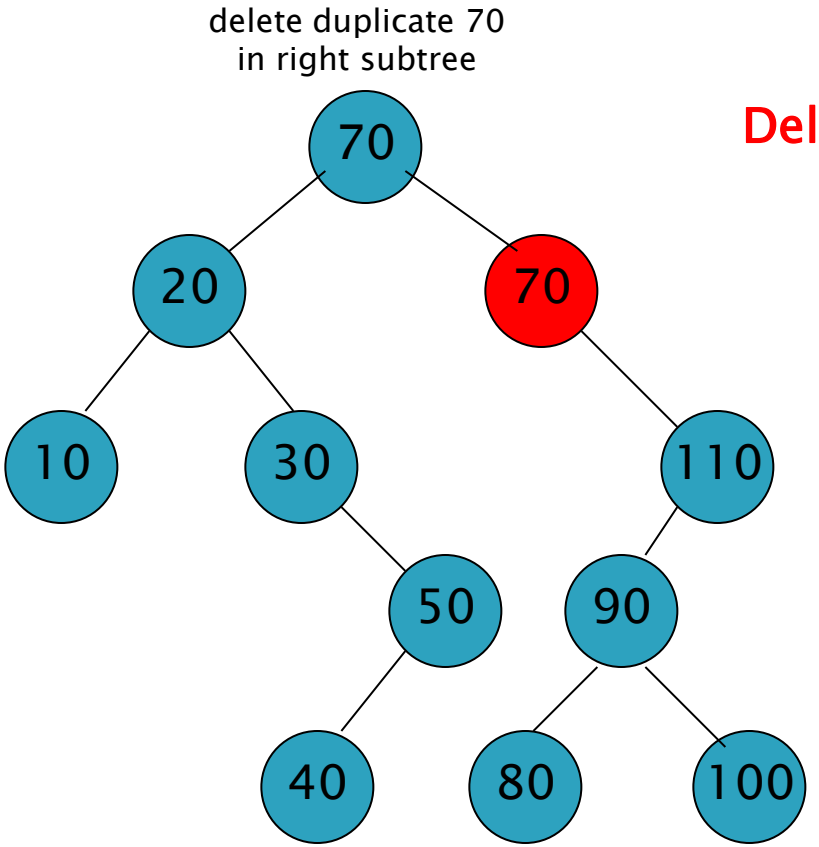


BST Deletion: Case 3



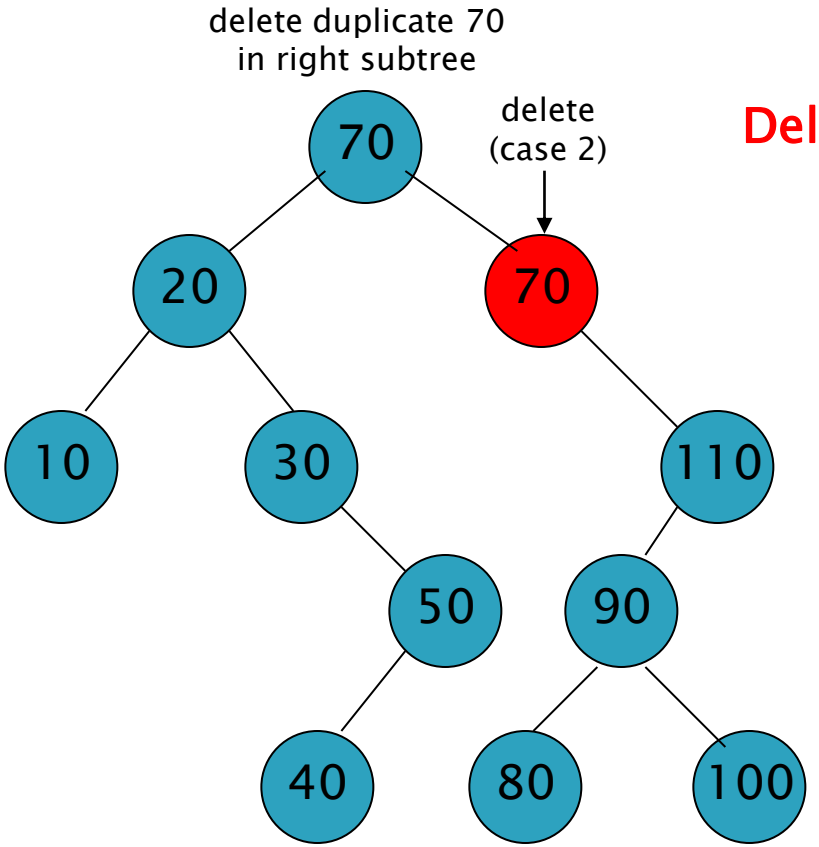
Delete 60

BST Deletion: Case 3



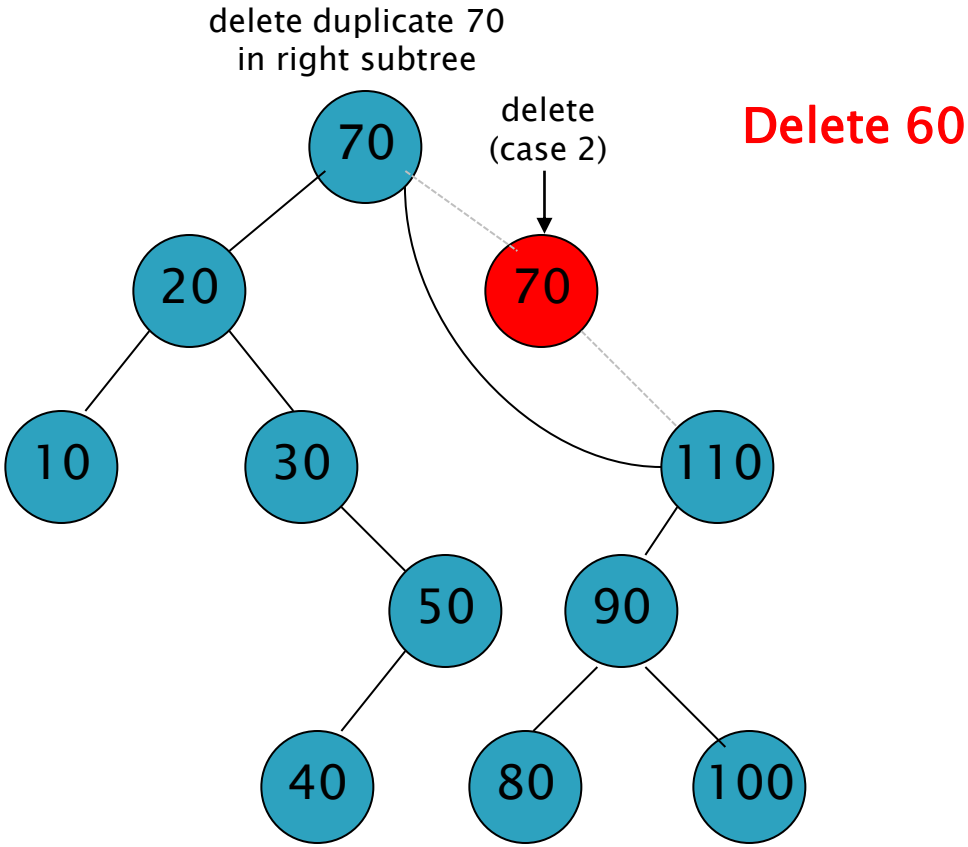
Delete 60

BST Deletion: Case 3



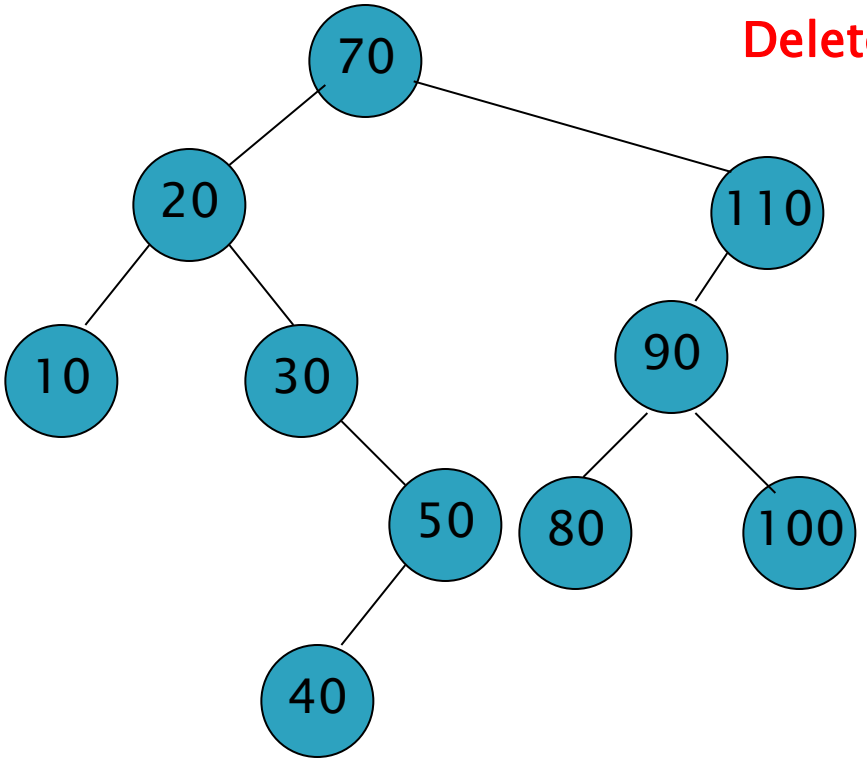
Delete 60

BST Deletion: Case 3

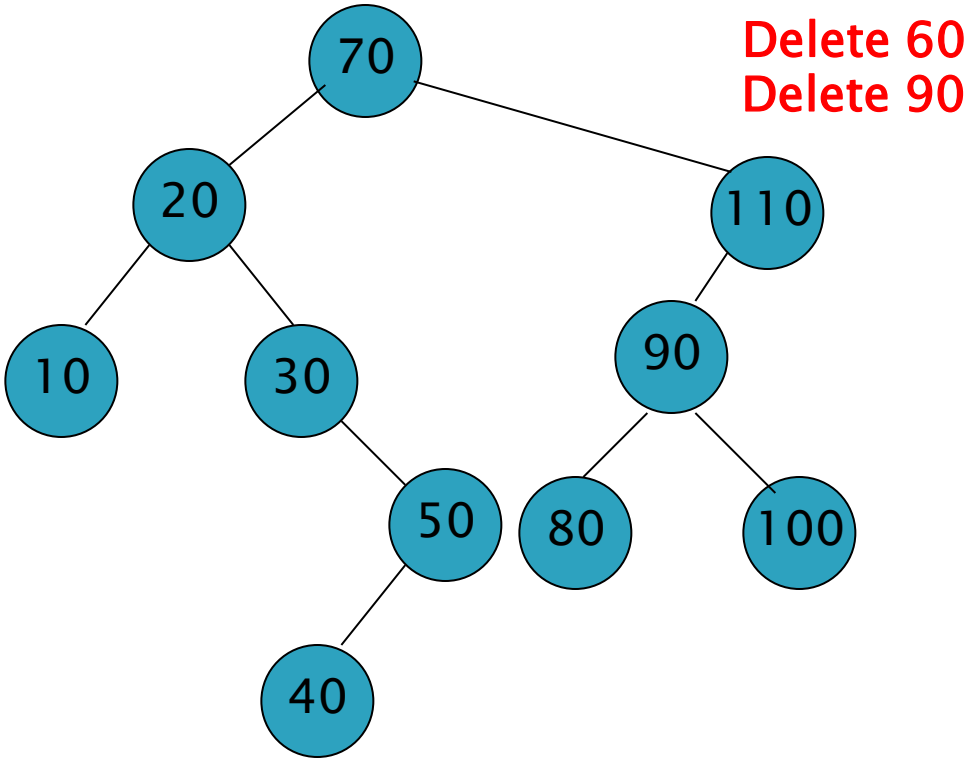


Delete 60

BST Deletion: Case 3

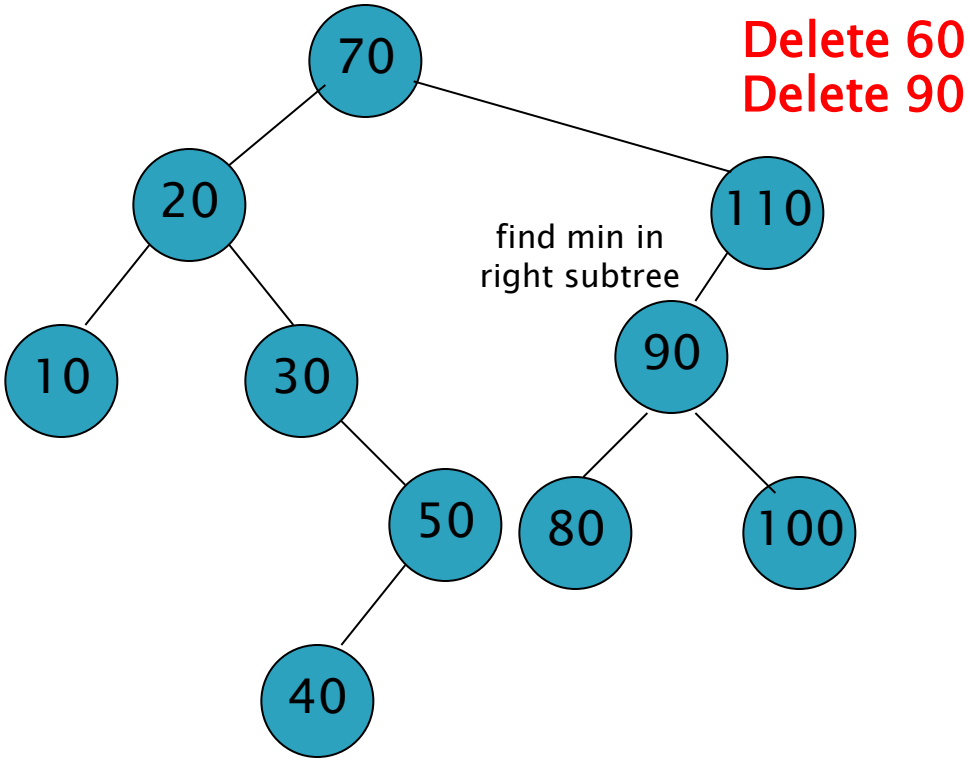


BST Deletion: Case 3

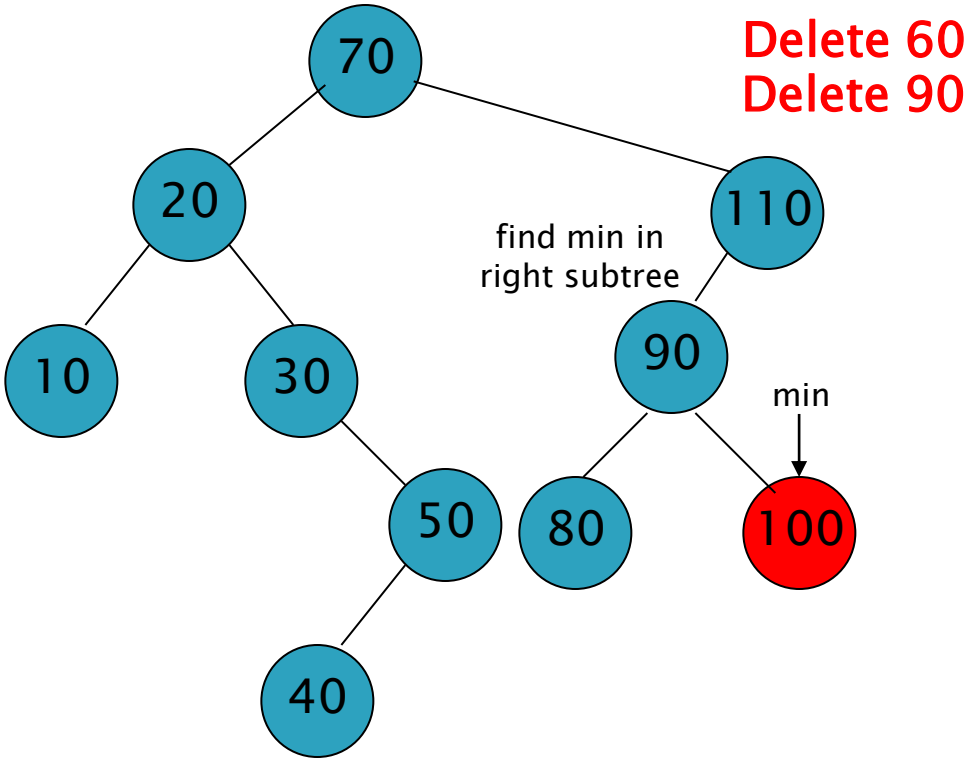


Delete 60
Delete 90

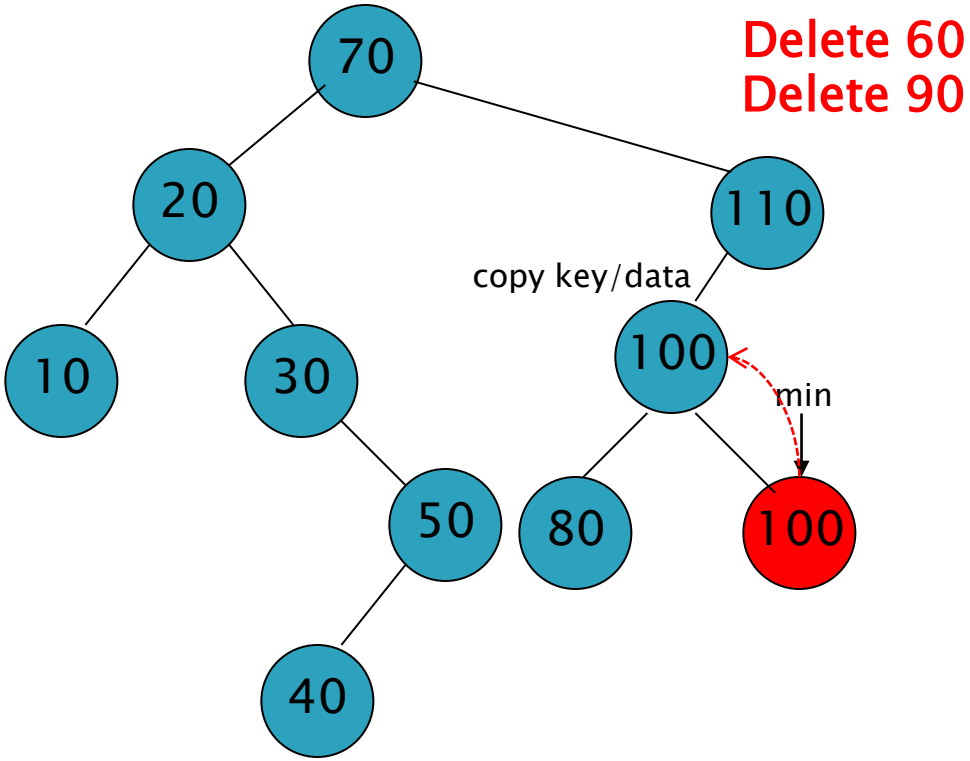
BST Deletion: Case 3



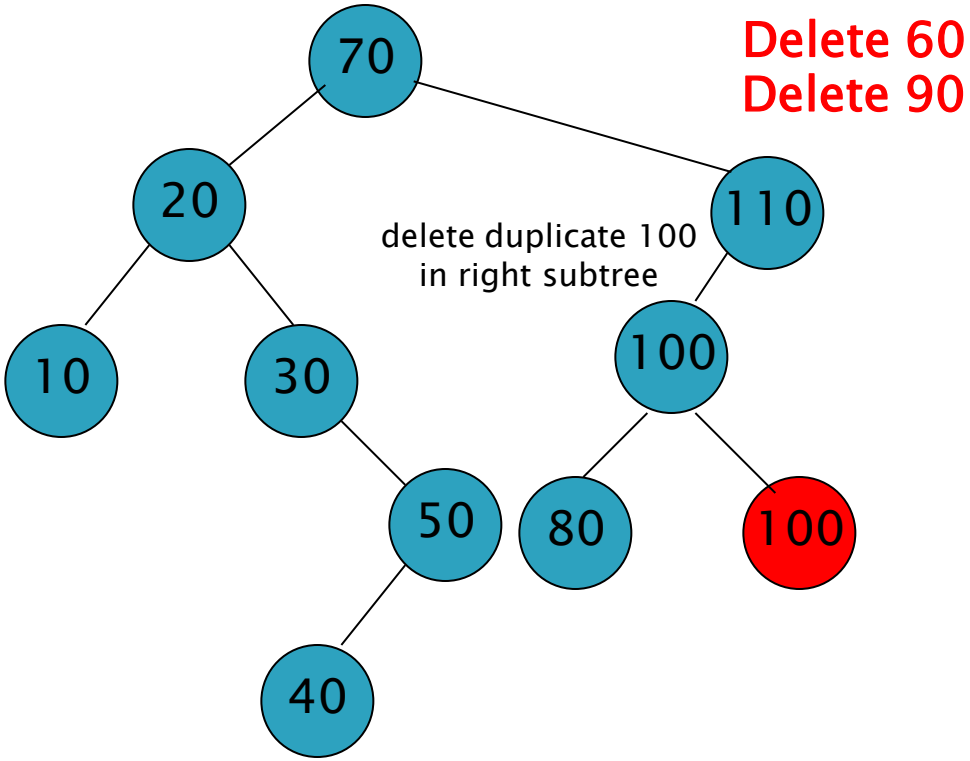
BST Deletion: Case 3



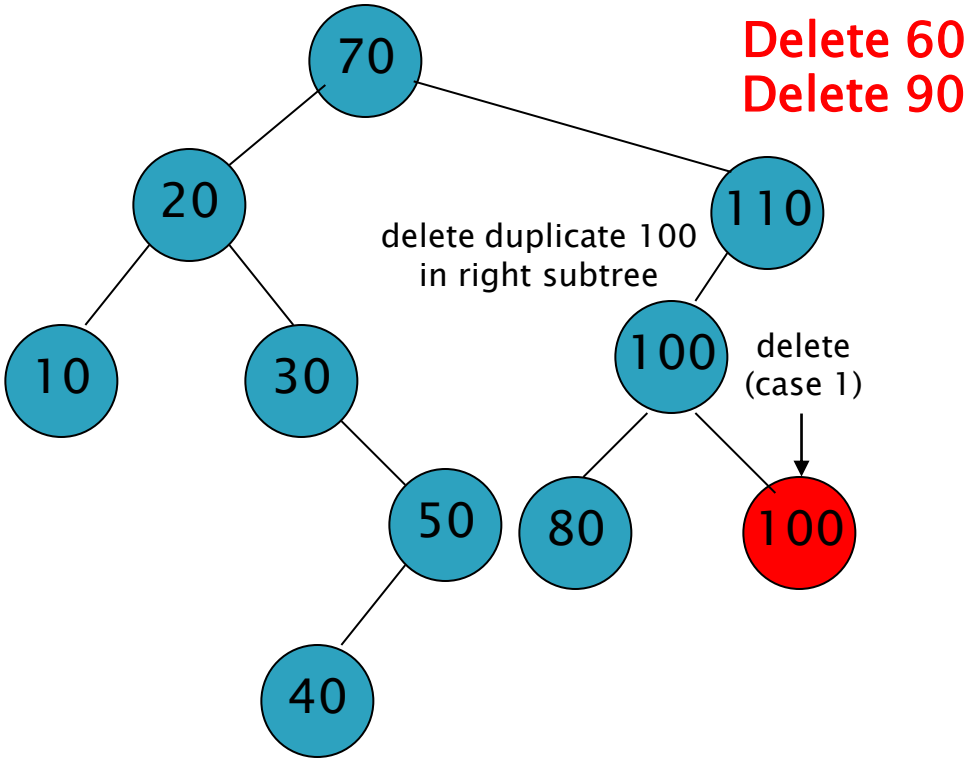
BST Deletion: Case 3



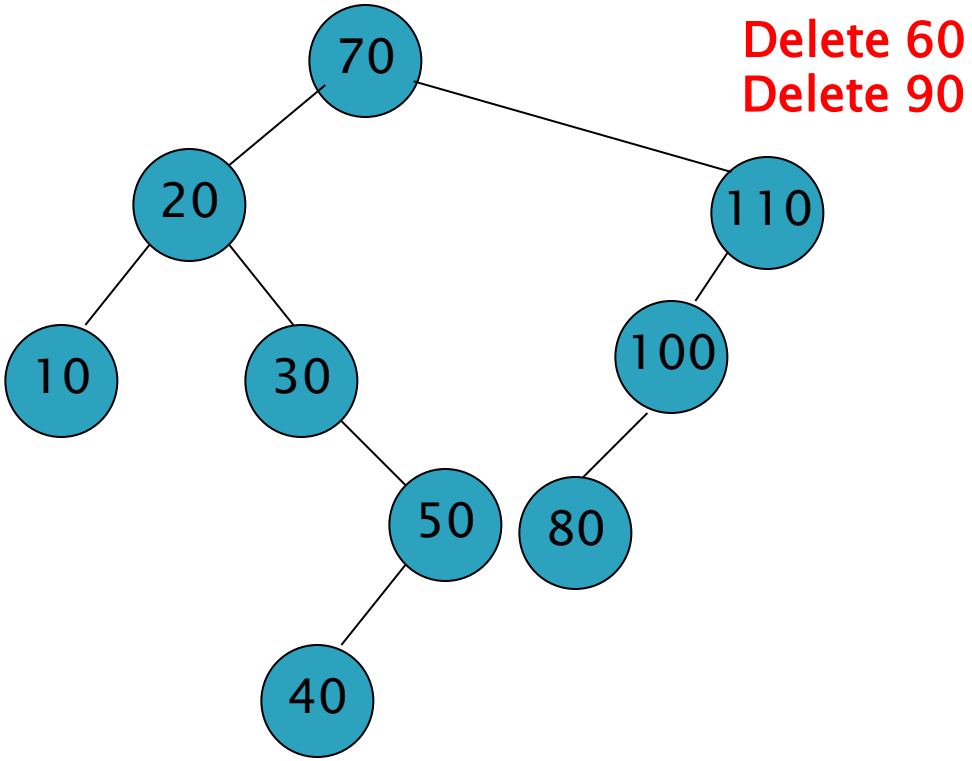
BST Deletion: Case 3



BST Deletion: Case 3

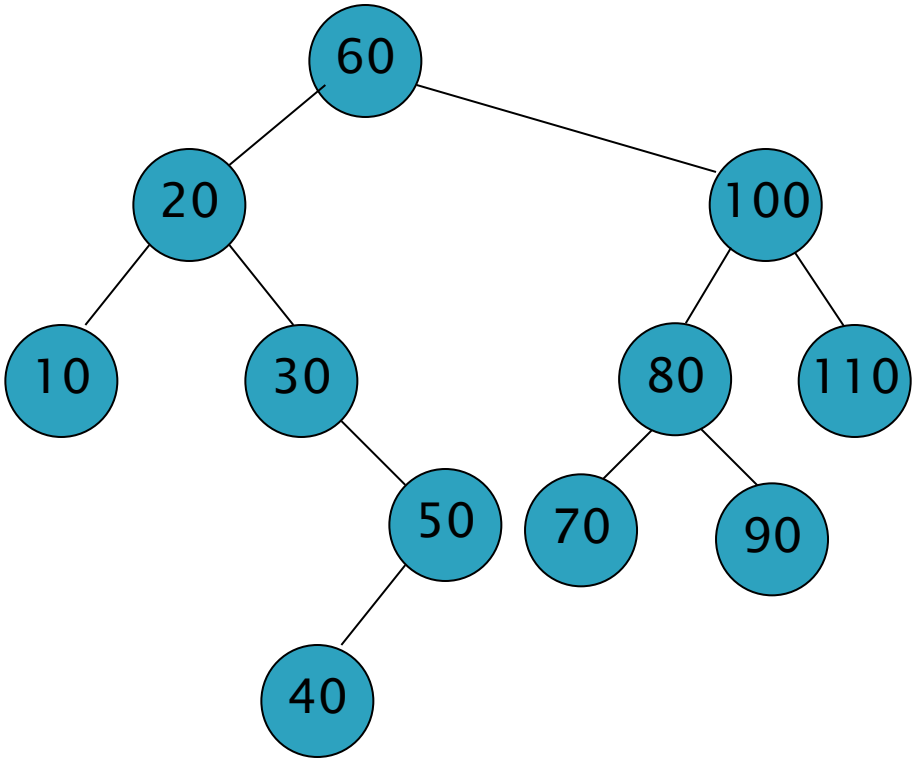


BST Deletion: Case 3



BST Deletion: Case 3

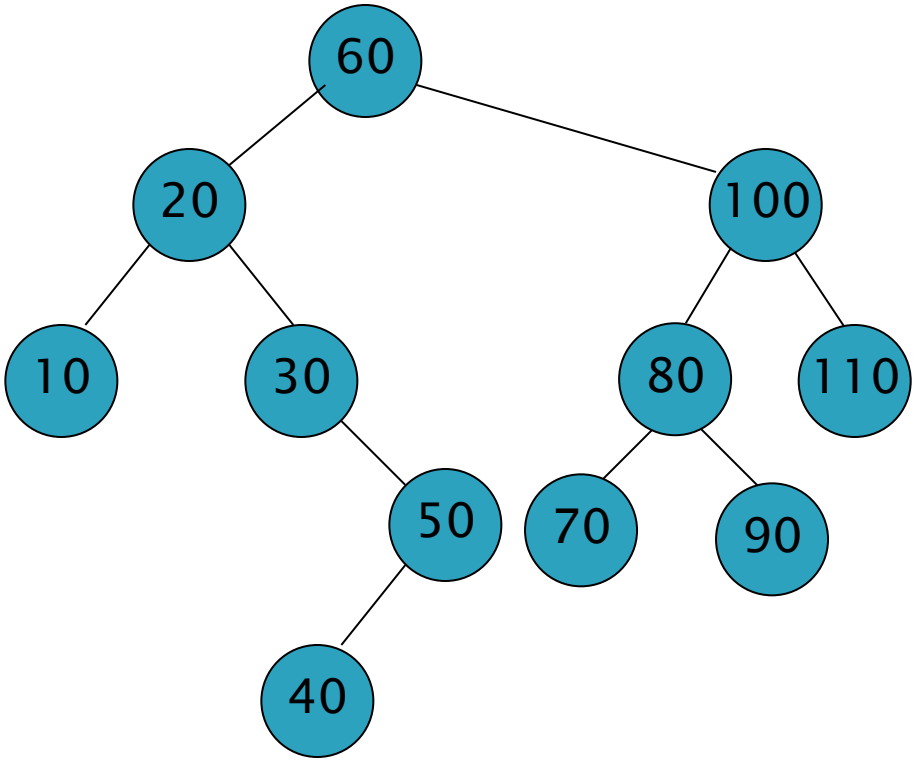
Delete 60



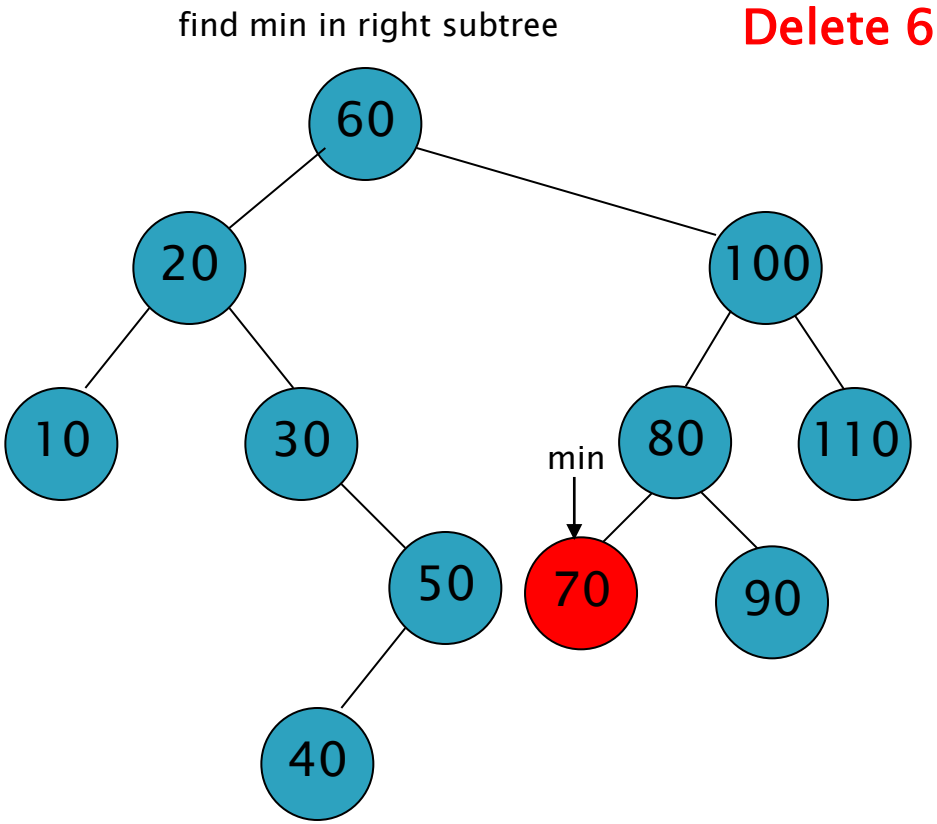
BST Deletion: Case 3

find min in right subtree

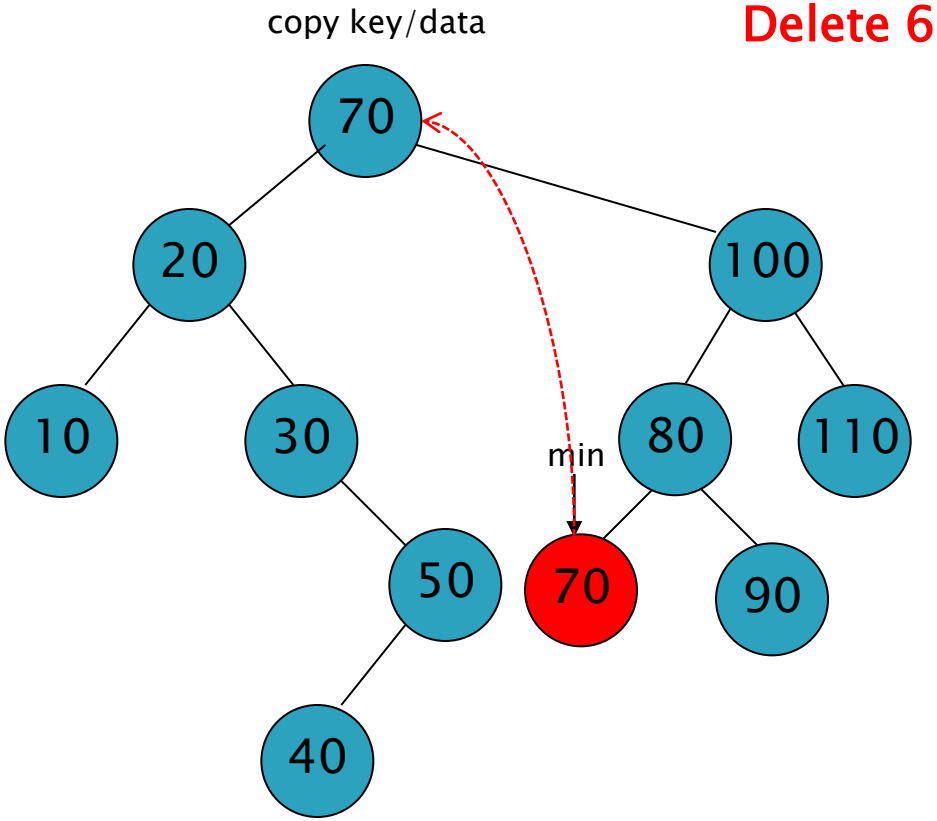
Delete 60



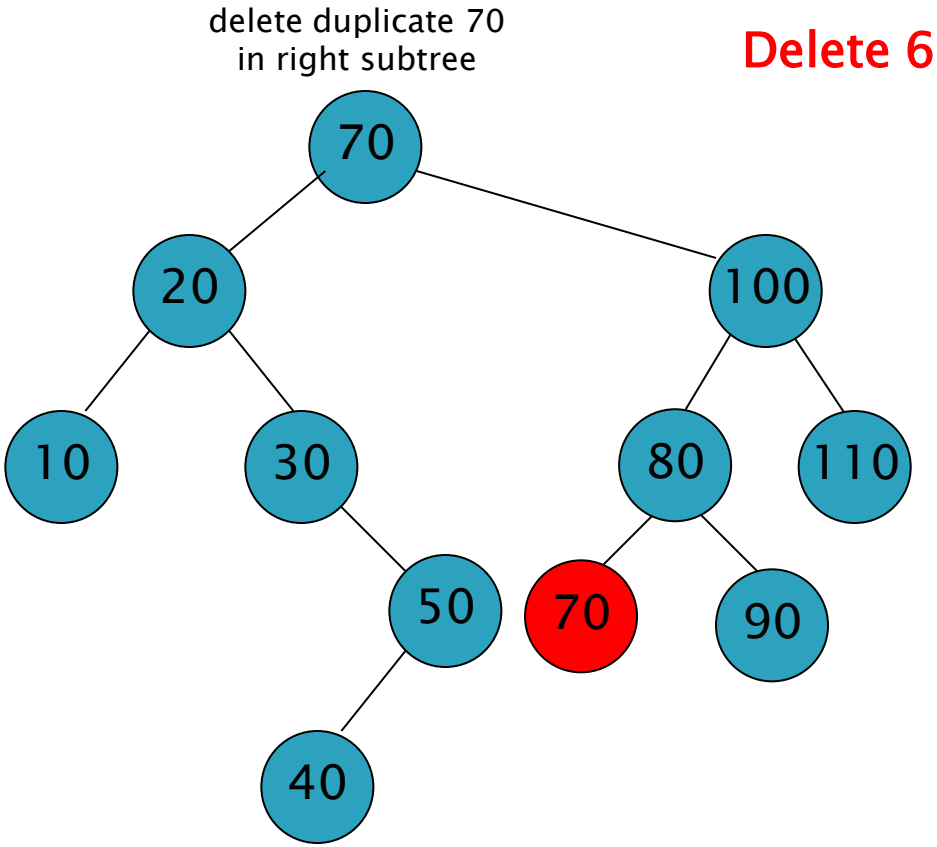
BST Deletion: Case 3



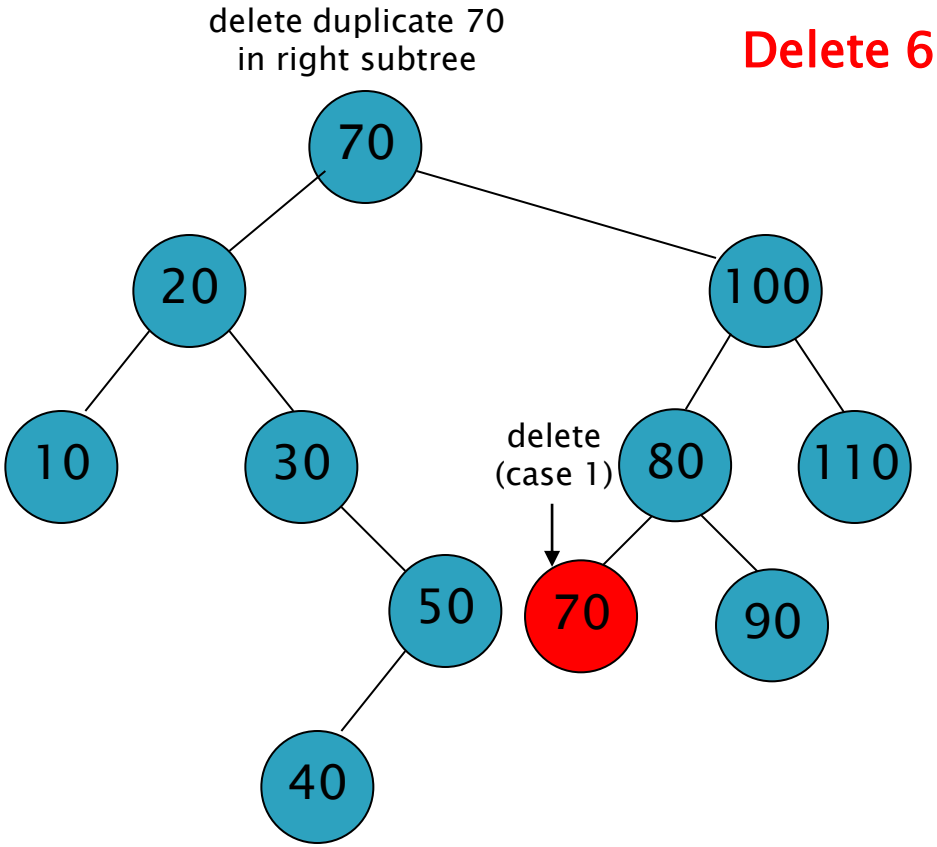
BST Deletion: Case 3



BST Deletion: Case 3

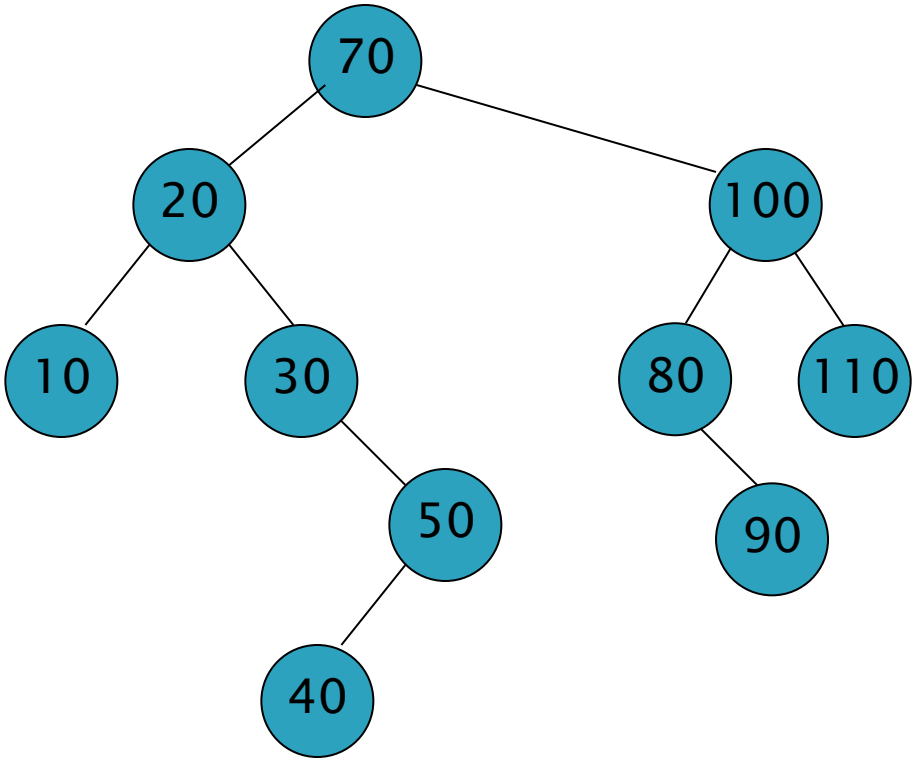


BST Deletion: Case 3



BST Deletion: Case 3

Delete 60



ADT Binary Search Tree: Implementation

```
public boolean remove_key (int tkey){
    Boolean removed = new Boolean(false);
    BSTNode<T> p;
    p = remove_aux(tkey, root, removed);
    current = root = p;
    return removed;
}
```

ADT Binary Search Tree: Implementation

```
public boolean remove_key (int tkey){
    Boolean removed = new Boolean(false);
    BSTNode<T> p;
    p = remove_aux(tkey, root, removed);
    current = root = p;
    return removed;
}
```

Traverse the tree to find the key
and handle remove cases (all 3 cases).
If found, it will *remove* and set removed
to (true). Otherwise, *removed*
will not change (false).
The method will return the modified tree.

ADT Binary Search Tree: Implementation

```
private BSTNode<T> remove_aux(int key, BSTNode<T> p, Boolean flag) {
     BSTNode<T> q, child = null;
     if(p == null)
             return null;
     if(key < p.key)
             p.left = remove_aux(key, p.left, flag); //go left
     else if(key > p.key)
             p.right = remove_aux(key, p.right, flag); //go right
     else {
             flag = true;
             if (p.left != null && p.right != null){ //two children
                     q = find_min(p.right);
                     p.key = q.key;
                     p.data = q.data;
                     p.right = remove_aux(q.key, p.right, flag);
             }
```

ADT Binary Search Tree: Implementation

```
            else {
                if (p.right == null) //one child
                    child = p.left;
                else if (p.left == null) //one child
                    child = p.right;
                return child;
            }
        }
        return p;
    }
```

ADT Binary Search Tree: Implementation

```
private BSTNode<T> find_min(BSTNode<T> p){
     if(p == null)
             return null;

     while(p.left != null){
             p = p.left;
     }

     return p;
}
```

Find left-most node (minimum key node) in any tree p

ADT Binary Search Tree: Implementation

```
    public boolean update(int key, T data){
        remove_key(current.key);
        return insert(key, data);
    }

}
```

To update the current key/value:
1). Remove the current node.
2). Insert a new node with the new key/data.
Note: The new node will be set the current after insert.