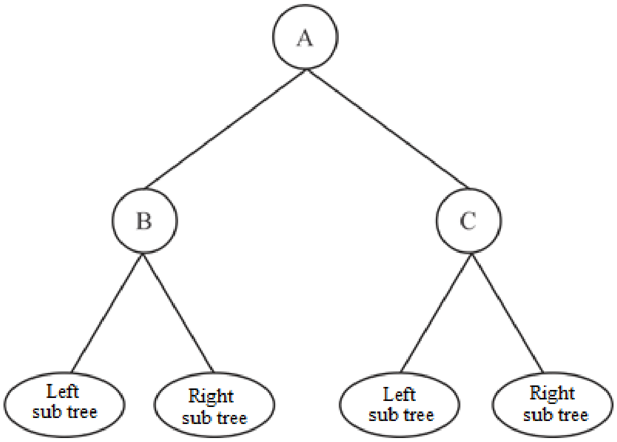
If in-order tree walk prints a sequence …*xyz*…, then *x* is called predecessor of *y* and *z* is called successor of *y*.

For suppose, consider a node *A* that has two children *B* and *C*, in the binary search tree. Also assume, node *B* is the predecessor of v and the node *C* is the successor of the v. That is , in-order tree walk algorithm prints a sequence …*BAC*…for the following tree.



Predecessor B has no right child:

• If the element *B* has a right child, then according to the key property of binary search tree, the right child must be greater than *B*.

• Thus, *B.right* will be printed after *B* and before *A*, by the in-order tree walk. That is, the sequence printed by the in-order tree walk is …*B,B.right,A,C*… . Hence, *B* is not the predecessor of *A*. but, it is the contradiction to our assumption.

Therefore, if *A* has two children *B,C* and *B* is the predecessor of *A* and *C* is the successor of *A* , then the predecessor of *A*(*B*) has no right child. That is, *B.right* is *NIL*.

Successor C has no left child:

• If the element *C* has a left child, then according to the property of binary search tree, the *C.left* must be less than or equal to *C*.

• Thus, *C.left* will be printed after *A* and before *C*, by the in-order tree walk. That is, the sequence printed by in-order tree walk is …*B,A,C.left,C*… . Hence, *C* is not the successor of *A*. but, it is the contradiction to our assumption.

Therefore, if *A* has two children *B,C* and *B* is the predecessor of *A* and *C* is the successor of *A* , then the successor of *A*(*C*) has no left child.

Hence, if a node in the binary search tree has two children, then its predecessor has no right child and its predecessor has no right child.