AVL Trees (a few more slides)

CSE 373

Data Structures

Lecture 8.5

Insertion in AVL Trees

- Insert at the leaf (as for all BST)
  - › only nodes on the path from insertion point to root node have possibly changed in height
  - › So after the Insert, go back up to the root node by node, updating heights
  - › If a new balance factor (the difference h_{left}-h_{right}) is 2 or –2, adjust tree by *rotation* around the node

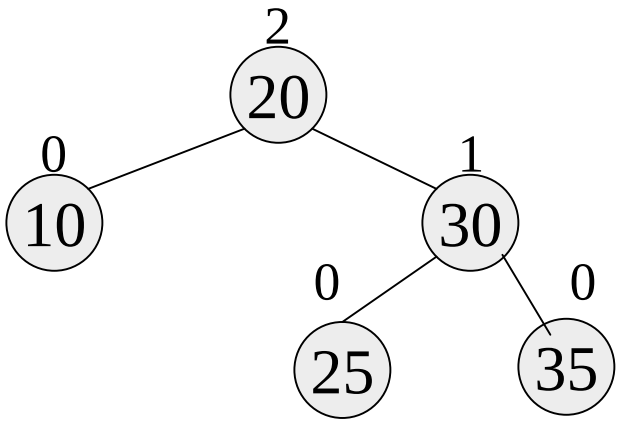
Insert in BST

```
Insert(T : reference tree pointer, x : element) : integer {
if T = null then
  T := new tree; T.data := x; return 1;//the links to
                                       //children are null
case
  T.data = x : return 0; //Duplicate do nothing
  T.data > x : return Insert(T.left, x);
  T.data < x : return Insert(T.right, x);
endcase
}
```

Insert in AVL trees

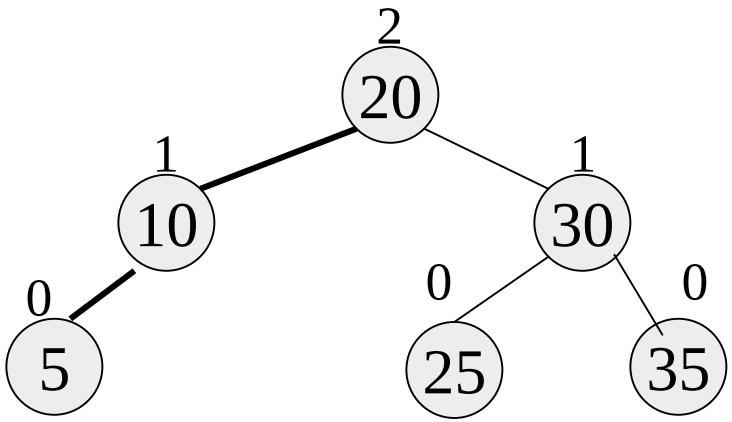
```
Insert(T : reference tree pointer, x : element) : {
if T = null then
  T := new tree; T.data := x; height := 0;
case
  T.data = x : return ; //Duplicate do nothing
  T.data > x : return Insert(T.left, x);
               if ((height(T.left)- height(T.right)) = 2){
                  if (T.left.data > x ) then //outside case
                         T = RotatefromLeft (T);
                  else                       //inside case
                         T = DoubleRotatefromLeft (T);}
  T.data < x : return Insert(T.right, x);
                code similar to the left case
Endcase
  T.height := max(height(T.left),height(T.right)) +1;
  return;
}
```

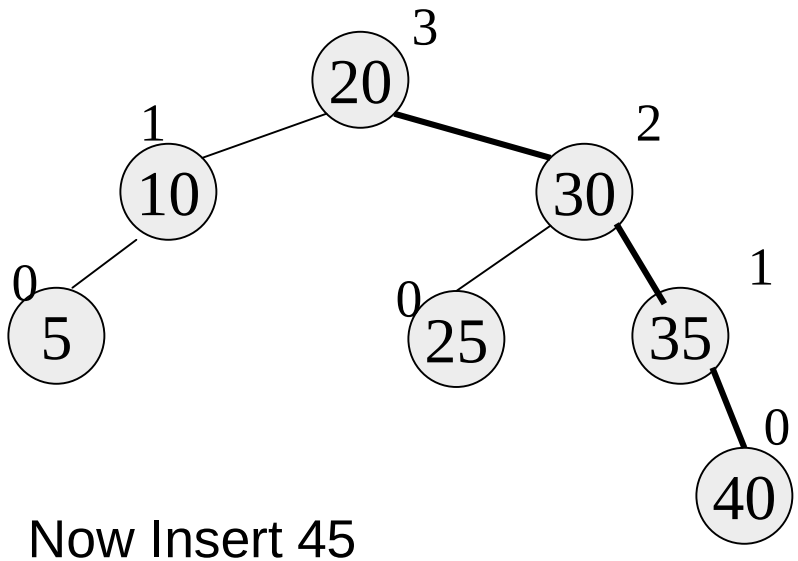
Example of Insertions in an AVL Tree



Insert 5, 40

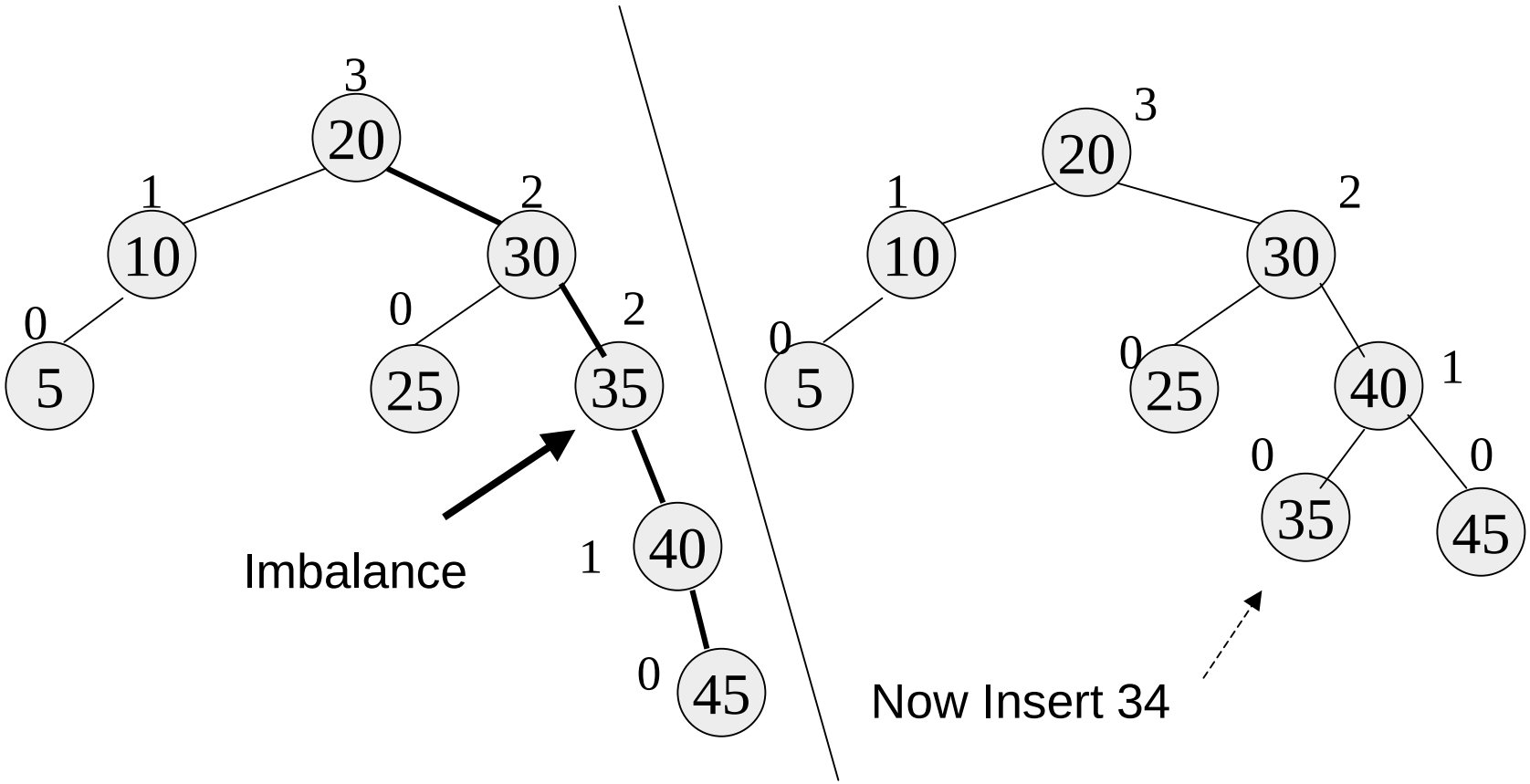
Example of Insertions in an AVL Tree



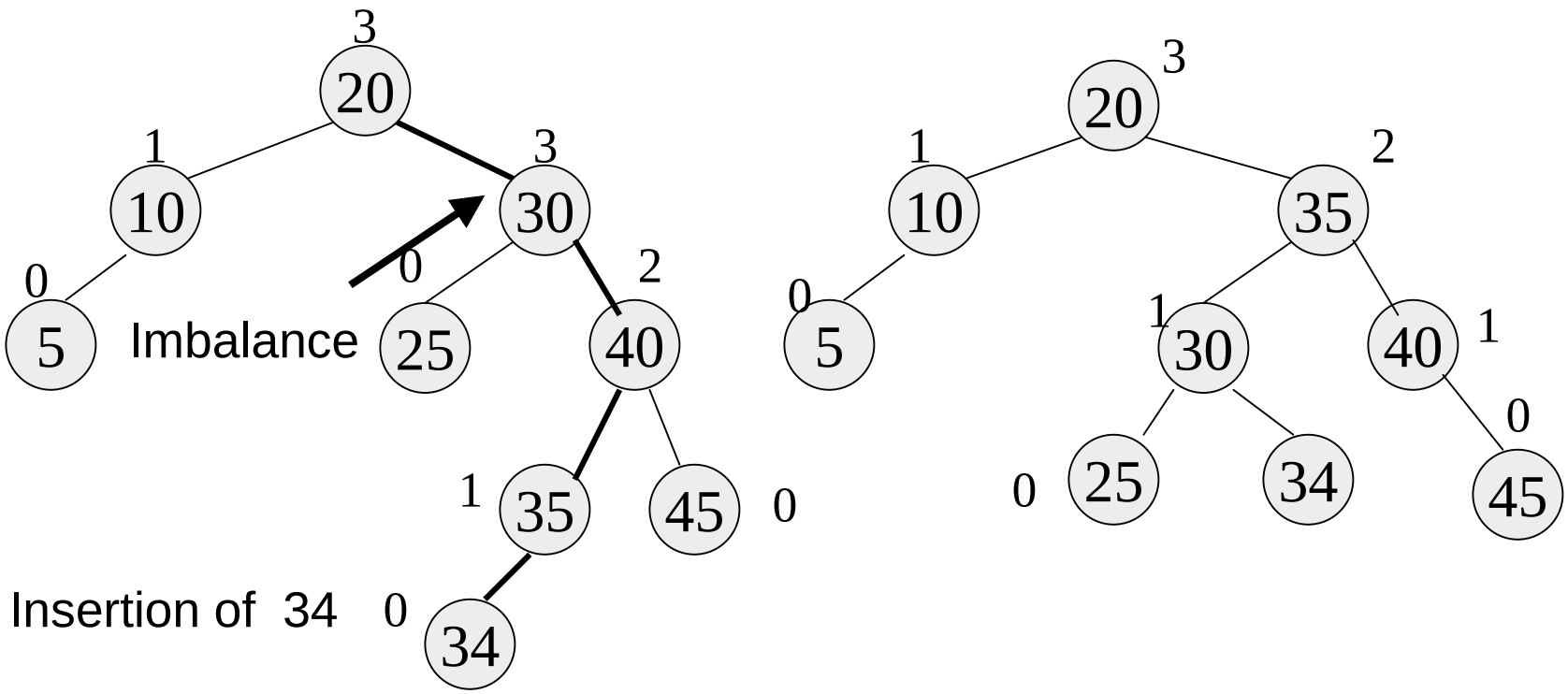


Now Insert 45

Single rotation (outside case)

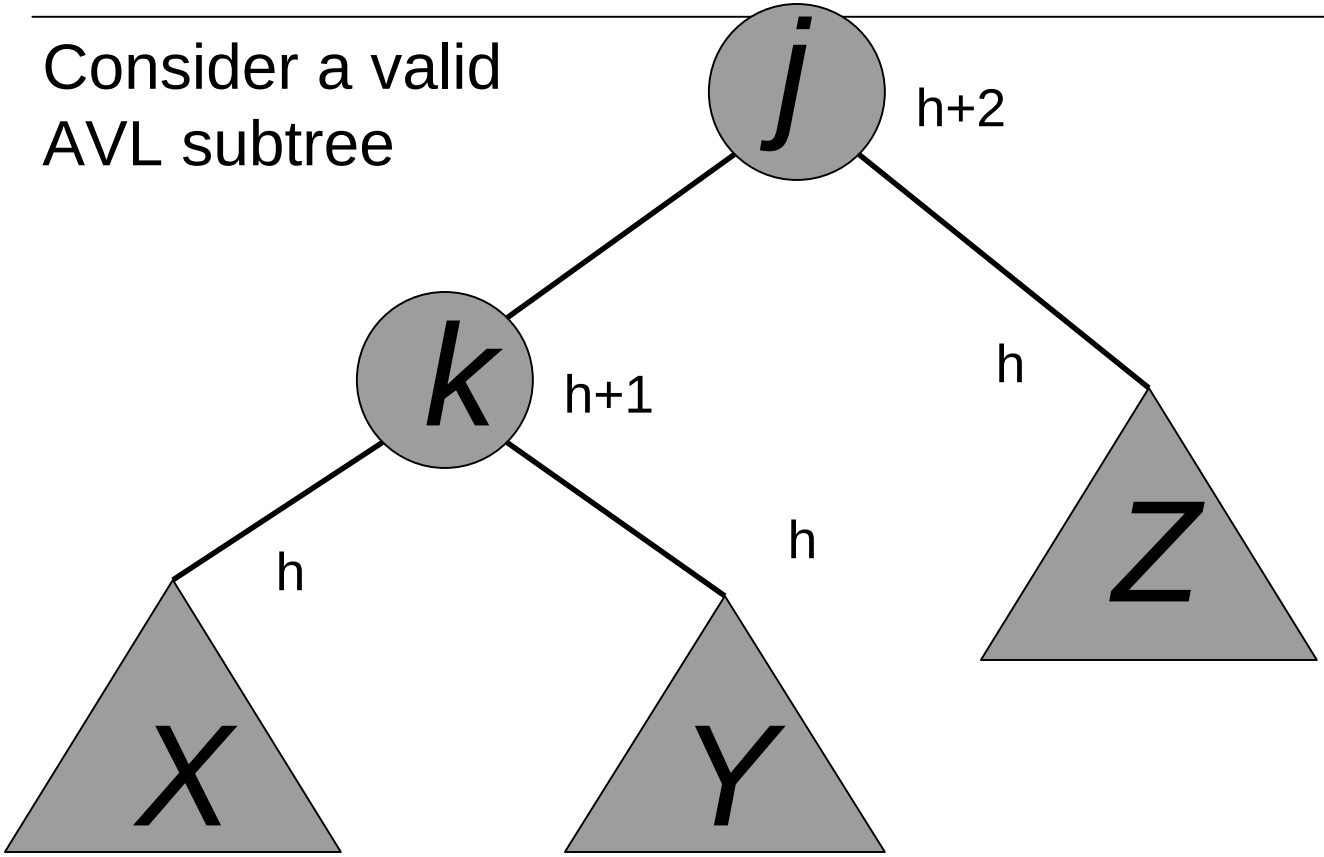


Double rotation (inside case)

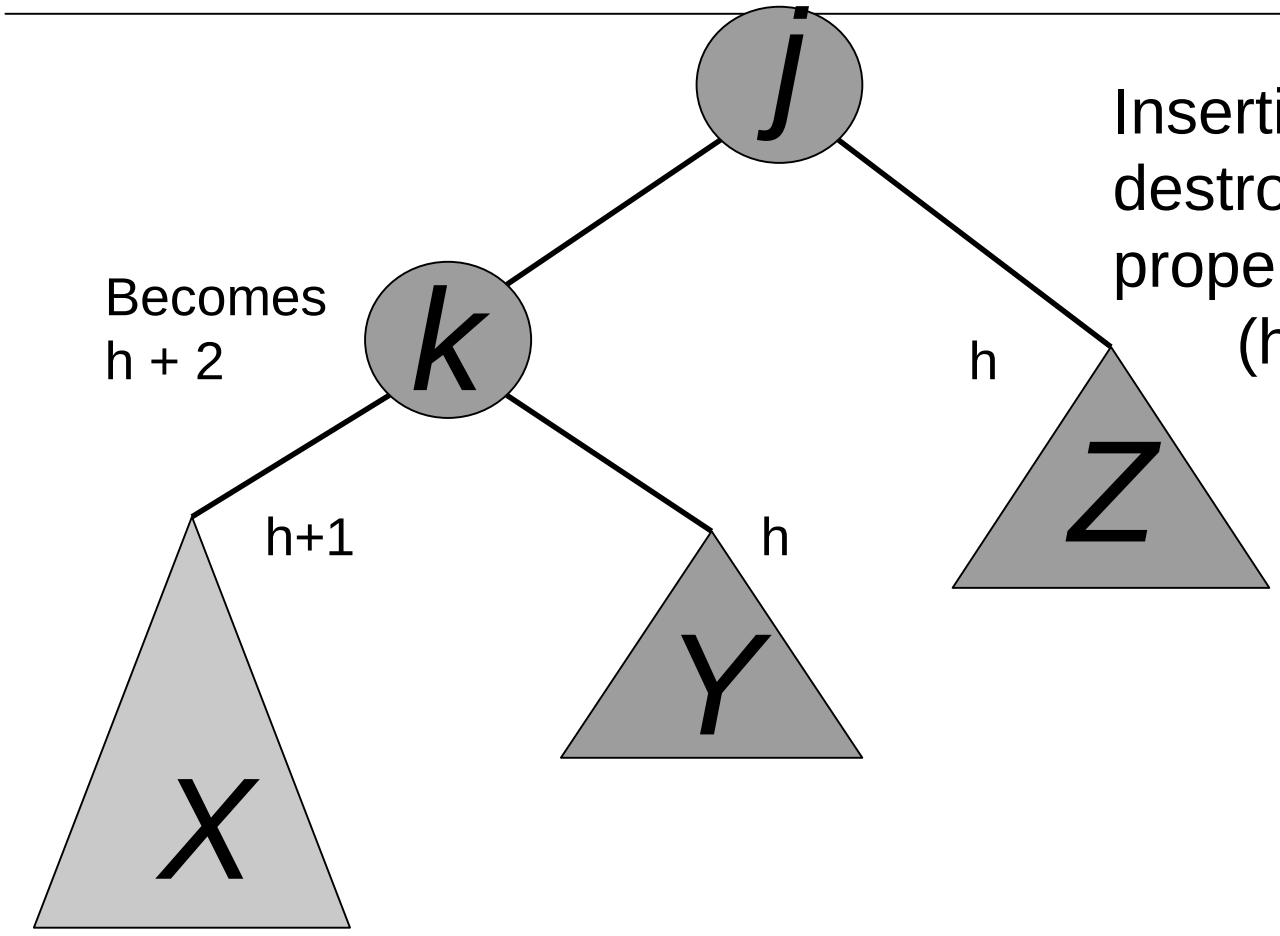


AVL Insertion: Outside Case

Consider a valid AVL subtree



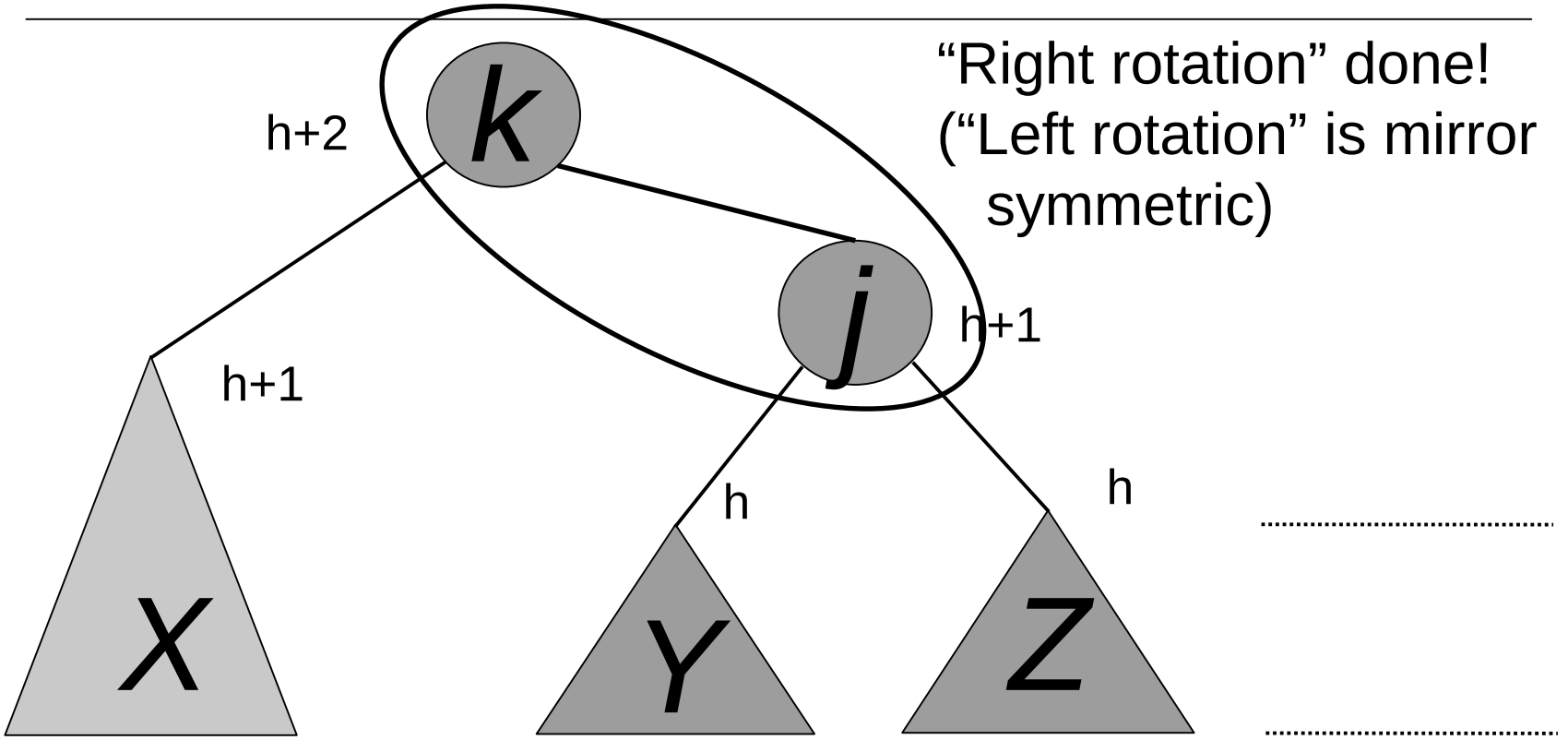
AVL Insertion: Outside Case



Inserting into X destroys the AVL property at node j

(h+2) - h

Outside Case Completed

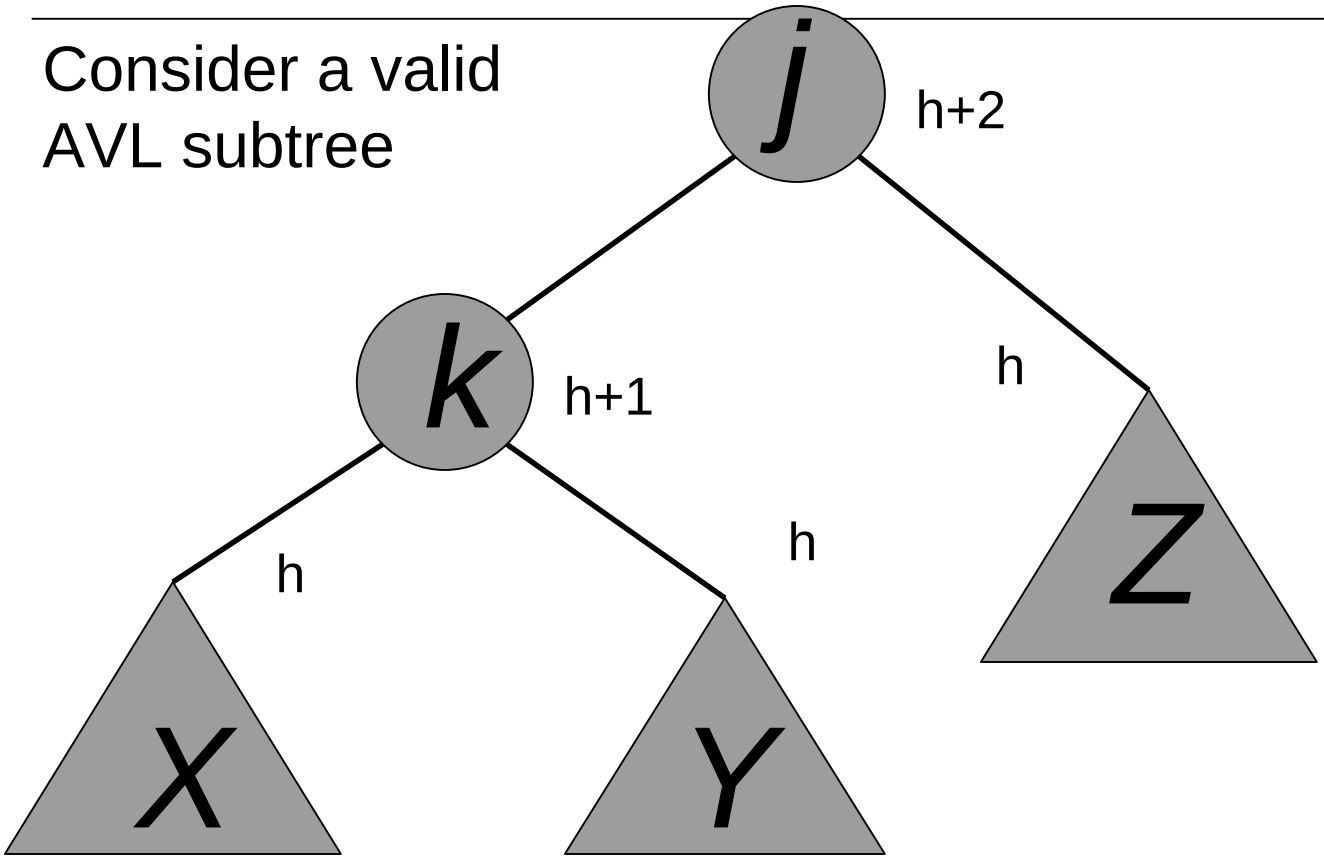


"Right rotation" done!
("Left rotation" is mirror symmetric)

AVL property has been restored!

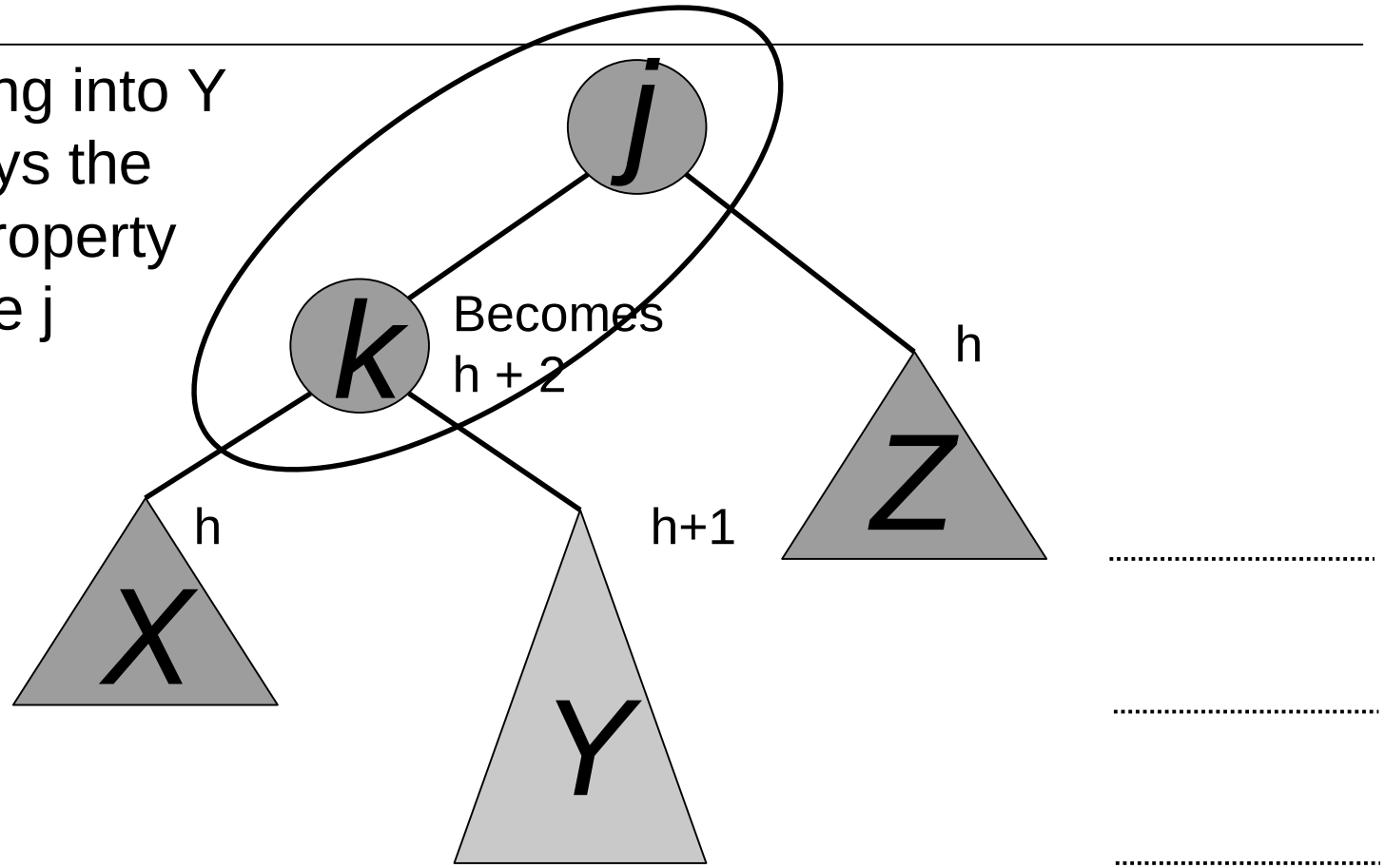
AVL Insertion: Inside Case

Consider a valid AVL subtree



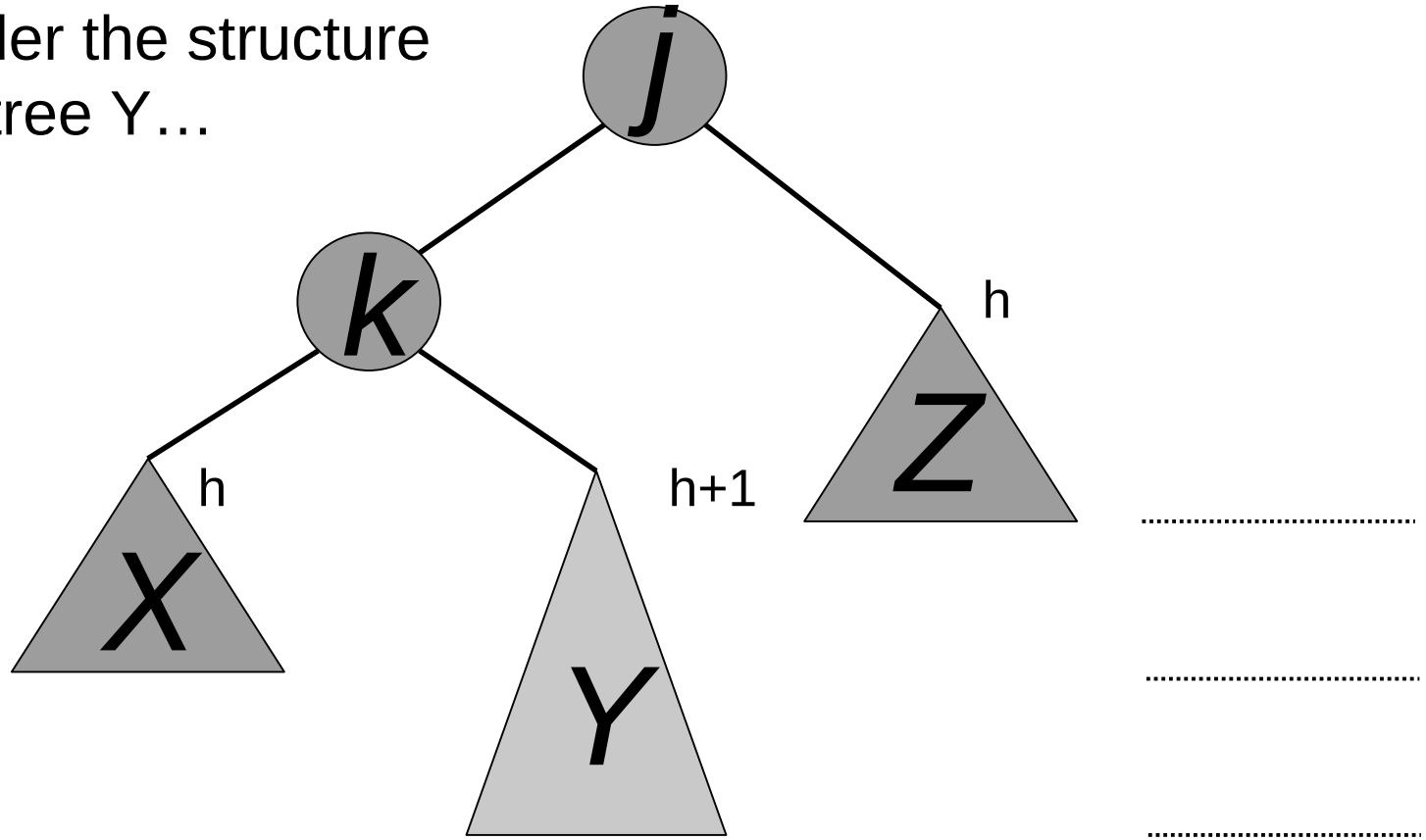
AVL Insertion: Inside Case

Inserting into Y destroys the AVL property at node j

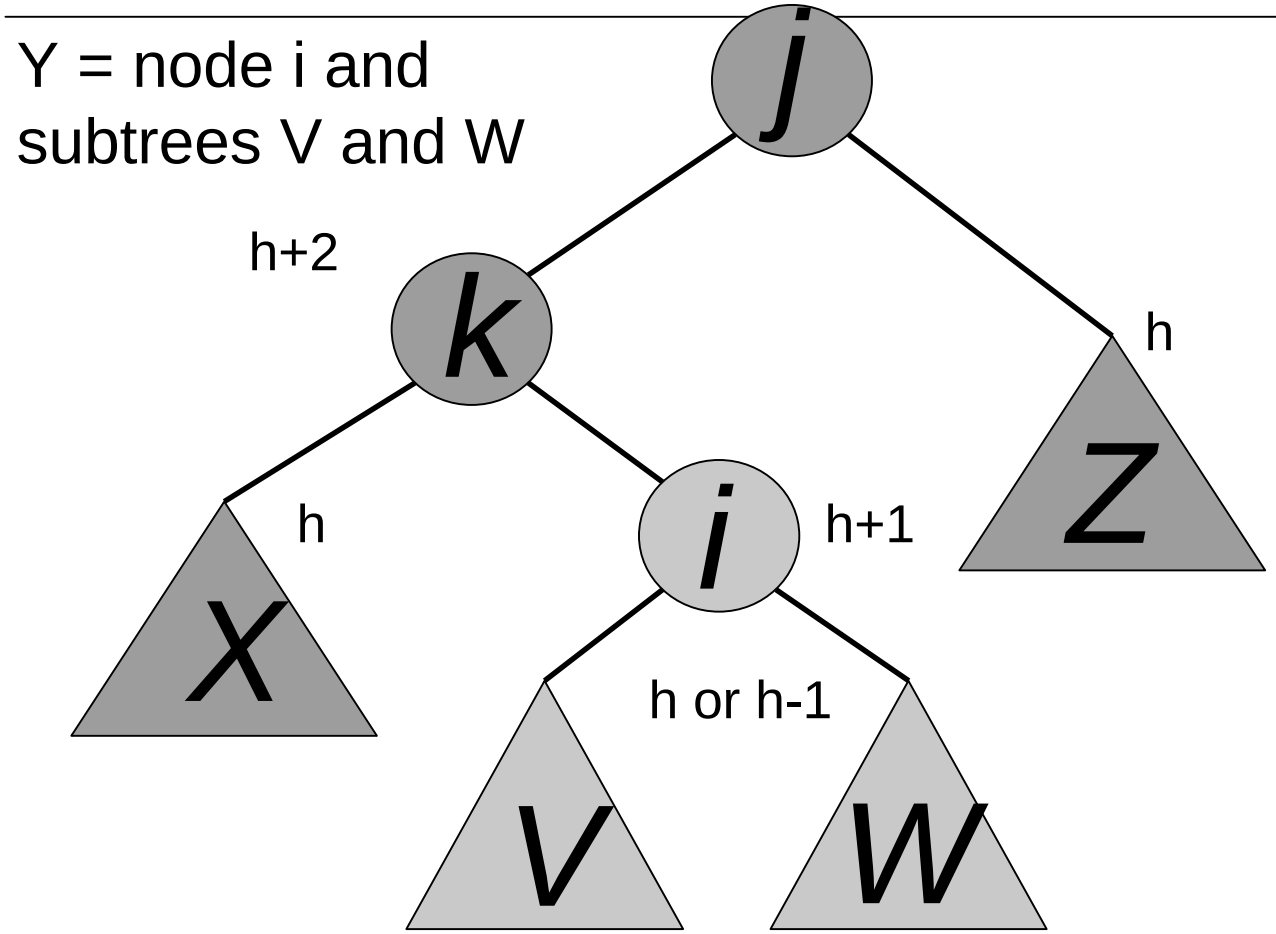


AVL Insertion: Inside Case

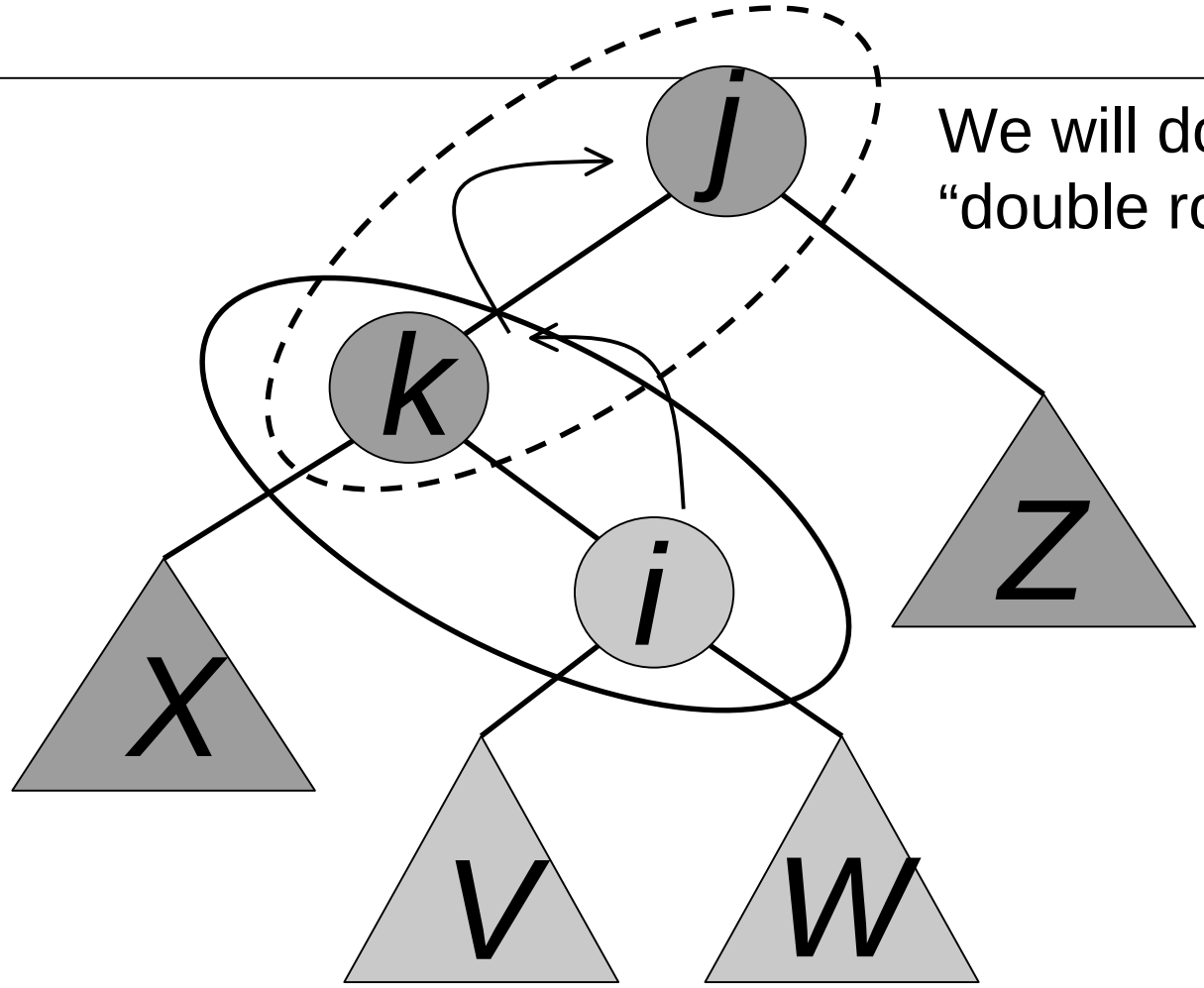
Consider the structure of subtree Y…



AVL Insertion: Inside Case



AVL Insertion: Inside Case

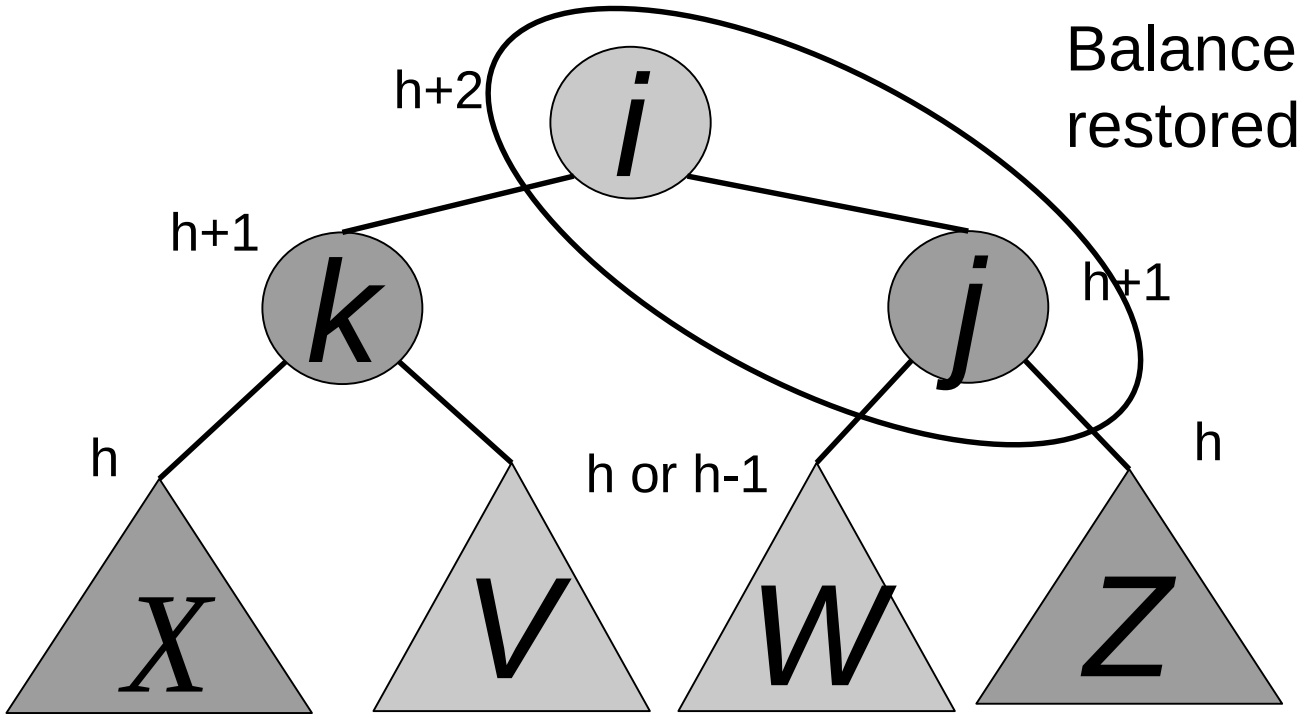


We will do a left-right "double rotation" . . .

Double rotation : second rotation

double rotation complete

Balance has been restored



Non-recursive insertion or the hacker's delight

- Key observations;
  - At most one rotation
  - Balance factor: 2 bits are sufficient (-1 left, 0 equal, +1 right)
  - There is one node on the path of insertion, say S, that is "critical". It is the node where a rotation can occur and nodes above it won't have their balance factors modified

Non-recursive insertion

- Step 1 (Insert and find S):
  - Find the place of insertion and identify the last node S on the path whose BF $\neq$ 0 (if all BF on the path = 0, S is the root).
  - Insert
- Step 2 (Adjust BF's)
  - Restart from the child of S on the path of insertion. (note: all the nodes from that node on on the path of insertion have BF = 0.)If the path traversed was left (right) set BF to –1 (+1) and repeat until you reach a null link (at the place of insertion)

Non-recursive insertion (ct'd)

- Step 3 (Balance if necessary):
  - If BF(S) = 0 (S was the root) set BF(S) to the direction of insertion (the tree has become higher)
  - If BF(S) = -1 (+1) and we traverse right (left) set BF(S) = 0 (the tree has become more balanced)
  - If BF(S) = -1 (+1) and we traverse left (right), the tree becomes unbalanced. Perform a single rotation or a double rotation depending on whether the path is left-left (right-right) or left-right (right-left)

Non-recursive Insertion with BF's