Insert at the leaf. The inserted node will have a height of 0.

Since inserting occurred by traversing from the root all the way to a leaf node, recursively, we have to go back until we end up back at the root. As we go back, rebalance will be called on the nodes along the path. This causes the height of the node to be updated since we just added a node. The balance factor is examined for the given node and if it is >1 or <-1, then rotation occurs.

When rotation occurs, the location of the node on which the rotation occurs will change. This means that this change has to be reflected back to the insert so that the right or the left members are updated.

Example: Inserts into an AVL tree

Insert (NULL, 9)

Insert (9,4)

Insert (9,10)

Insert (9,2)

Insert (9,3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ins  9 | |  | | --- | | Insert (NULL, 9) NULL  Return(9) //root | | 9 (0,0) //only one node. Rebalance is not called  Note: 1st number is height, 2nd number is balance. Only height is stored. | |
| ins  4 | |  | | --- | | Insert (9,4) 9  Left=insert(NULL, 4)  Node= rebalance (9)  Return 9 //root | | Insert (NULL, 4)  Create node  Return (4) |   Inserts at leaf node.  9 (0,0) //needs updating  4 (0,0) | |  | | --- | | rebalance (9) 9  H9=Max(H4,NULL)+1  Balance=0 - -1 (1)  Return 9 | | 9 (1,1) //updated  4 (0,0) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ins  10 | |  | | --- | | Insert (9,10) 9  right=insert(NULL, 10)  Node= rebalance (9)  Return 9 //root | | insert (NULL,10)  Create node  return (10) |   Inserts at leaf node.  9 (1,0) //needs updating  4 (0,0) 10 (0,0) | |  | | --- | | rebalance (9) 9  H9=Max(H4,H10)+1  Balance=0-0  Return 9 | | 9 (1,0) //stays the same  4(0,0) 10 (0,0) |
| ins  2 | |  | | --- | | Insert (9,2) 9  left=insert(4, 2)  Node= rebalance (9)  Return 9 //root | | Insert (4, 2) 4  Left=insert (NULL,2)  Node= rebalance (4)  Return 4 | | insert (NULL,2)  create node  return (2) |   9(1,0) //needs update  4(0,0) 10(0,0) //needs update  2(0,0) | |  | | --- | | rebalance (9) 9  H9=max(H4,H10)+1  Balance=1-0  Return 9 |  |  | | --- | | rebalance (4) 4  H4=Max(H2,NULL)+1  Balance=0 - -1  Return 4 | | ================================  9(2,1) //update 9  4(1,1) 10(0,0)  2(0,0)  ==============================  9(1,0)  4(1,1) 10(0,0) //update 4  2(0,0) |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ins  3 | |  | | --- | | 1  Insert (9,3) 9  left=insert(4, 3)  Node= rebalance (9)  Return 9 //root | | 2  Insert (4, 3) 3  Left=insert (2,3)  Node= rebalance (4)  Return Node (3)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  NOTE what is being returned | | 3  insert (2,3) 2  right=insert(NULL, 3)  Node= rebalance (2)  Return (2) | | 4  Insert (NULL, 3)  Create node  Return 3 |   This is what happens after 3 has been inserted. The height is automatically set to 0.  9(2,1)  4(1,1) 10 (0,0)  2(0,0)  3(0,0) | |  | | --- | | 7  rebalance (9) 9  H9=max(H3,H10)+1  Balance=2-1  Return 9 | | 6  rebalance (4) 4  H4=max(H2,NULL)+1  Balance=1- -1 (2)  Left= RotLeft(2) 3  Return RotRightt(4) 3 | | 5  rebalance (2) 2  H2=max(NULL,H3)+1  Balance=-1-0  Return 2 |   The heights are updated for 2, 4, 9. The balances are checked. If it is >1 or <-1 then rotation is needed. | ==================    9 (2,1) //update  3(1,0) 10 (0,0)  2(0,0) 4(0,0)  ================  9  4(2,2) 10  2(1,-1)  3(0,0)  =================  9  4 10  2(1, -1) //updated  3(0,0) | double rotation  Rotate right on 2  Rotate left on 4  Given the following: first rotate left on 2  9  4(2,2) 10  2(1,-1)  3(0,0)  The result of left rotation is  9  4(2,2) 10  3(1,1)  2(0,0)  Now we need to rotate right on 4  =======================  Rotate right on 4  9  3(1,0) 10 (0,0)  2(0,0) 4(0,0) |

|  |  |  |
| --- | --- | --- |
| Rotate left  2 (x)  3 (Y)  NULL(T2)  Rotate right    4(Y)  3(x)  2 NULL(T2) | 3  2  NULL  3  2 4 | node \*y = x->right;  node \*T2 = y->left;  // Perform rotation  y->left = x;  x->right = T2;  node \*x = y->left;  node \*T2 = x->right;  // Perform rotation  x->right = y;  y->left = T2; |