

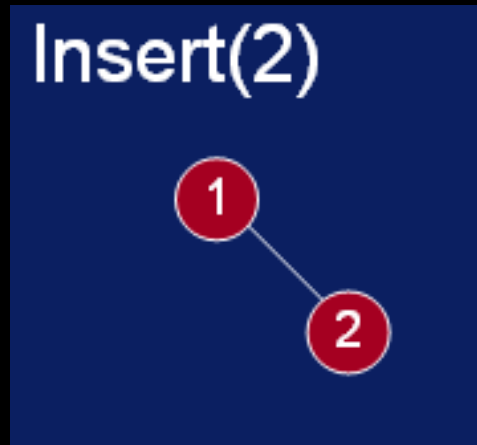
Lecture # 13

AVL Tree Building Example

Insert(1)

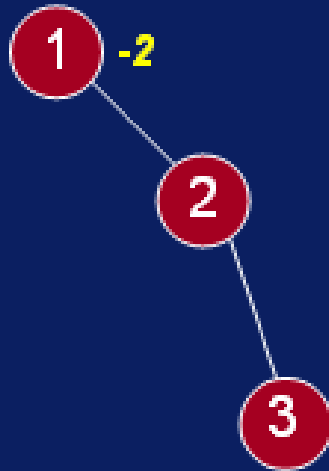
1

AVL Tree Building Example



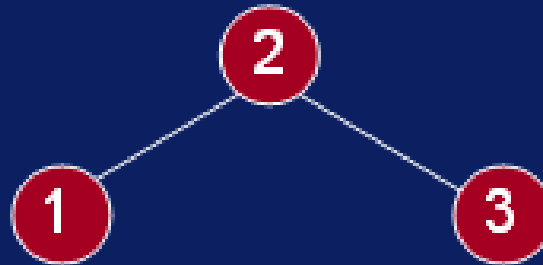
AVL Tree Building Example

Insert(3) single left rotation



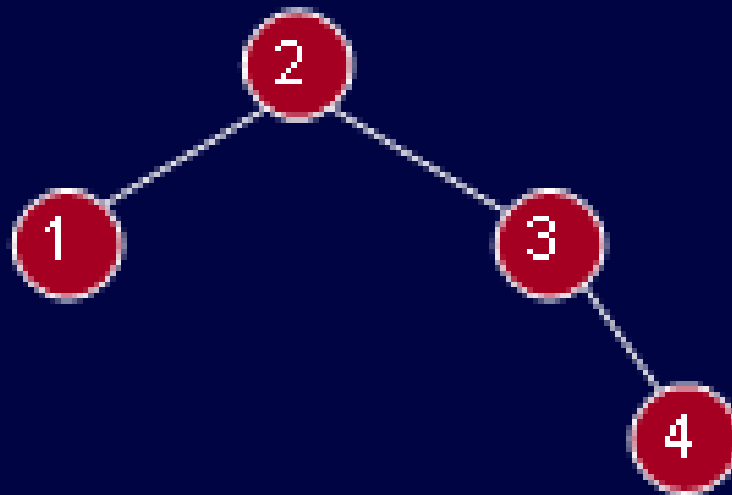
AVL Tree Building Example

Insert(3)



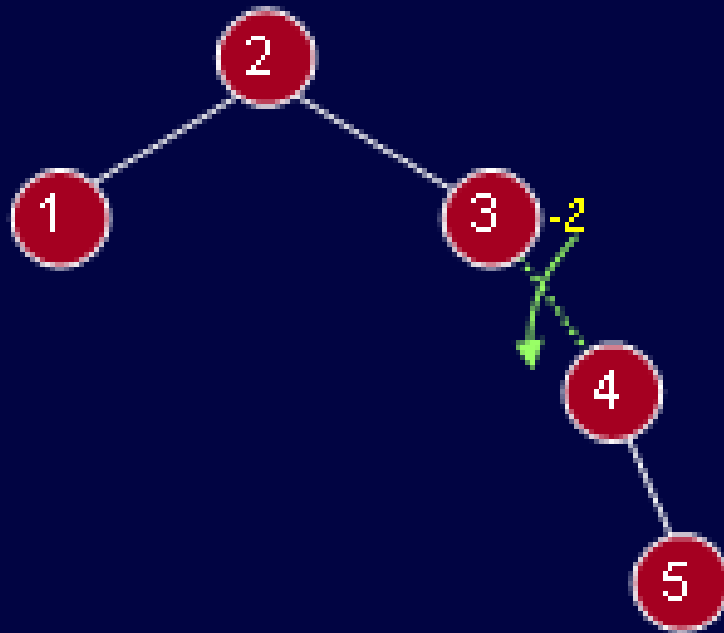
AVL Tree Building Example

Insert(4)



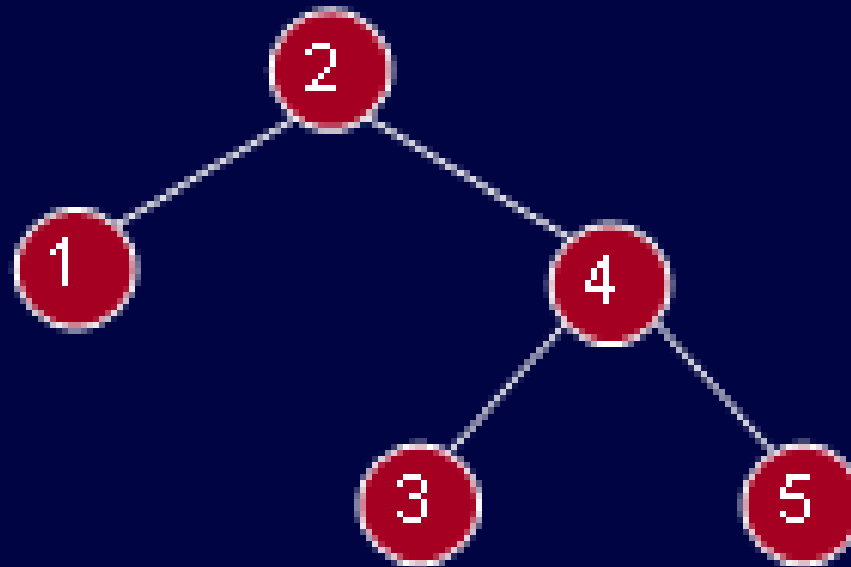
AVL Tree Building Example

Insert(5)



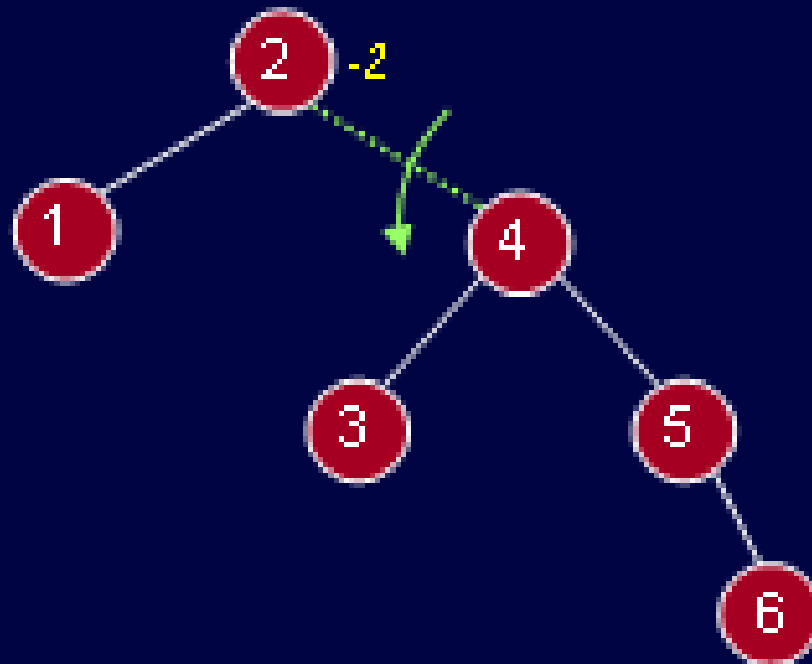
AVL Tree Building Example

Insert(5)



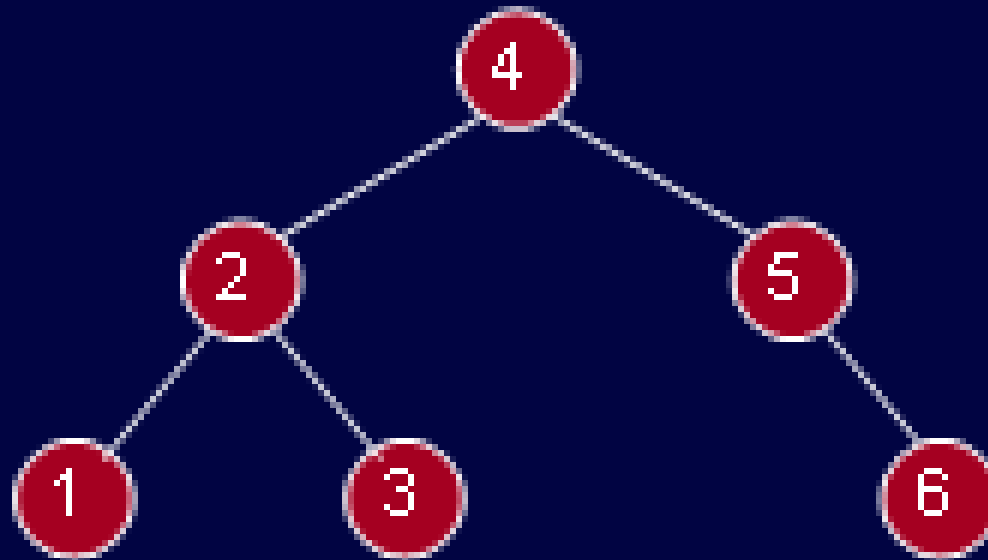
AVL Tree Building Example

Insert(6)



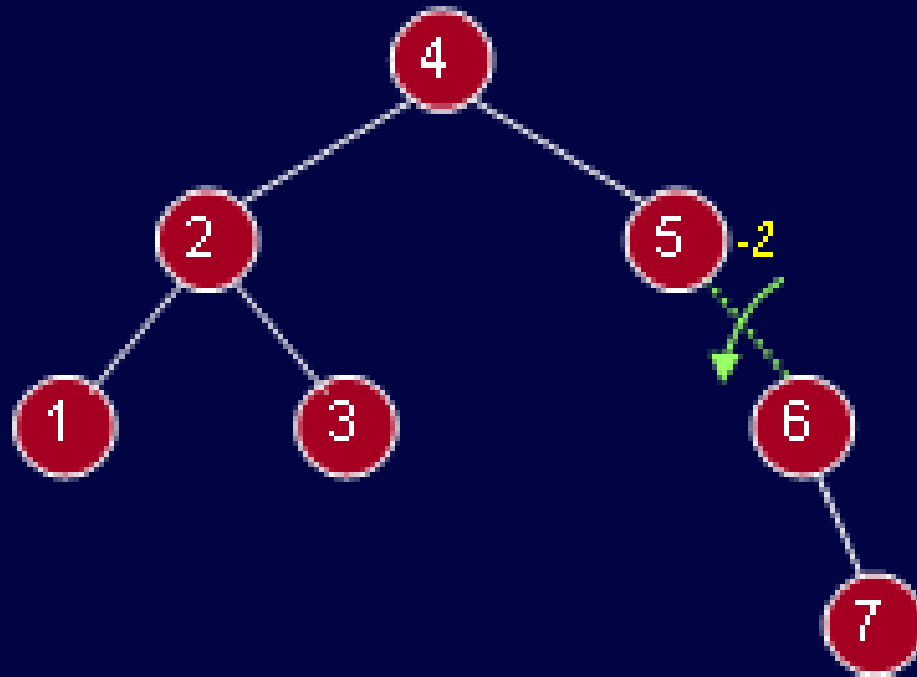
AVL Tree Building Example

Insert(6)



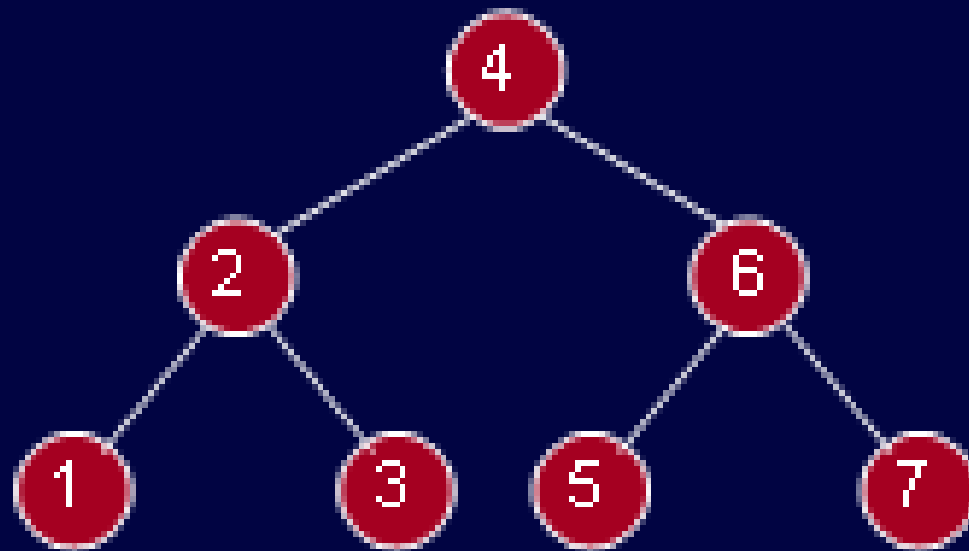
AVL Tree Building Example

Insert(7)



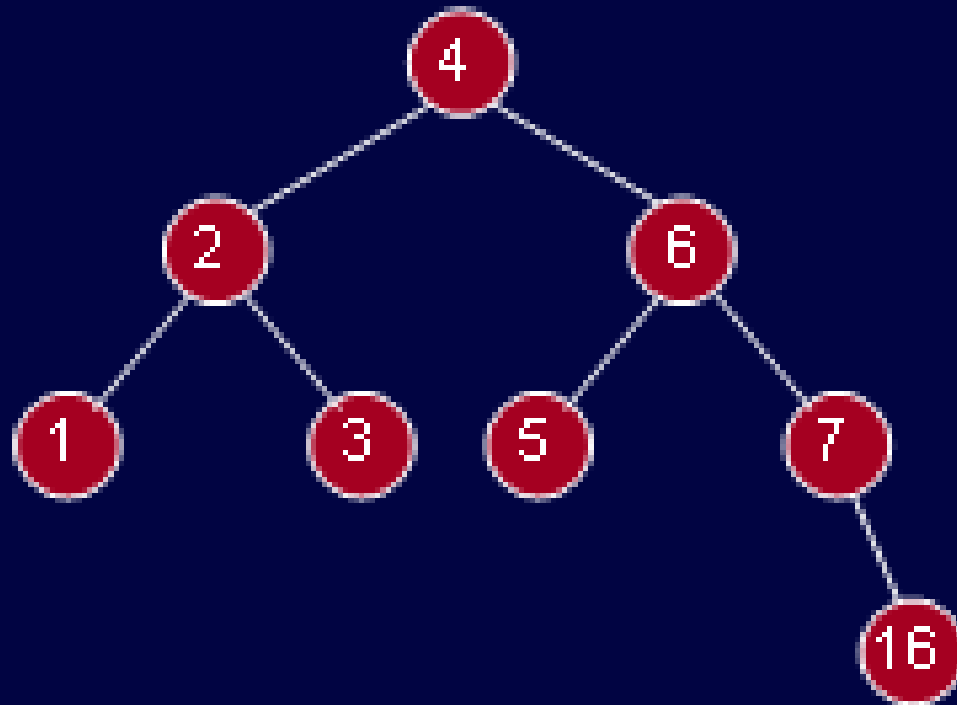
AVL Tree Building Example

Insert(7)



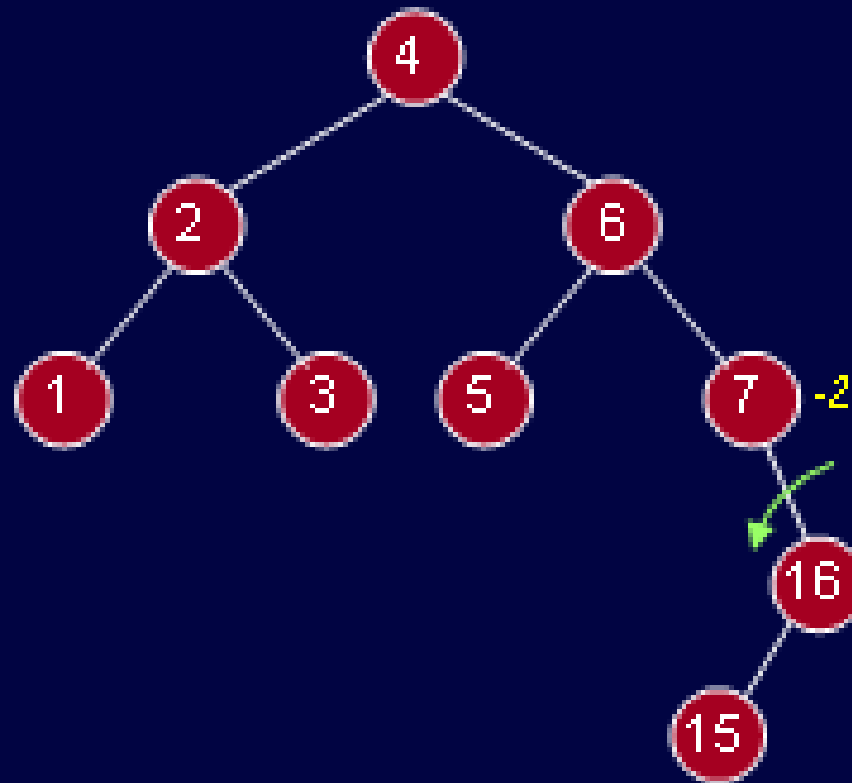
AVL Tree Building Example

Insert(16)



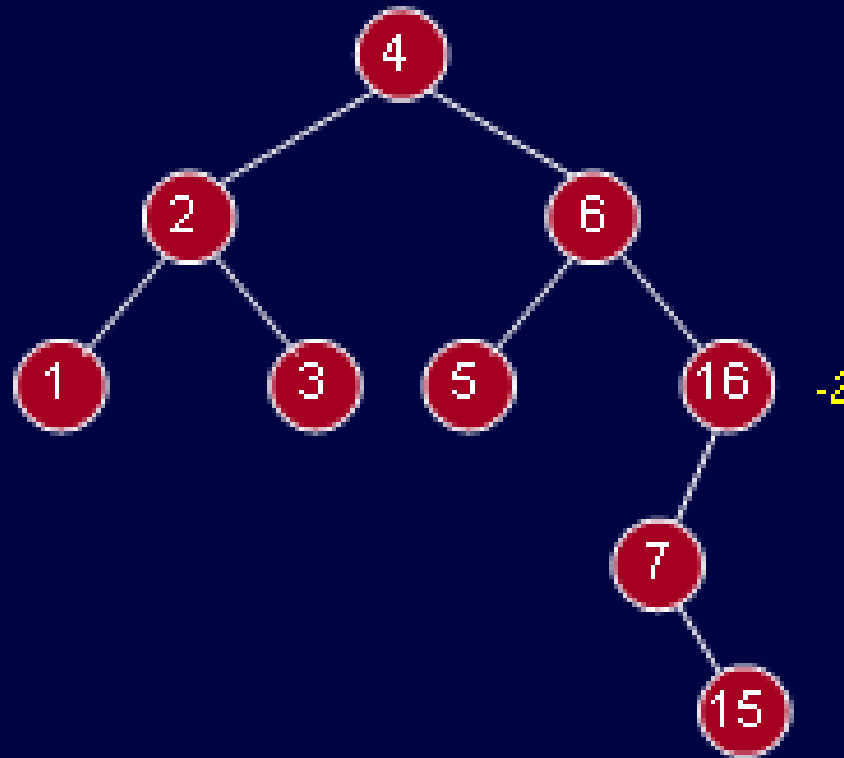
AVL Tree Building Example

Insert(15)



AVL Tree Building Example

Insert(15)



Cases for Rotation

- Single rotation does not seem to restore the balance.
- The problem is the node 15 is in an inner subtree that is too deep.
- Let us revisit the rotations.

Cases for Rotation

- Let us call the node that must be rebalanced \check{N} .
- Since any node has at most two children, and a height imbalance requires that \check{N} 's two subtrees differ by two (or -2), the violation will occur in four cases:

Cases for Rotation

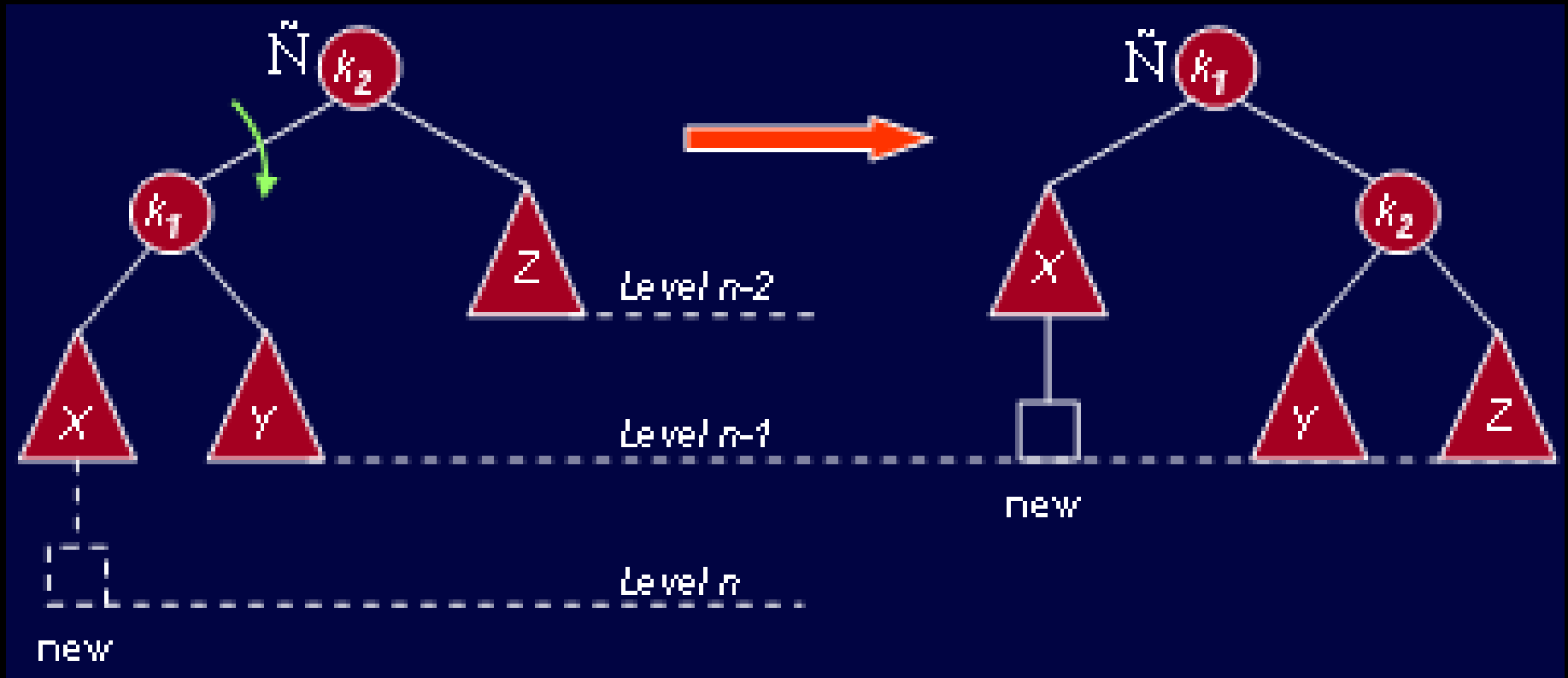
- An insertion into left subtree of the left child of \check{N} .
- An insertion into right subtree of the left child of \check{N} .
- An insertion into left subtree of the right child of \check{N} .
- An insertion into right subtree of the right child of \check{N} .

Cases for Rotation

- The insertion occurs on the “outside” (i.e., left-left or right-right) in cases 1 and 4
- Single rotation can fix the balance in cases 1 and 4.
- Insertion occurs on the “inside” in cases 2 and 3 which single rotation cannot fix.

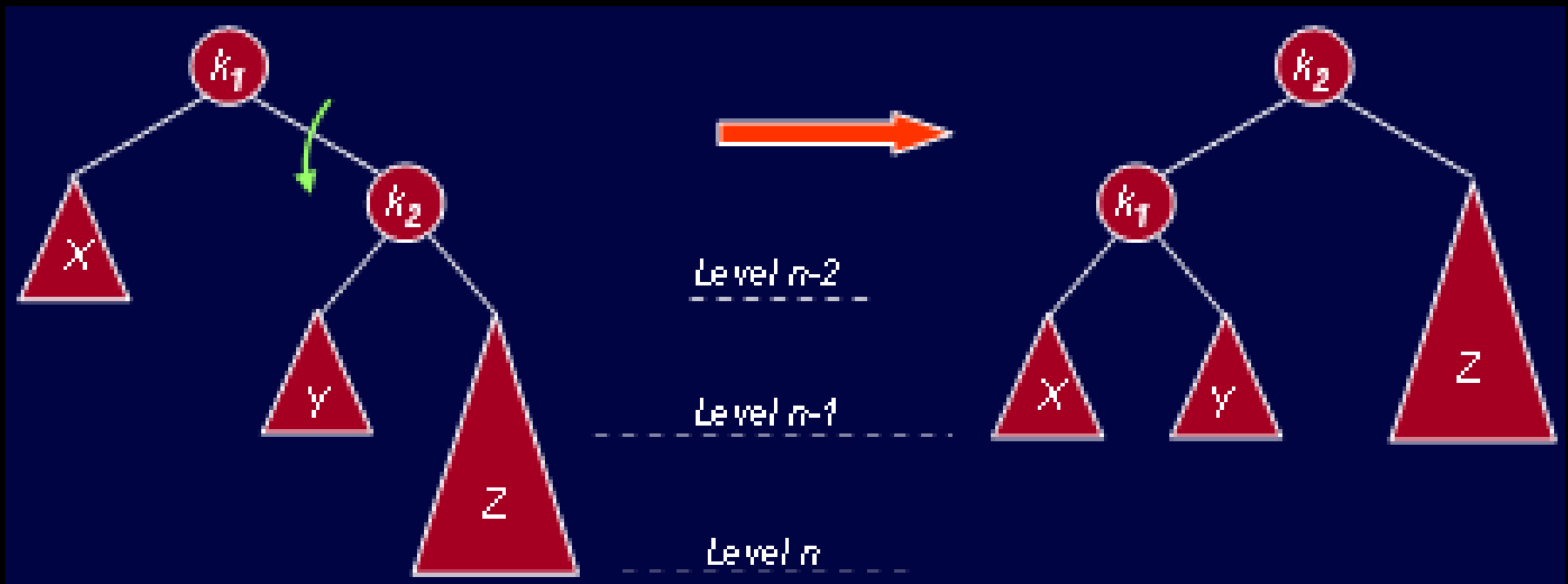
Cases for Rotation

- Single right rotation to fix case 1.



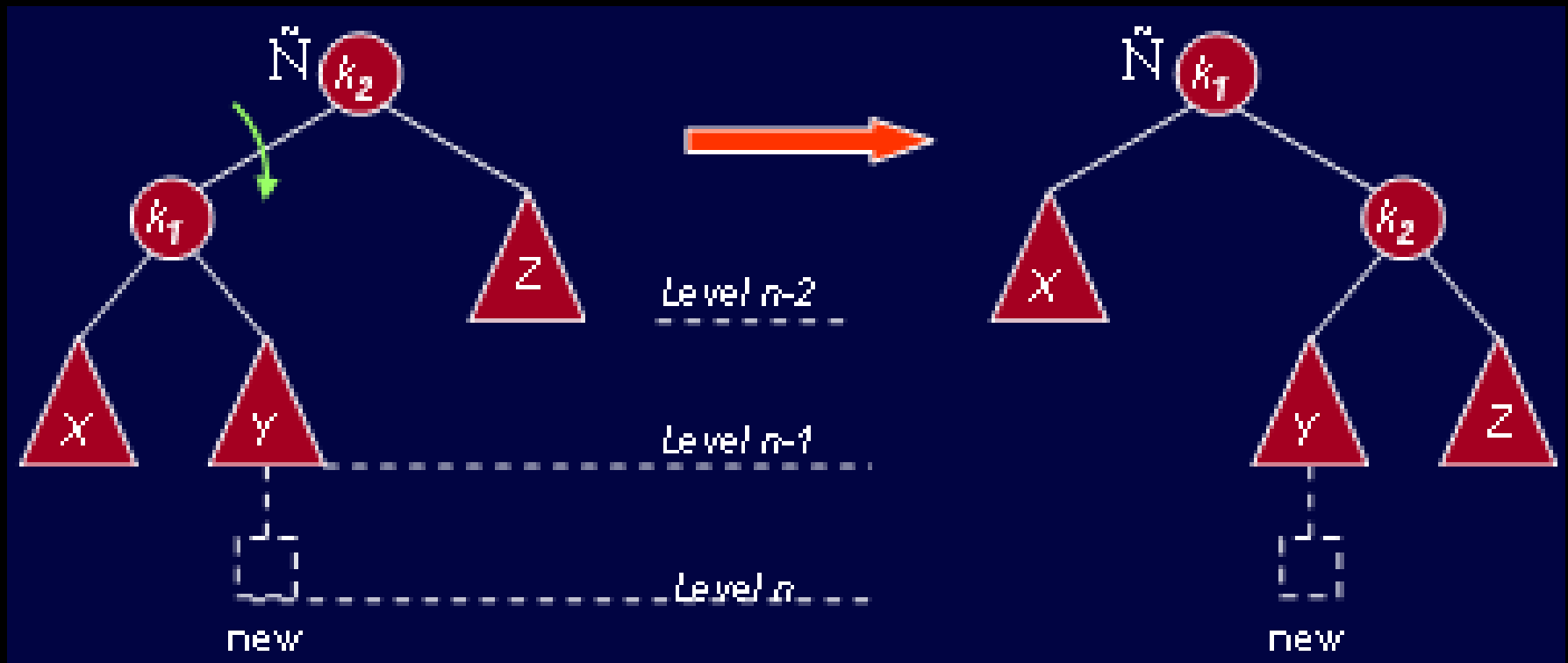
Cases for Rotation

- Single left rotation to fix case 4.

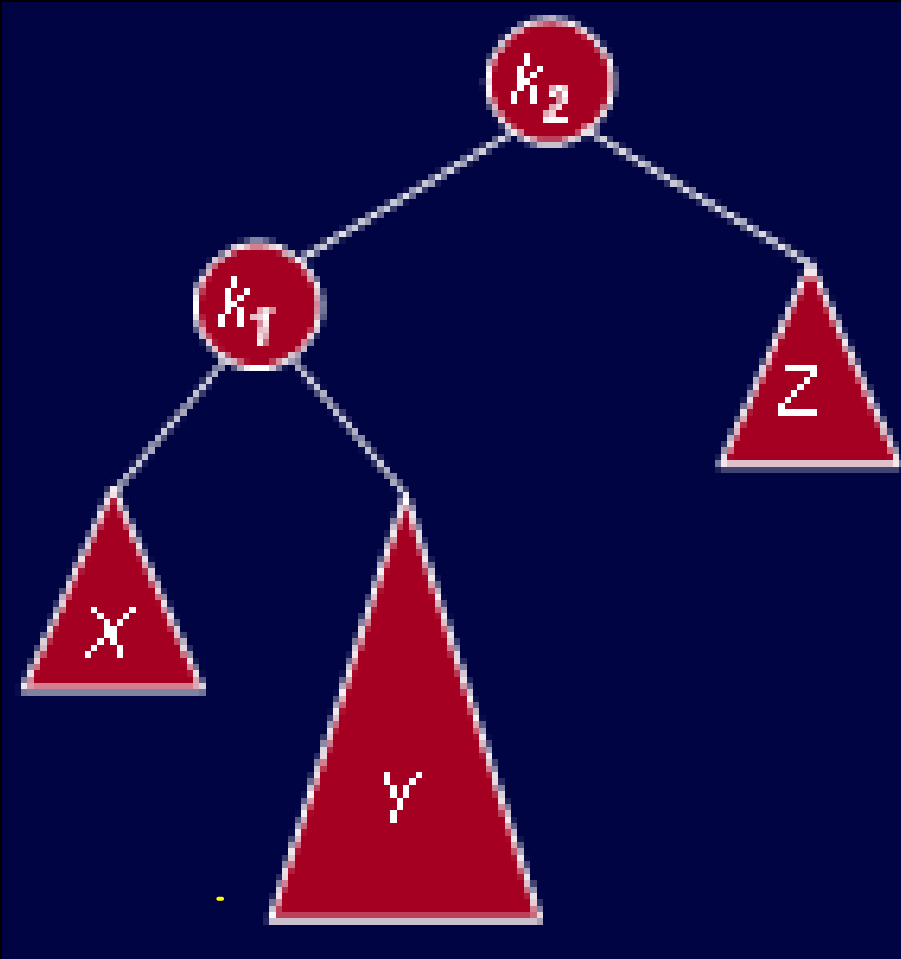


Cases for Rotation

- Single right rotation *fails* to fix **case 2**.

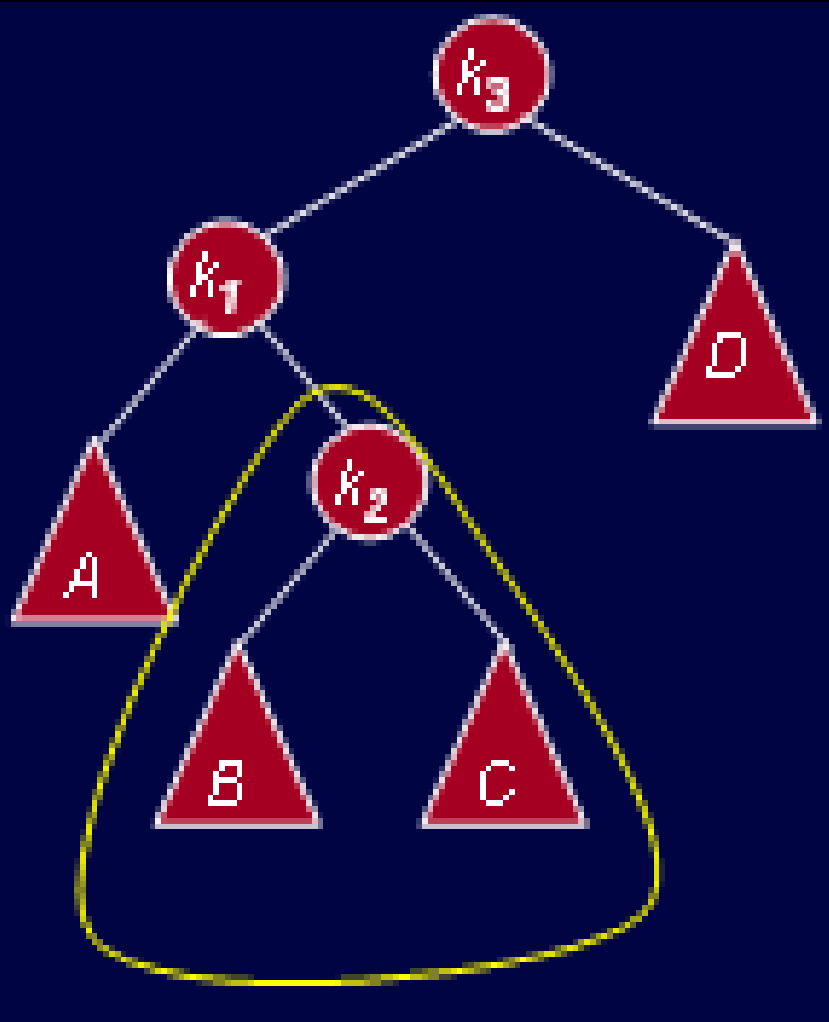


Cases for Rotation



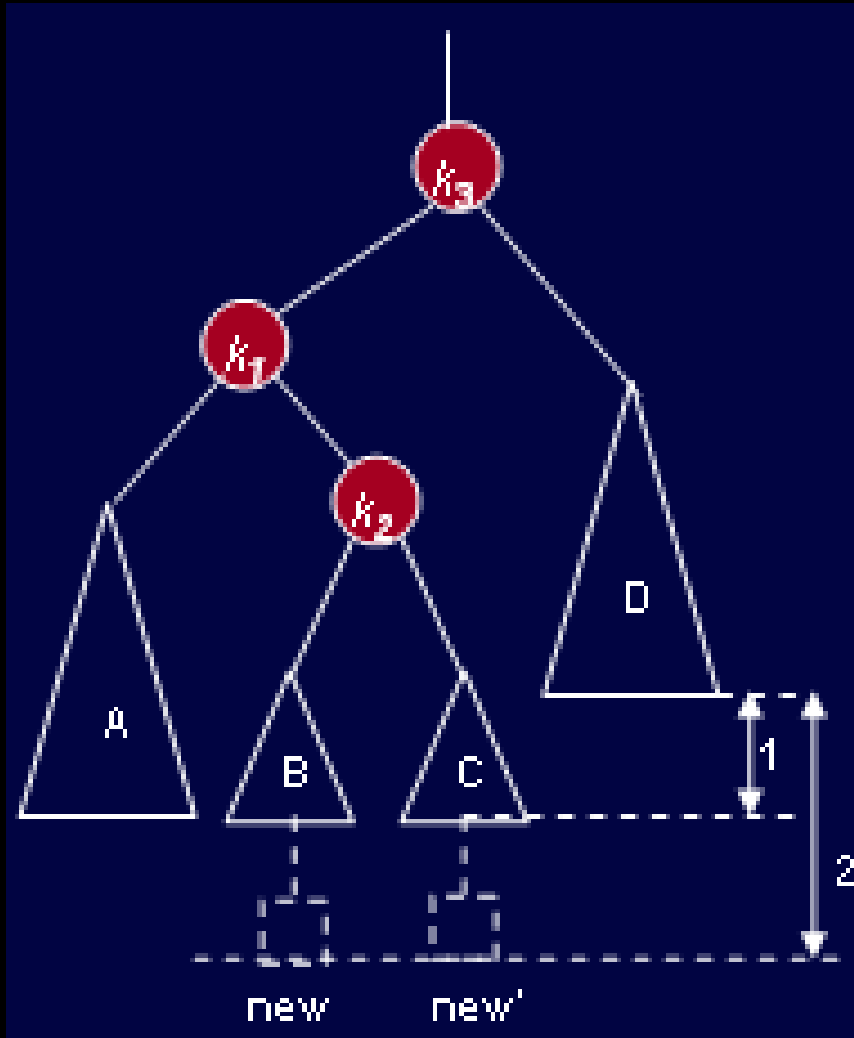
- Y is non-empty because the new node was inserted in Y .
- Thus Y has a root and two subtrees.
- View the entire tree with four subtrees connected with 3 nodes.

Cases for Rotation



- Y is non-empty because the new node was inserted in Y.
- Thus Y has a root and two subtrees.
- View the entire tree with four subtrees connected with 3 nodes.

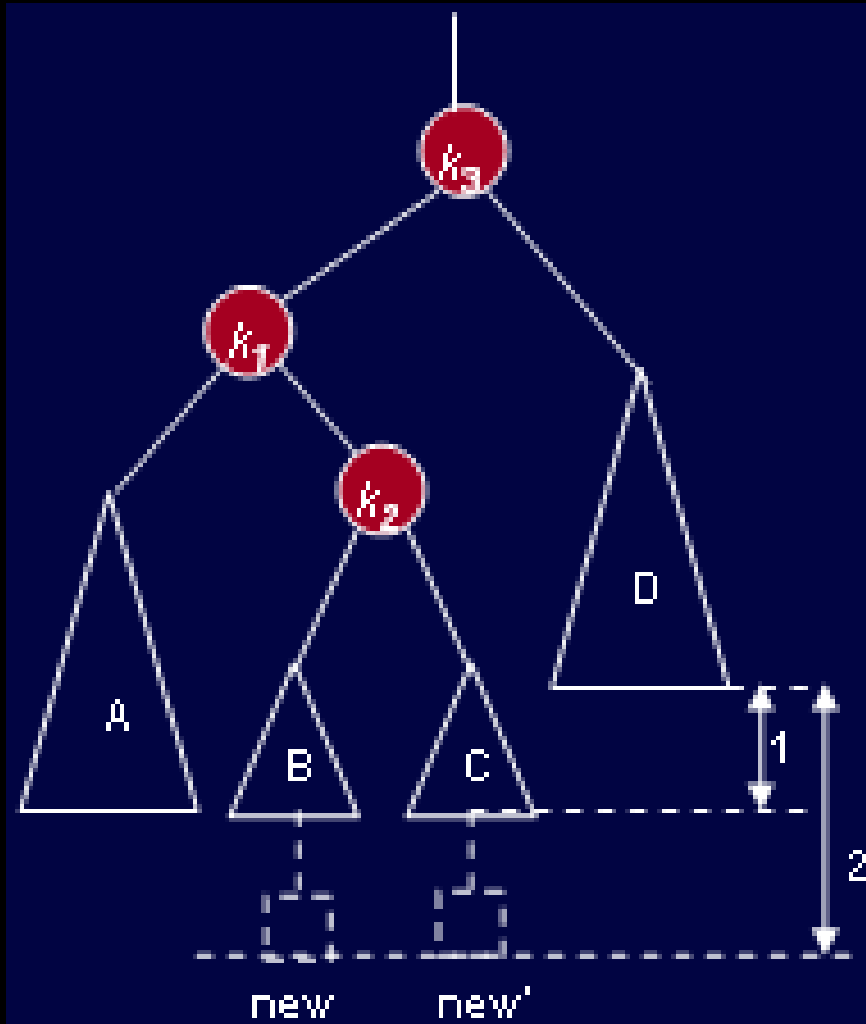
Cases for Rotation



- Exactly one of tree B or C is two levels deeper than D ; we are not sure which one.
- Good thing: it does not matter where it is added.
- To rebalance, k_3 cannot be left as the root.

New node inserted at either of the two spots

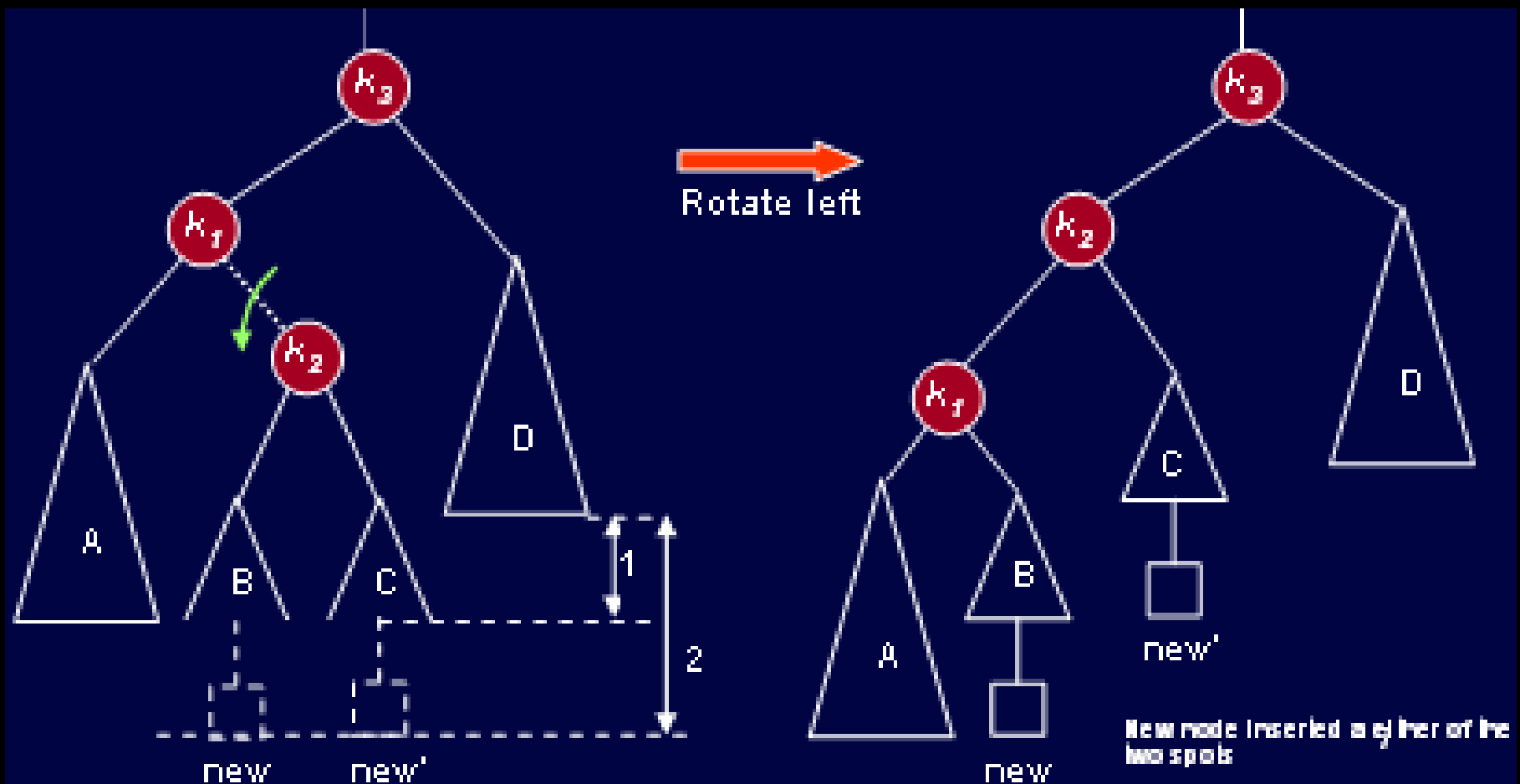
Cases for Rotation



- A rotation between k_3 and k_1 (k_3 was k_2 then) was shown to not work.
- The only alternative is to place k_2 as the new root.
- This forces k_1 to be k_2 's left child and k_3 to be its right child.

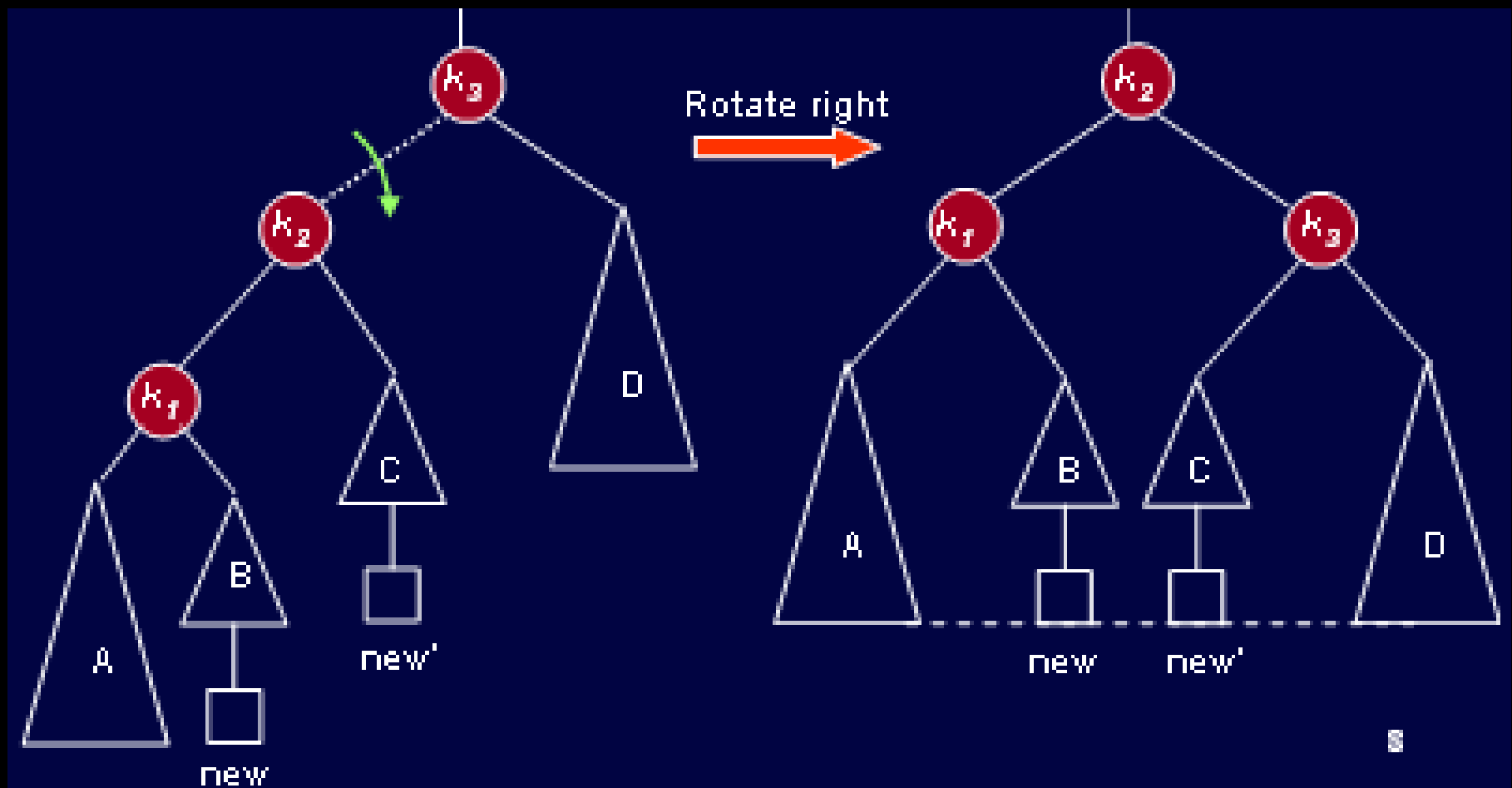
Cases for Rotation

- Left-right *double* rotation to fix case 2.



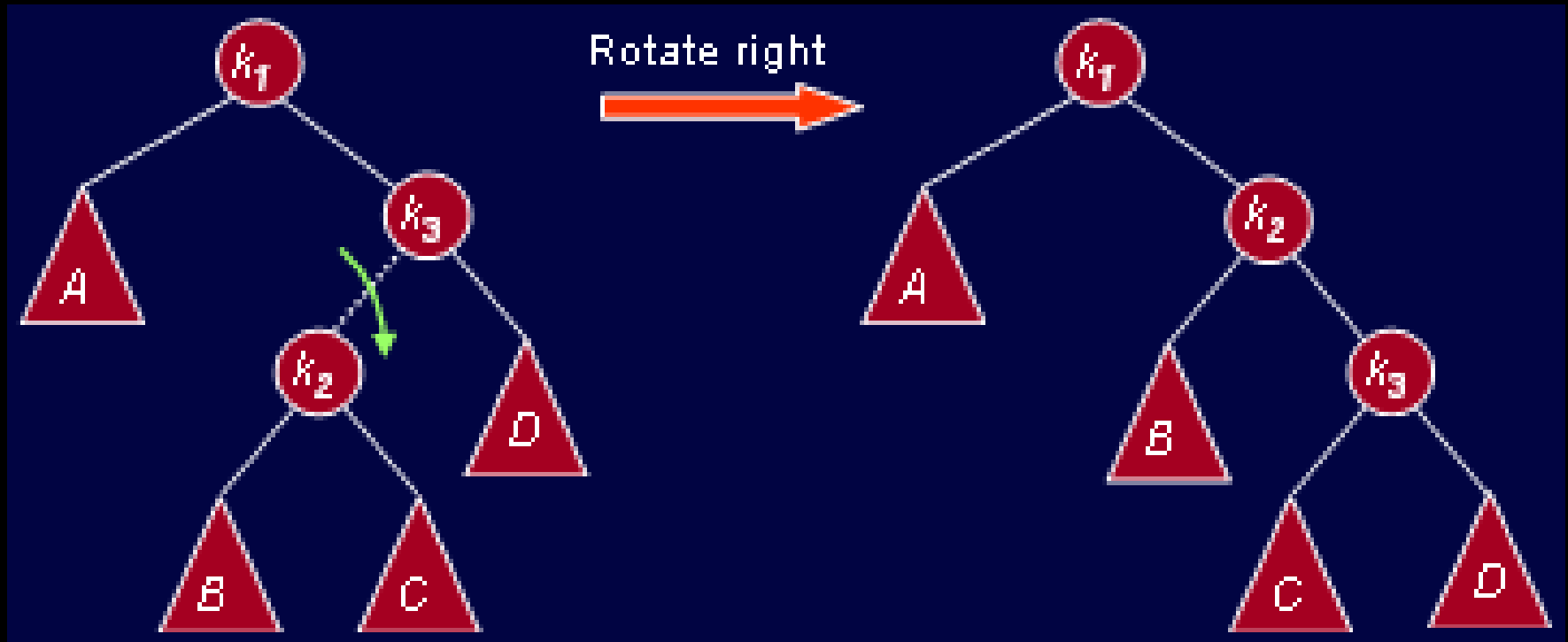
Cases for Rotation

- Left-right *double* rotation to fix case 2.



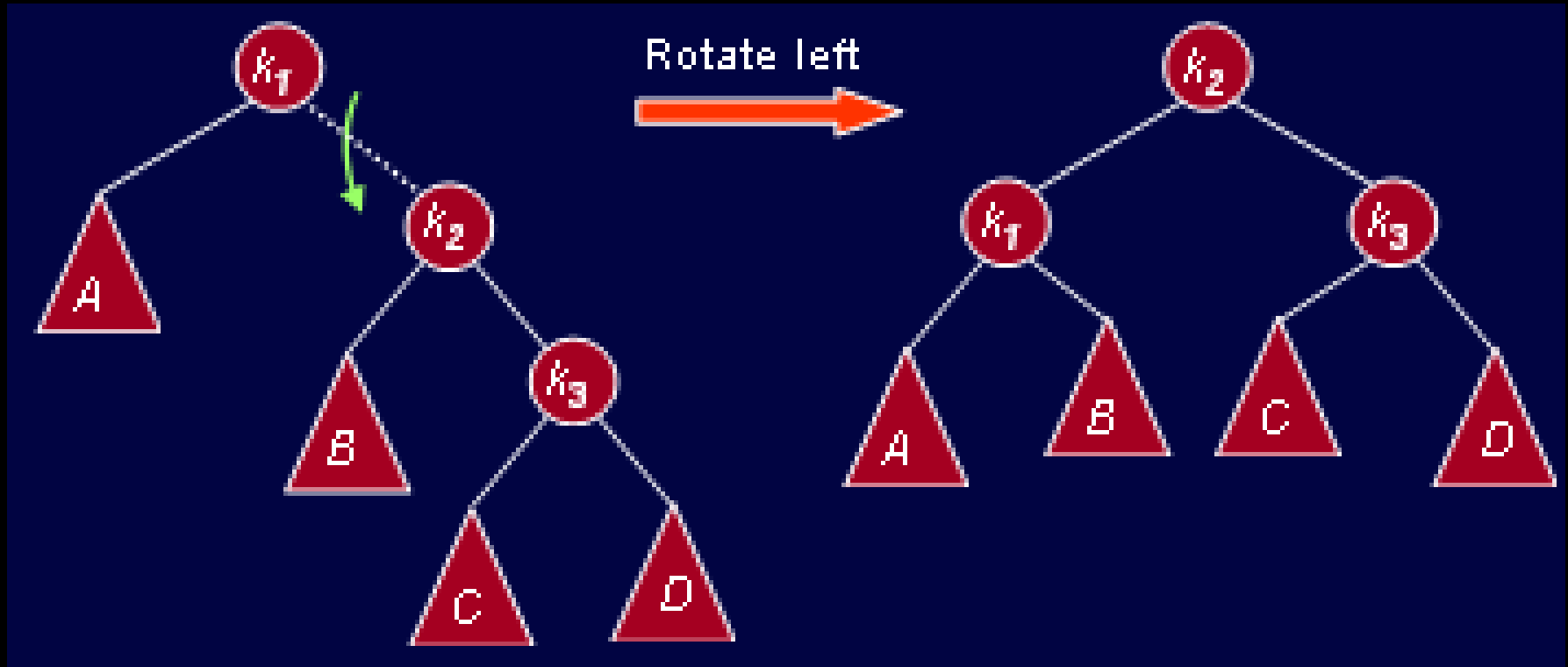
Cases for Rotation

- Right-left *double* rotation to fix case 3 .



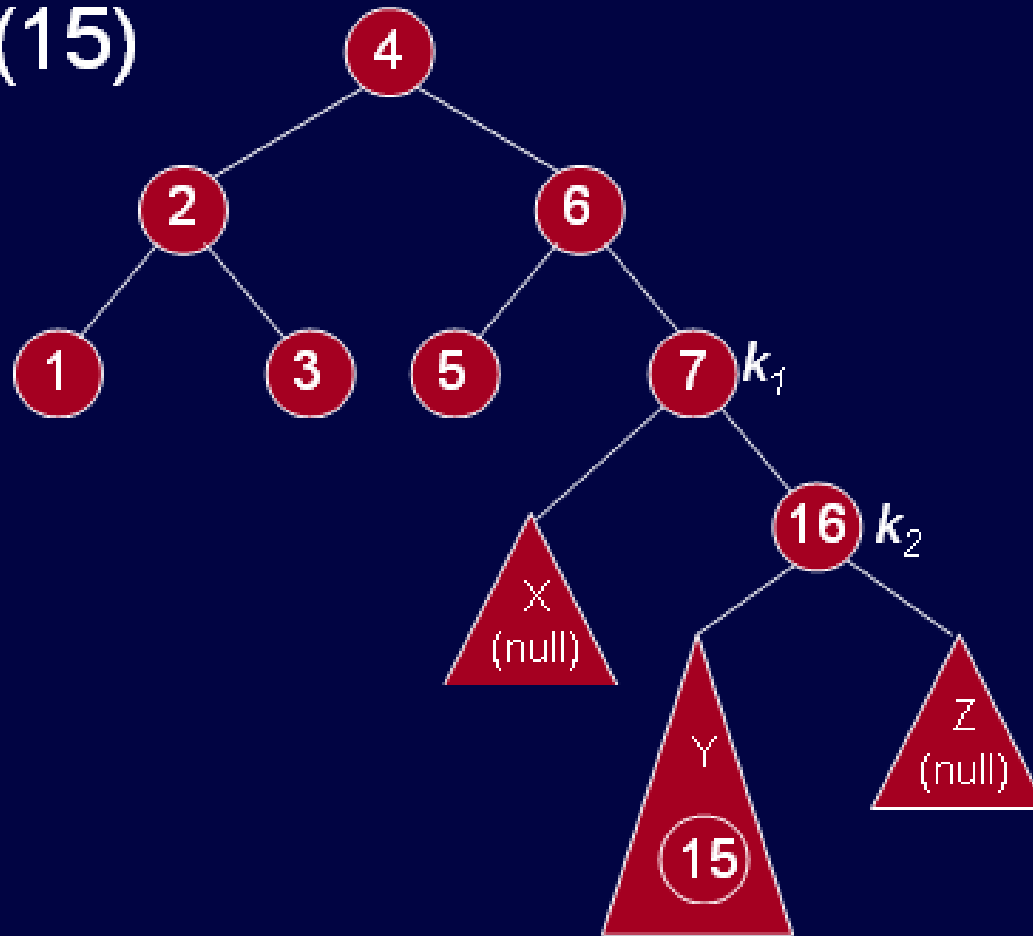
Cases for Rotation

- Right-left *double* rotation to fix case 3 .



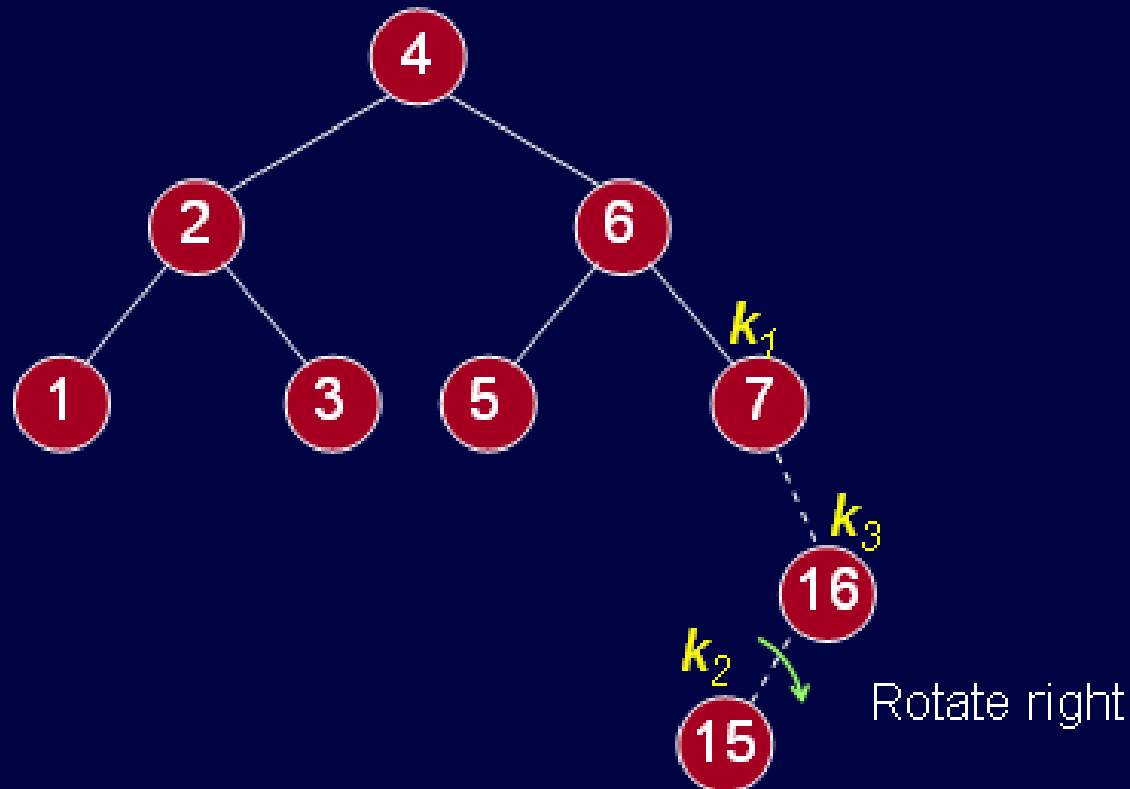
AVL Tree Building Example

Insert(15)



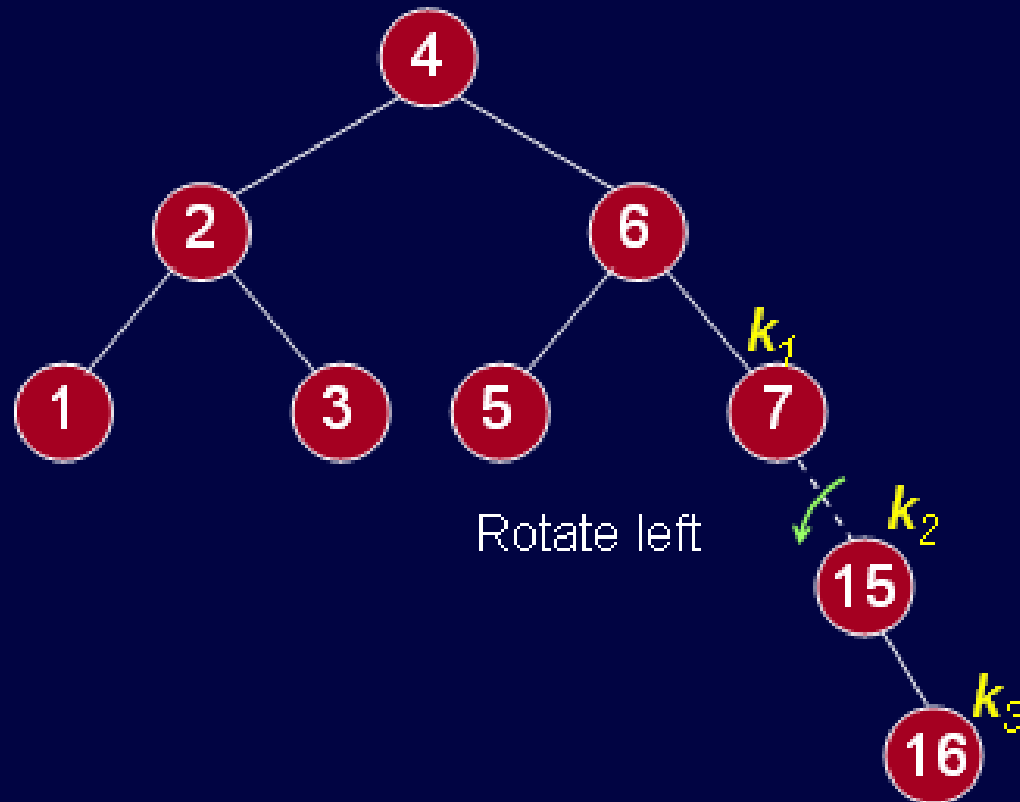
AVL Tree Building Example

Insert(15) right-left double rotation



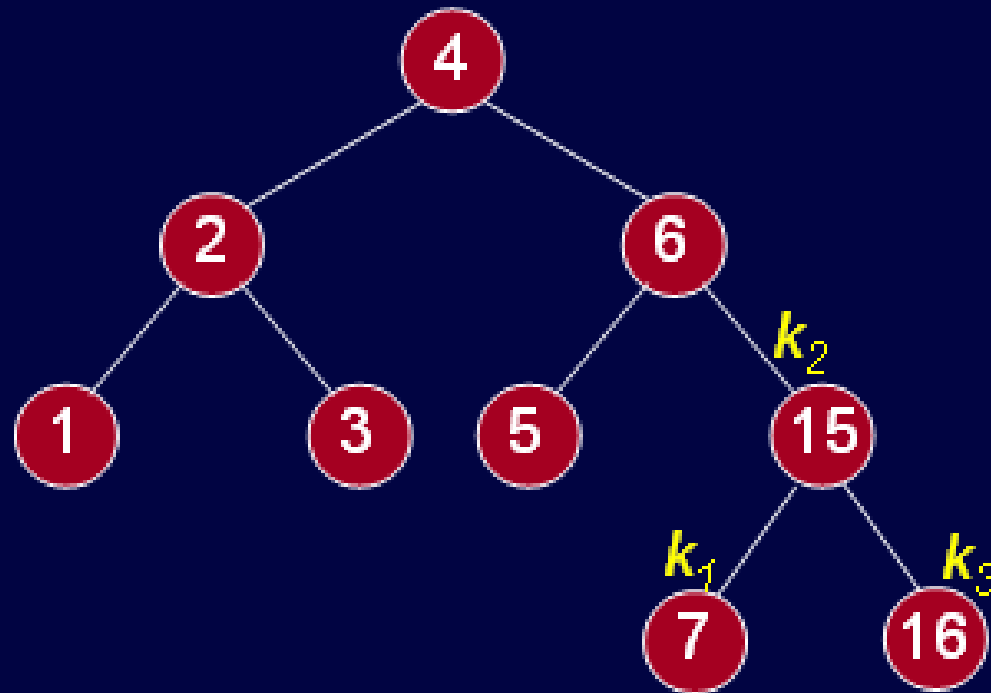
AVL Tree Building Example

Insert(15) right-left double rotation



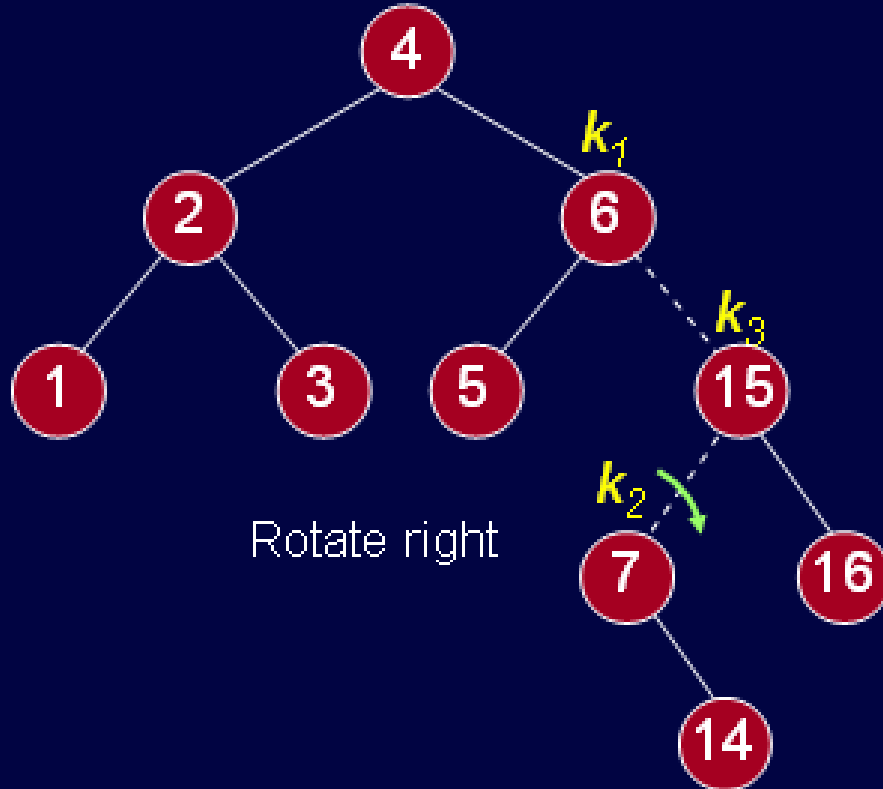
AVL Tree Building Example

Insert(15) right-left double rotation



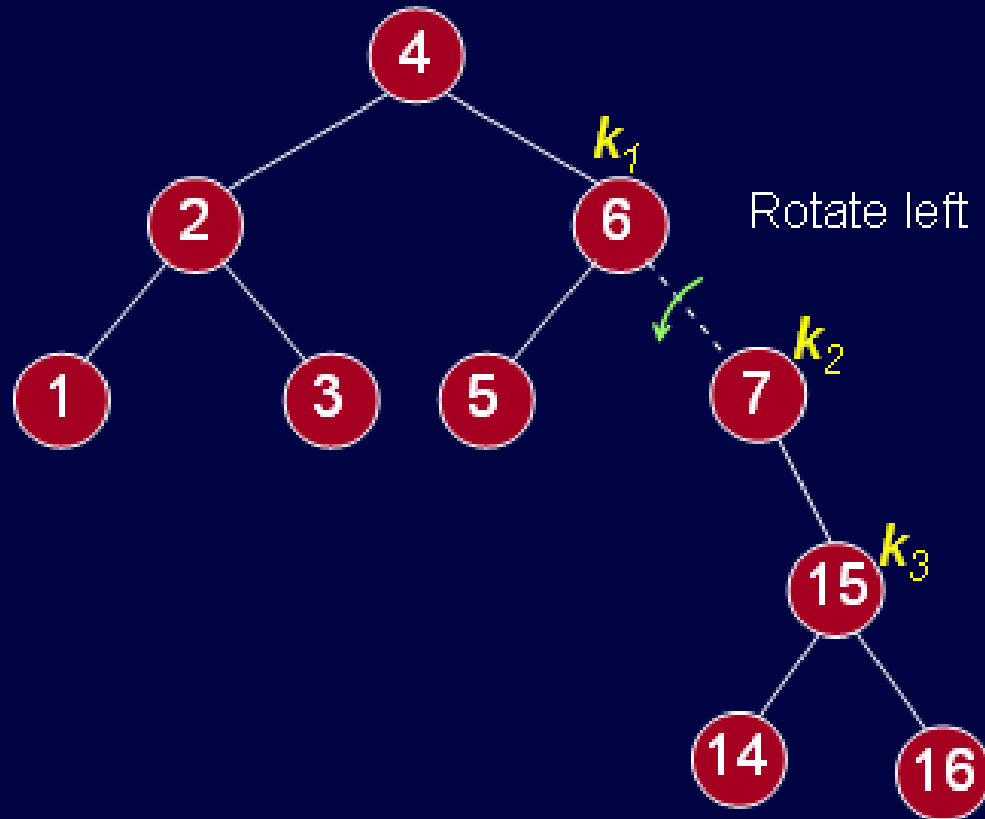
AVL Tree Building Example

Insert(14): right-left double rotation



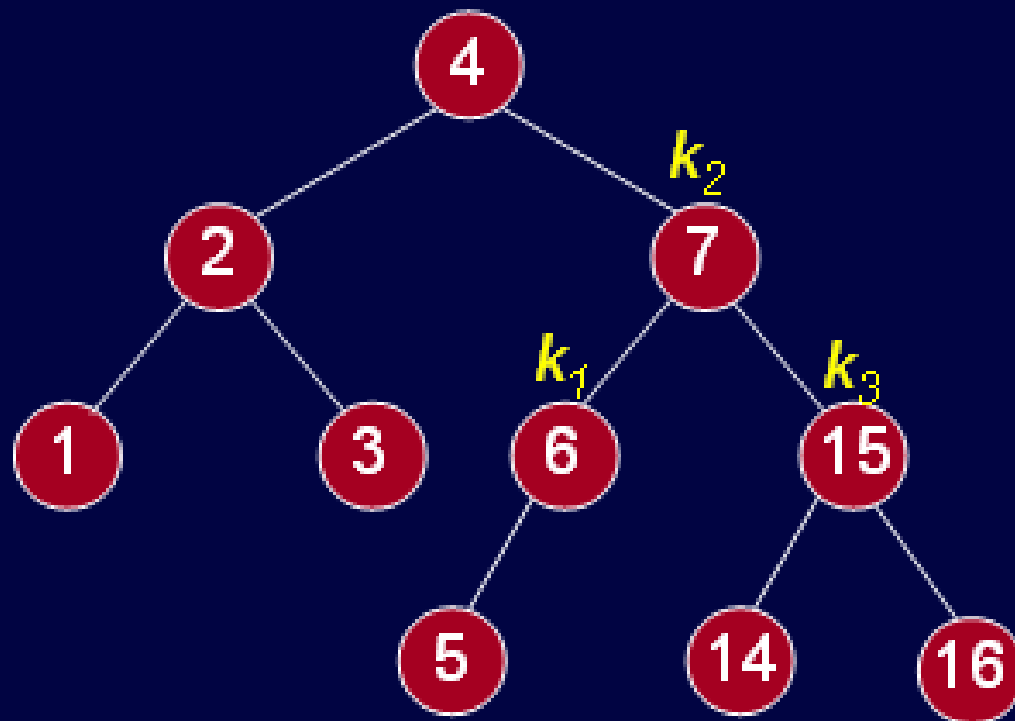
AVL Tree Building Example

Insert(14): right-left double rotation



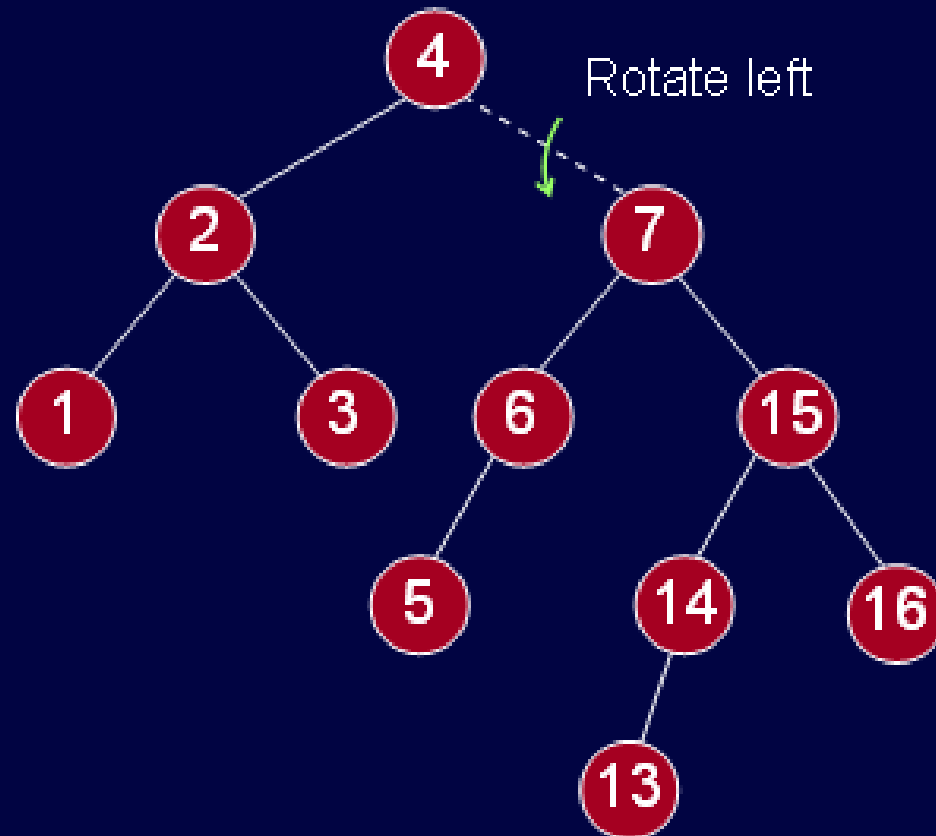
AVL Tree Building Example

Insert(14): right-left double rotation



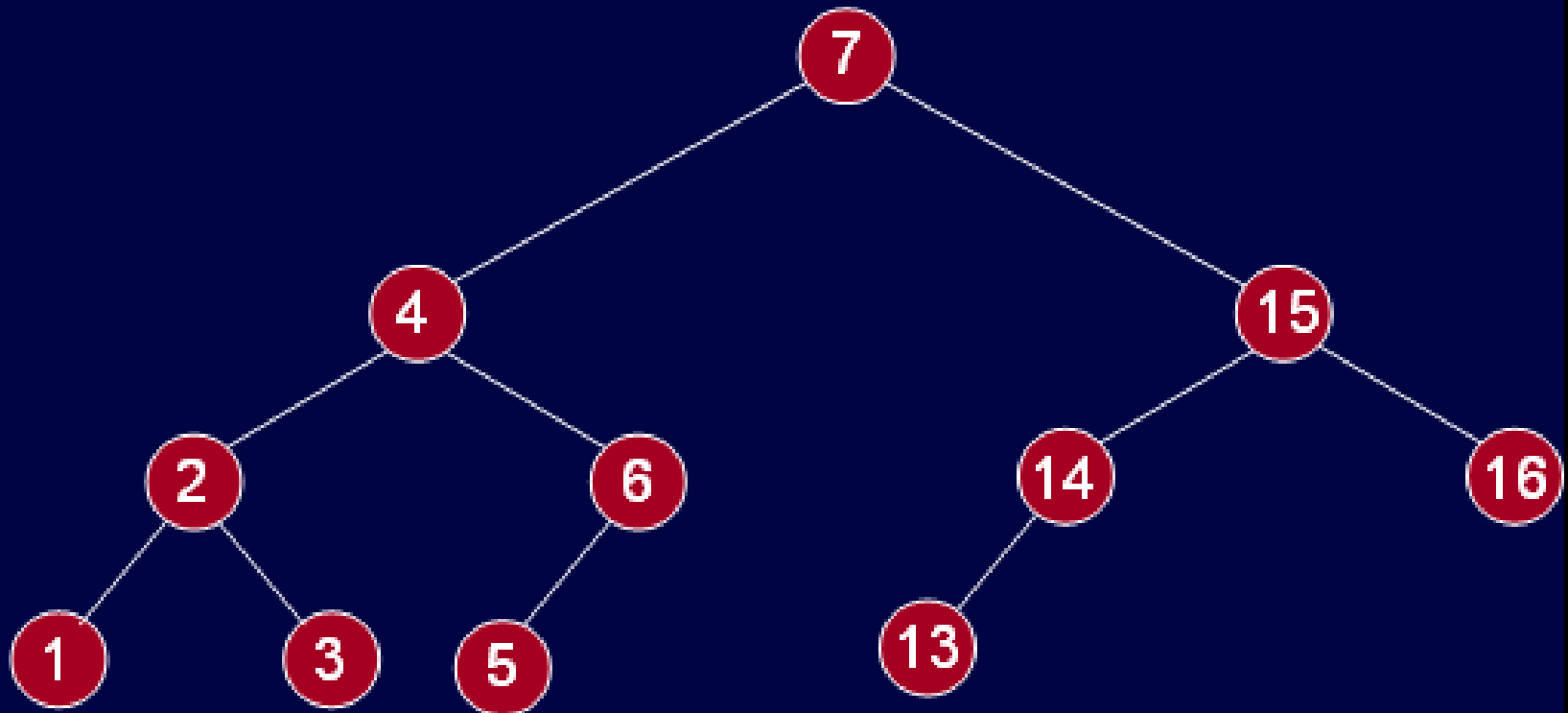
AVL Tree Building Example

Insert(13): single rotation



AVL Tree Building Example

Insert(13): single rotation



Thanks ...