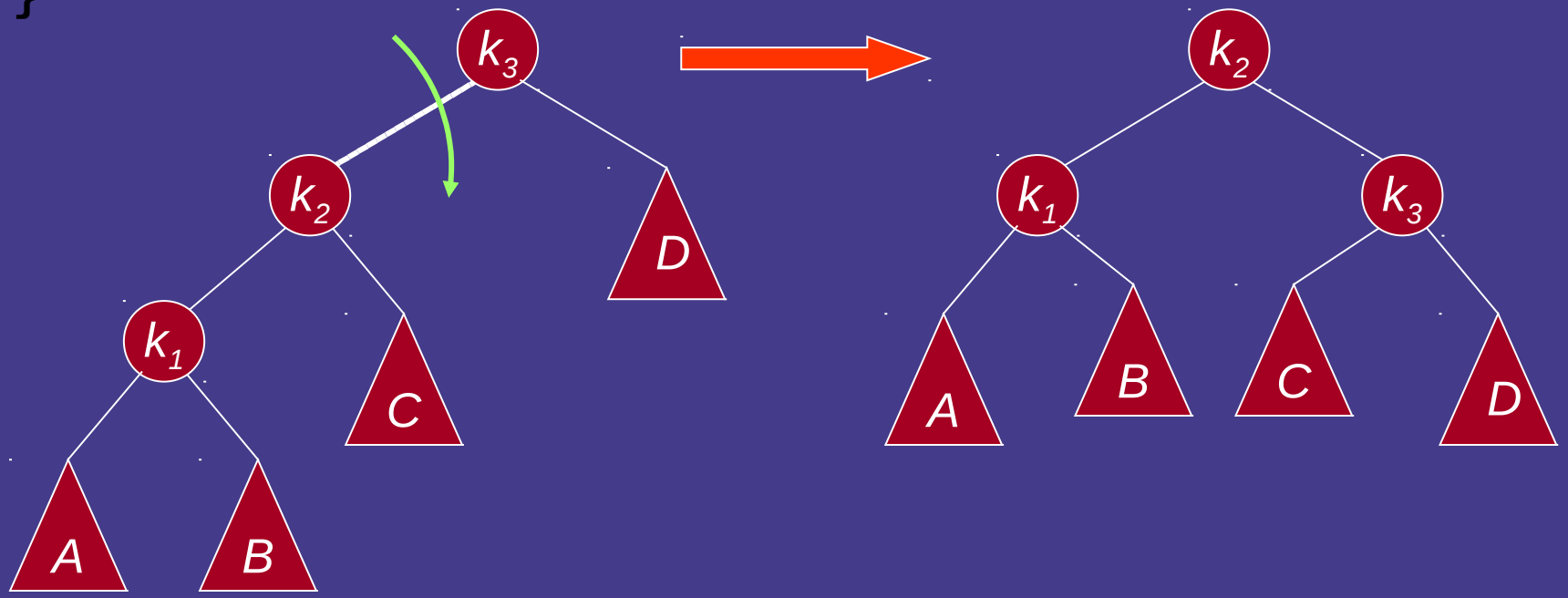
Lecture No.17
Deletion in AVL Trees

CC-213 Data Structures
Department of Computer Science
University of the Punjab

Slides modified very slightly from the late Dr. Sohail Aslam's lectures at VU

doubleRightLeftRotation

```
TreeNode<int>* doubleLeftRightRotation(TreeNode<int>* k3)
{
    if( k3 == NULL ) return NULL;
    // single left rotate with k1 (k3's left child)
    k3->setLeft( singleLeftRotation(k3->getLeft()));
    // now single right rotate with k3 as the root
    return singleRightRotation(k3);
}
```



Deletion in AVL Tree

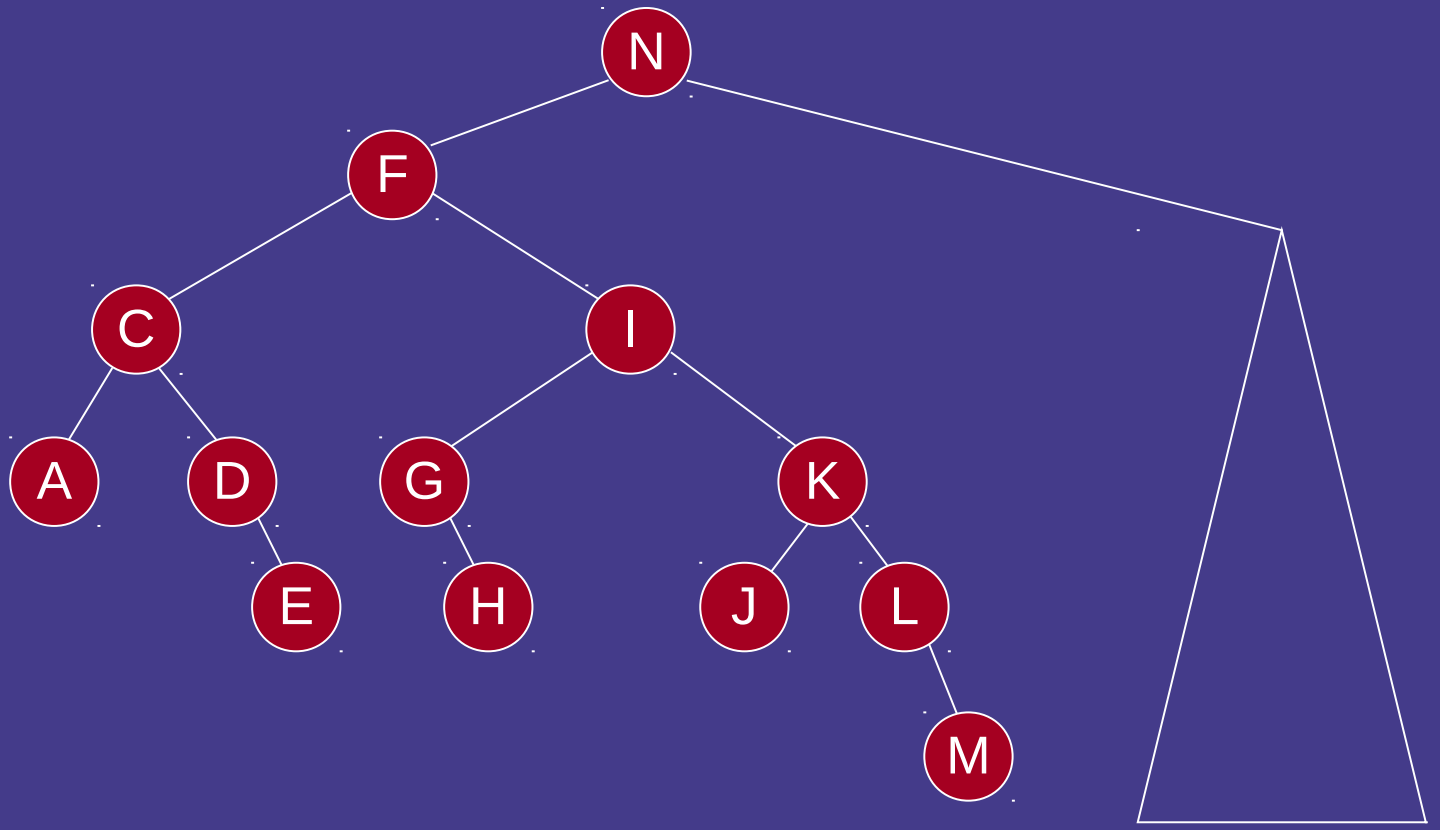
- Delete is the inverse of insert: given a value X and an AVL tree T, delete the node containing X and rebalance the tree, if necessary.
- Turns out that deletion of a node is considerably more complex than insert

Deletion in AVL Tree

- Insertion in a height-balanced tree requires at most one single rotation or one double rotation.
- We can use rotations to restore the balance when we do a deletion.
- We may have to do a rotation at every level of the tree: $\log_2 N$ rotations in the worst case.

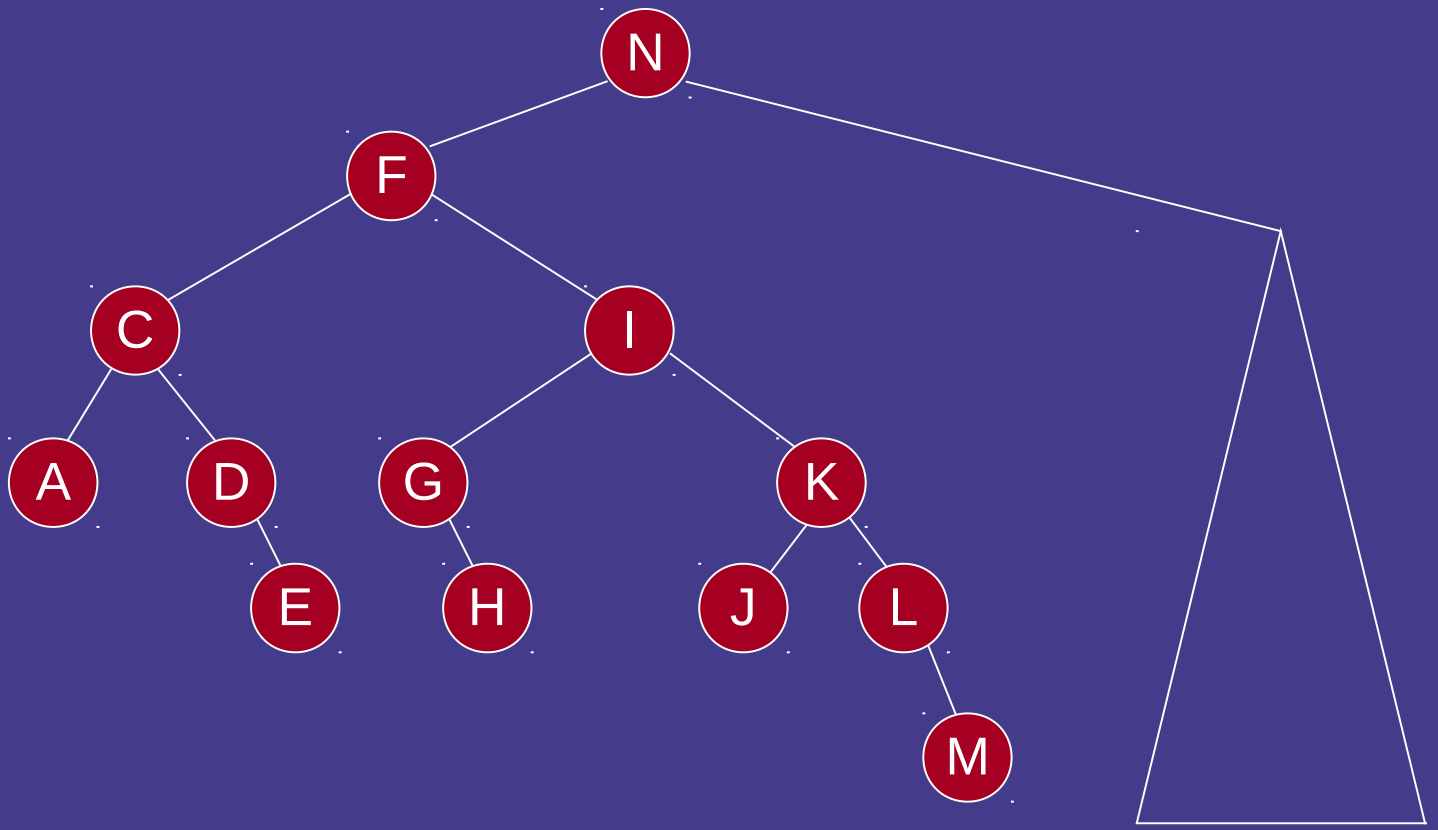
Deletion in AVL Tree

- Here is a tree that causes this worse case number of rotations when we delete A. At every node in N's left subtree, the left subtree is one shorter than the right subtree.



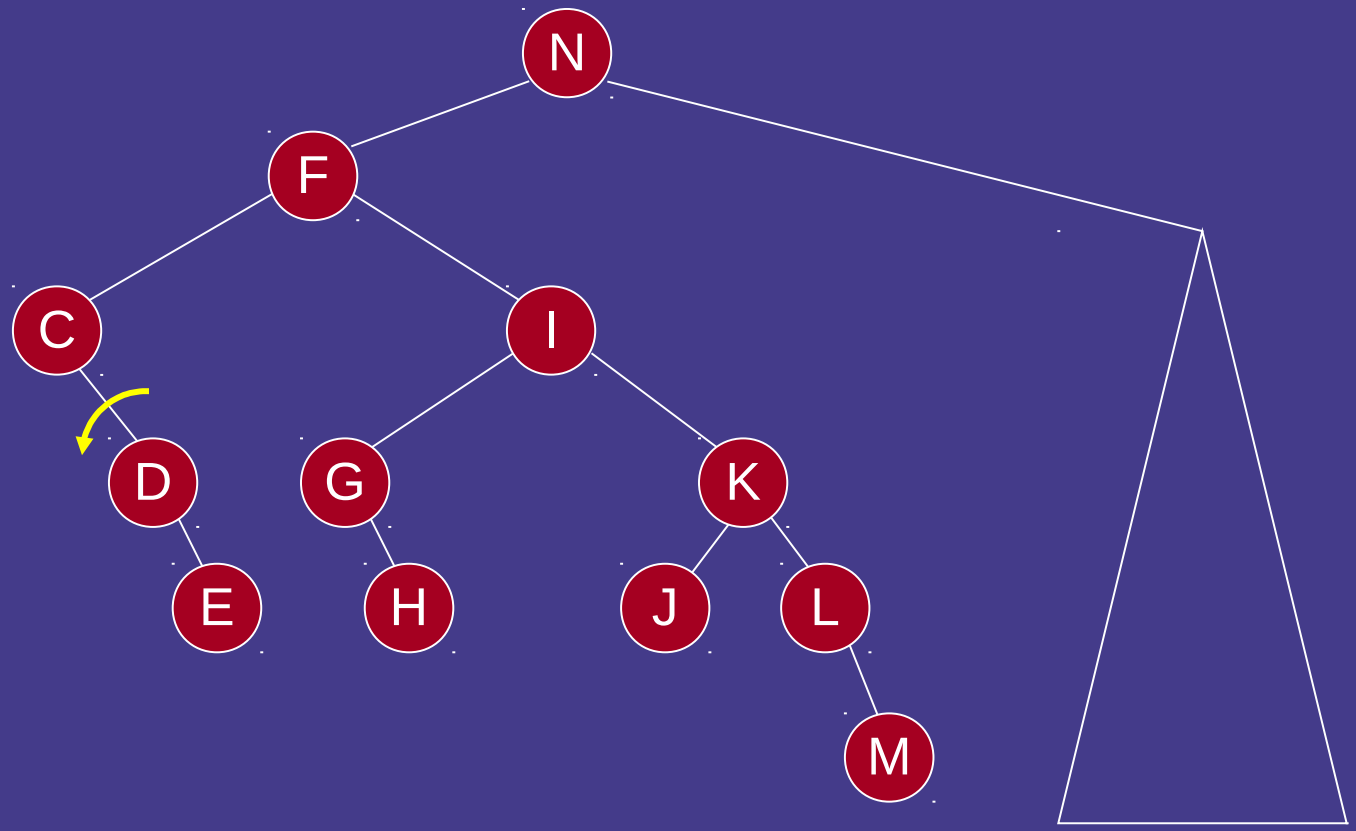
Deletion in AVL Tree

- Deleting A upsets balance at C. When rotate (D up, C down) to fix this



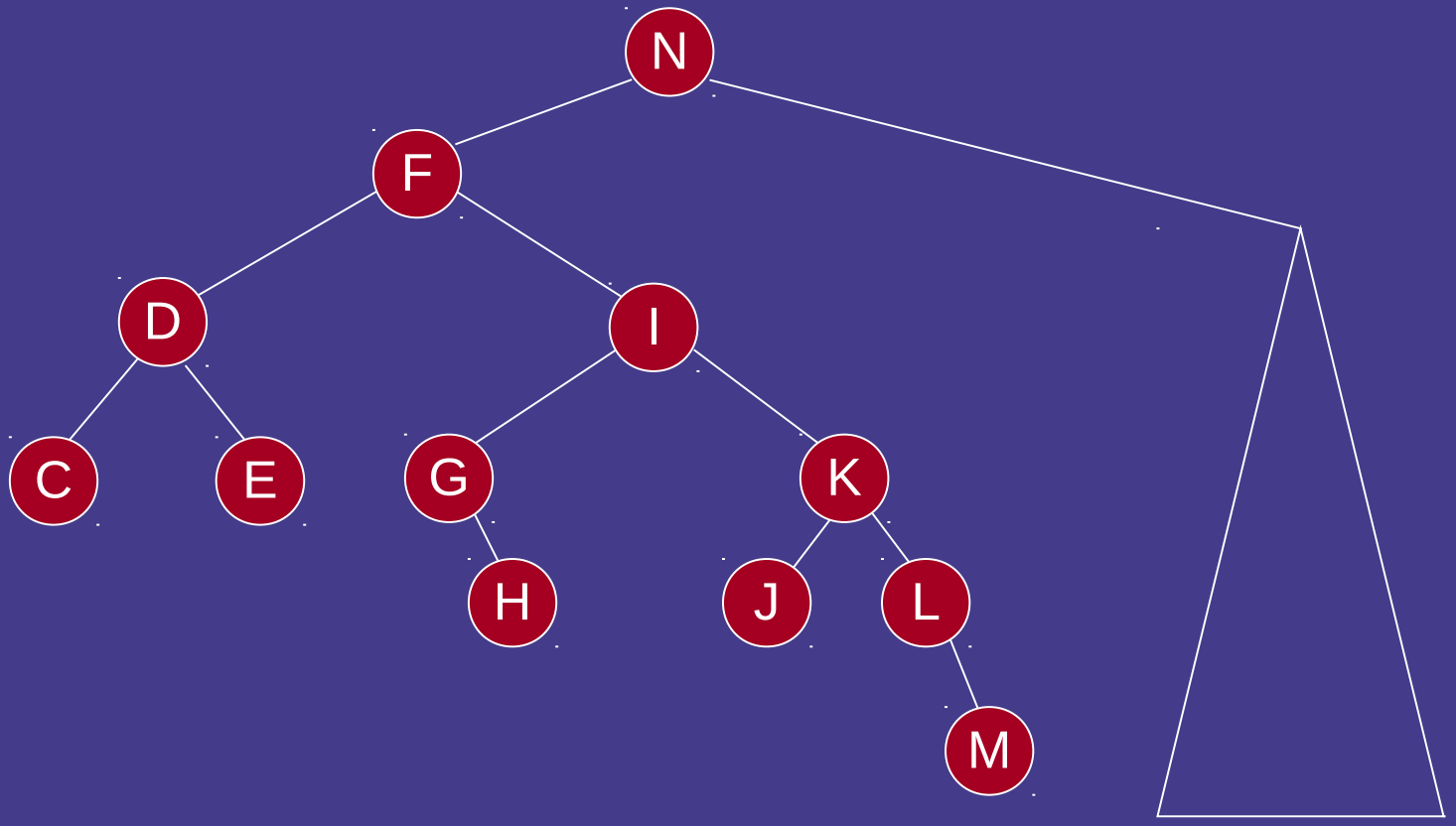
Deletion in AVL Tree

- Deleting A upsets balance at C. When rotate (D up, C down) to fix this



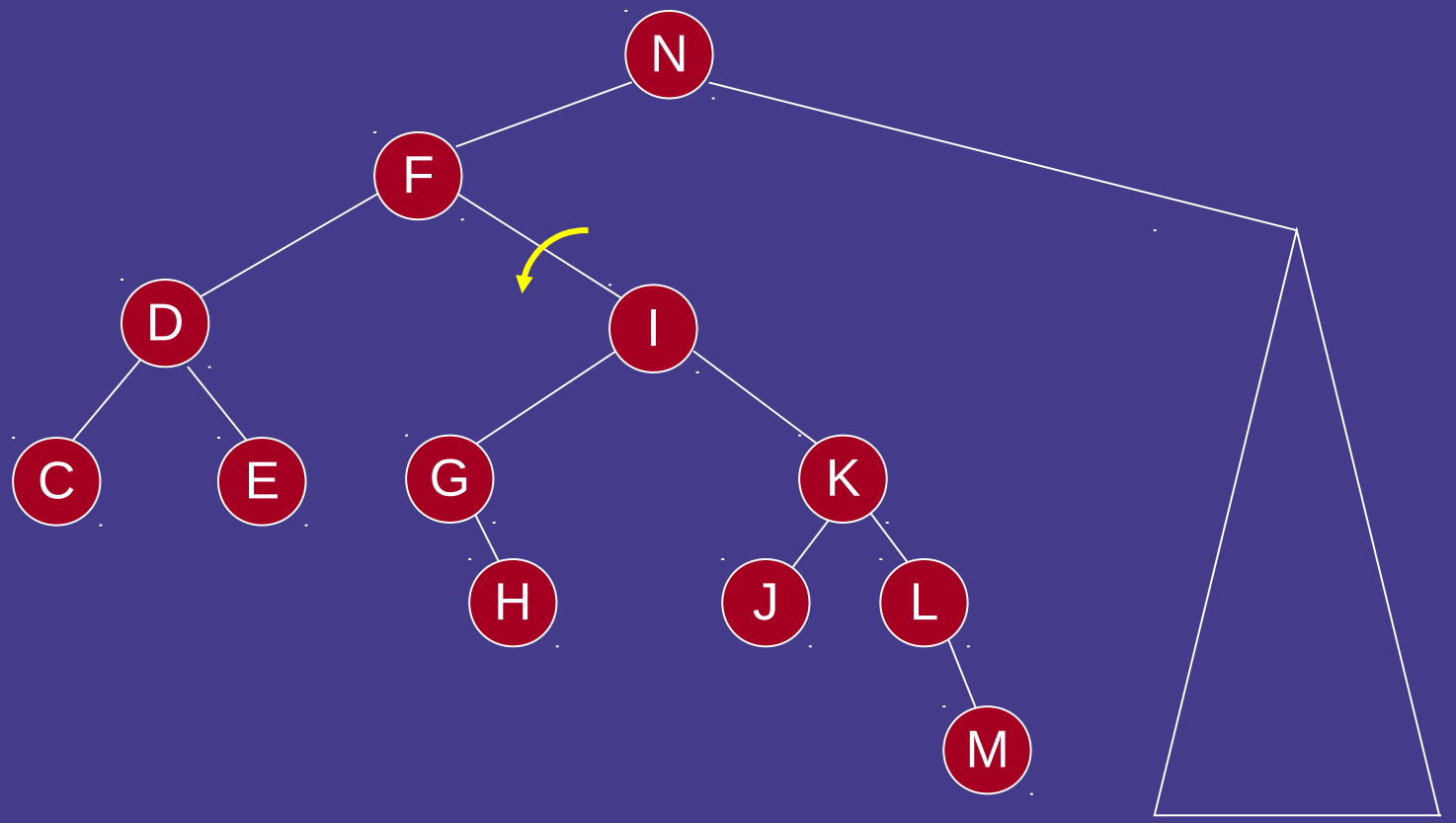
Deletion in AVL Tree

- The whole of F's left subtree gets shorter. We fix this by rotation about F-I: F down, I up.



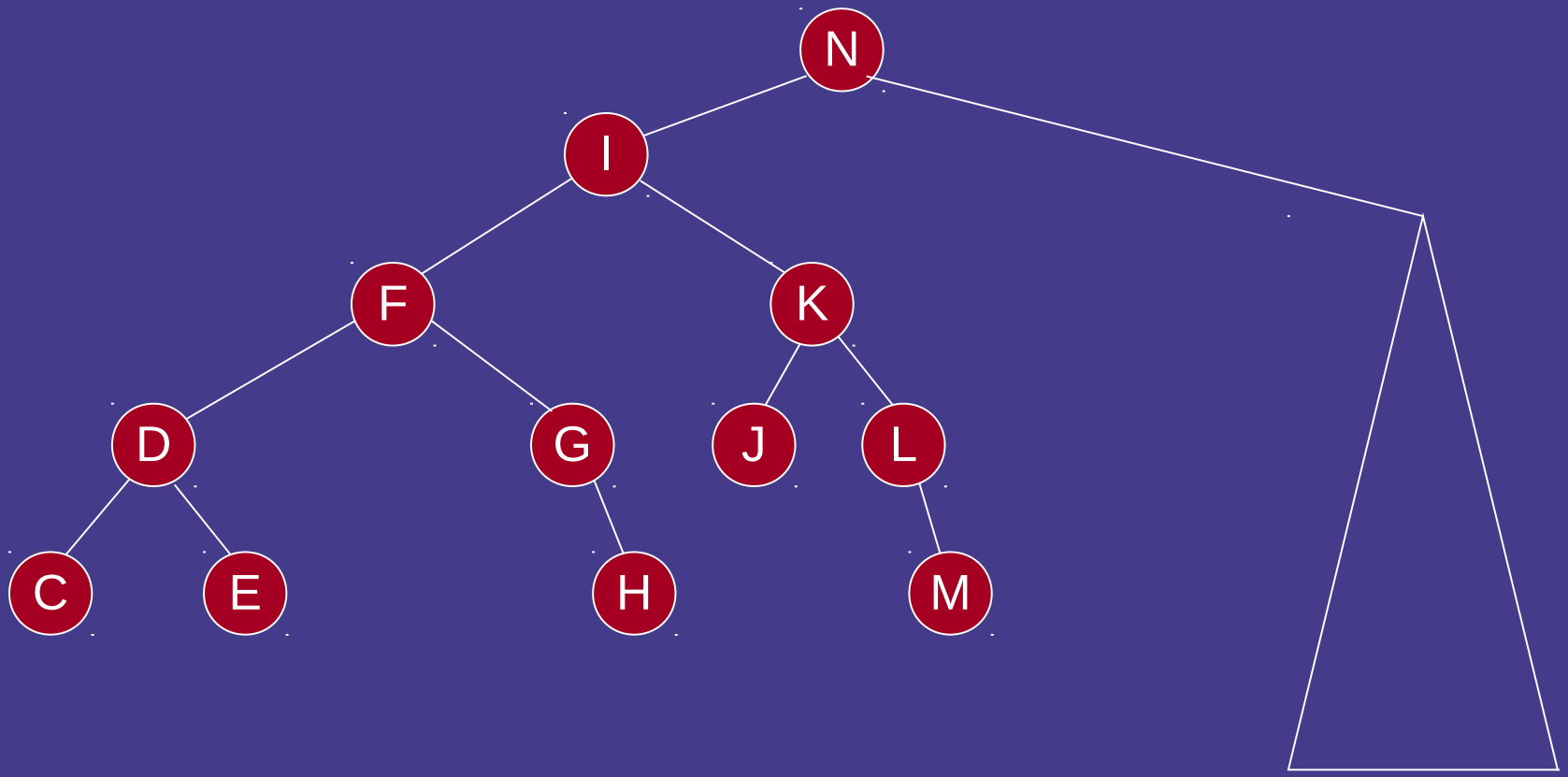
Deletion in AVL Tree

- The whole of F's left subtree gets shorter. We fix this by rotation about F-I: F down, I up.



Deletion in AVL Tree

- This could cause imbalance at N.
- The rotations propagated to the root.



Deletion in AVL Tree

Procedure

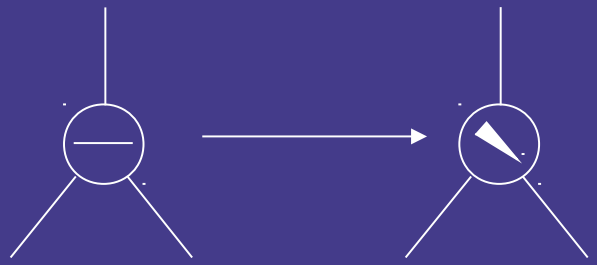
- Delete the node as in binary search tree (BST).
- The node deleted will be either a leaf or have just one subtree.
- Since this is an AVL tree, if the deleted node has one subtree, that subtree contains only one node (why?)
- Traverse up the tree from the deleted node checking the balance of each node.

Deletion in AVL Tree

- There are 5 cases to consider.
- Let us go through the cases graphically and determine what action to take.
- We will not develop the C++ code for deleteNode in AVL tree. This will be left as an exercise.

Deletion in AVL Tree

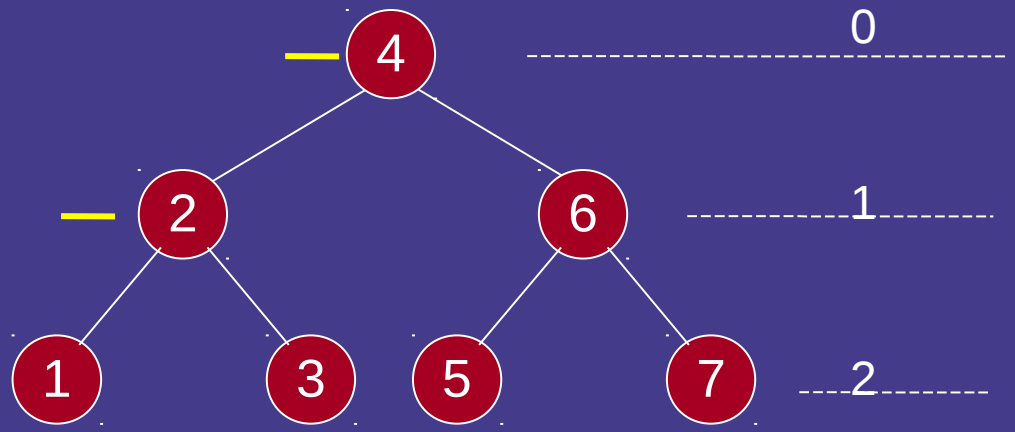
Case 1a: the parent of the deleted node had a balance of 0 and the node was deleted in the parent's *left* subtree.



Action: change the balance of the parent node and stop. No further effect on balance of any higher node.

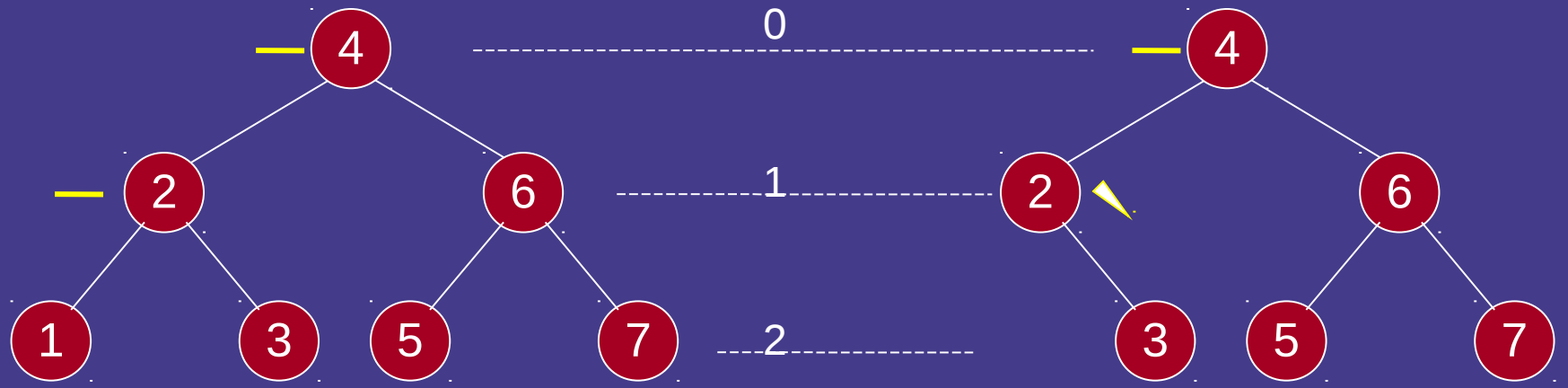
Deletion in AVL Tree

Here is why; the height of left tree does not change.



Deletion in AVL Tree

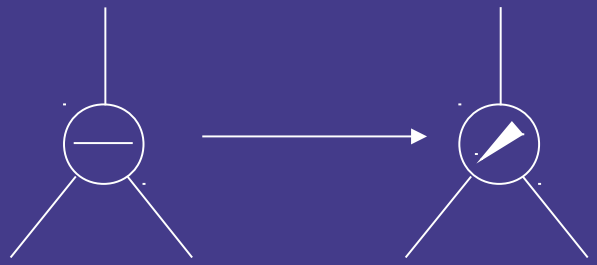
Here is why; the height of left tree does not change.



remove(1)

Deletion in AVL Tree

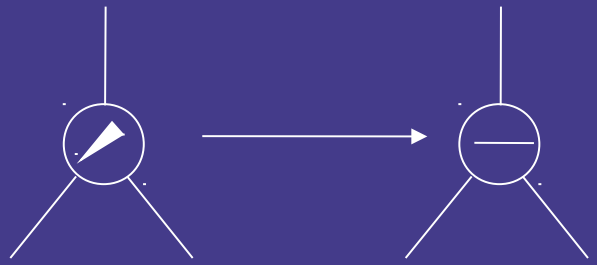
Case 1b: the parent of the deleted node had a balance of 0 and the node was deleted in the parent's *right* subtree.



Action: (same as *1a*) change the balance of the parent node and stop. No further effect on balance of any higher node.

Deletion in AVL Tree

Case 2a: the parent of the deleted node had a balance of 1 and the node was deleted in the parent's *left* subtree.



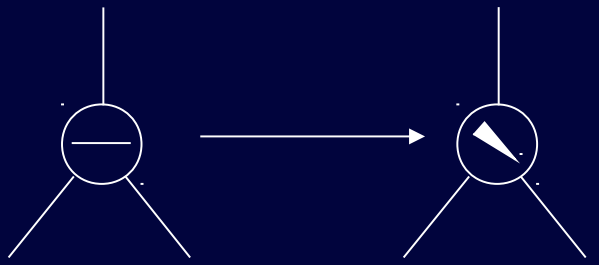
Action: change the balance of the parent node. May have caused imbalance in higher nodes so continue up the tree.

Deletion in AVL Tree

- There are 5 cases to consider.
- Let us go through the cases graphically and determine what action to take.
- We will not develop the C++ code for deleteNode in AVL tree. This will be left as an exercise.

Deletion in AVL Tree

Case 1a: the parent of the deleted node had a balance of 0 and the node was deleted in the parent's *left* subtree.

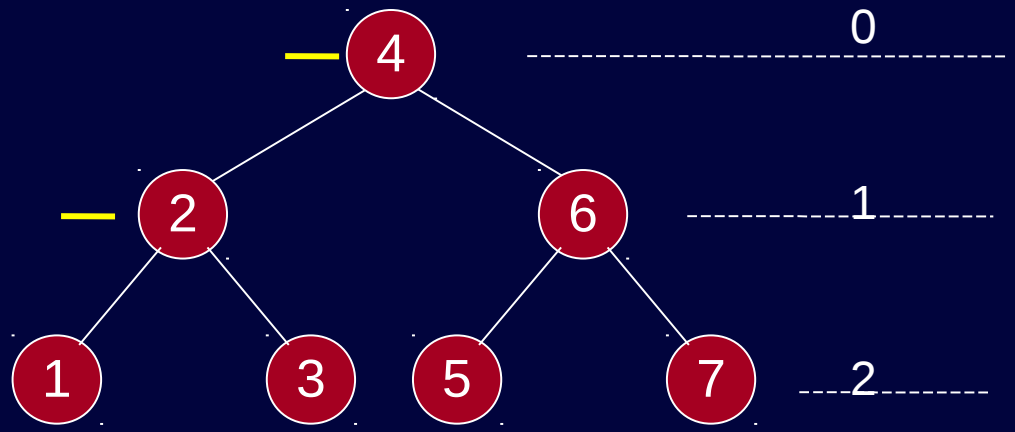


Delete on
this side

Action: change the balance of the parent node and stop. No further effect on balance of any higher node.

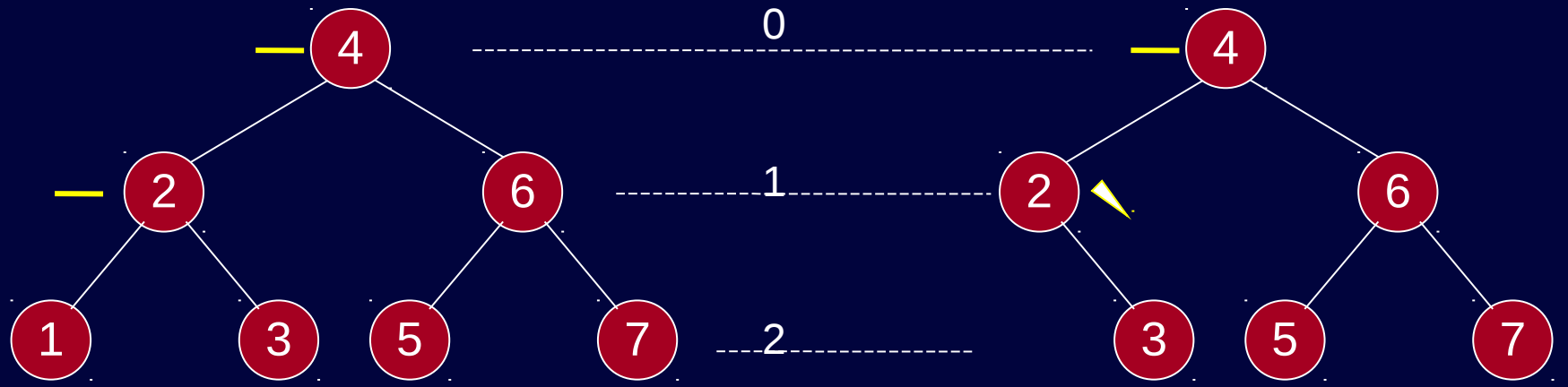
Deletion in AVL Tree

Here is why; the height of left tree does not change.



Deletion in AVL Tree

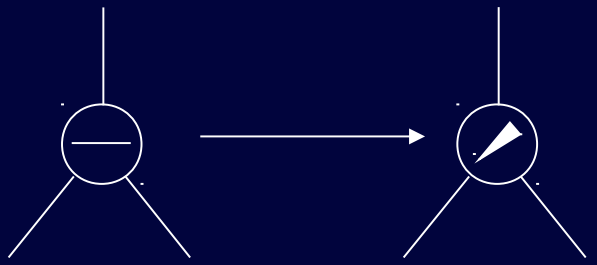
Here is why; the height of left tree does not change.



remove(1)

Deletion in AVL Tree

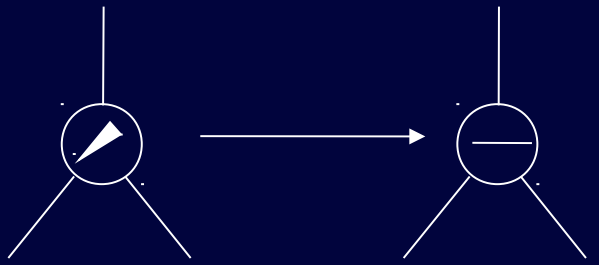
Case 1b: the parent of the deleted node had a balance of 0 and the node was deleted in the parent's *right* subtree.



Action: (same as *1a*) change the balance of the parent node and stop. No further effect on balance of any higher node.

Deletion in AVL Tree

Case 2a: the parent of the deleted node had a balance of 1 and the node was deleted in the parent's *left* subtree.

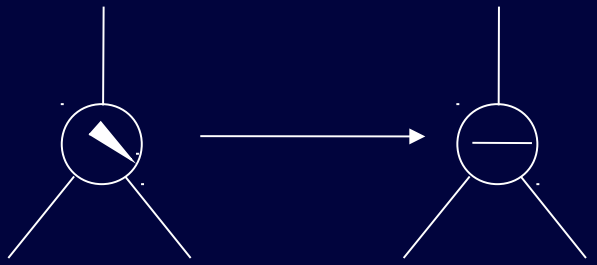


Delete on
this side

Action: change the balance of the parent node. May have caused imbalance in higher nodes so continue up the tree.

Deletion in AVL Tree

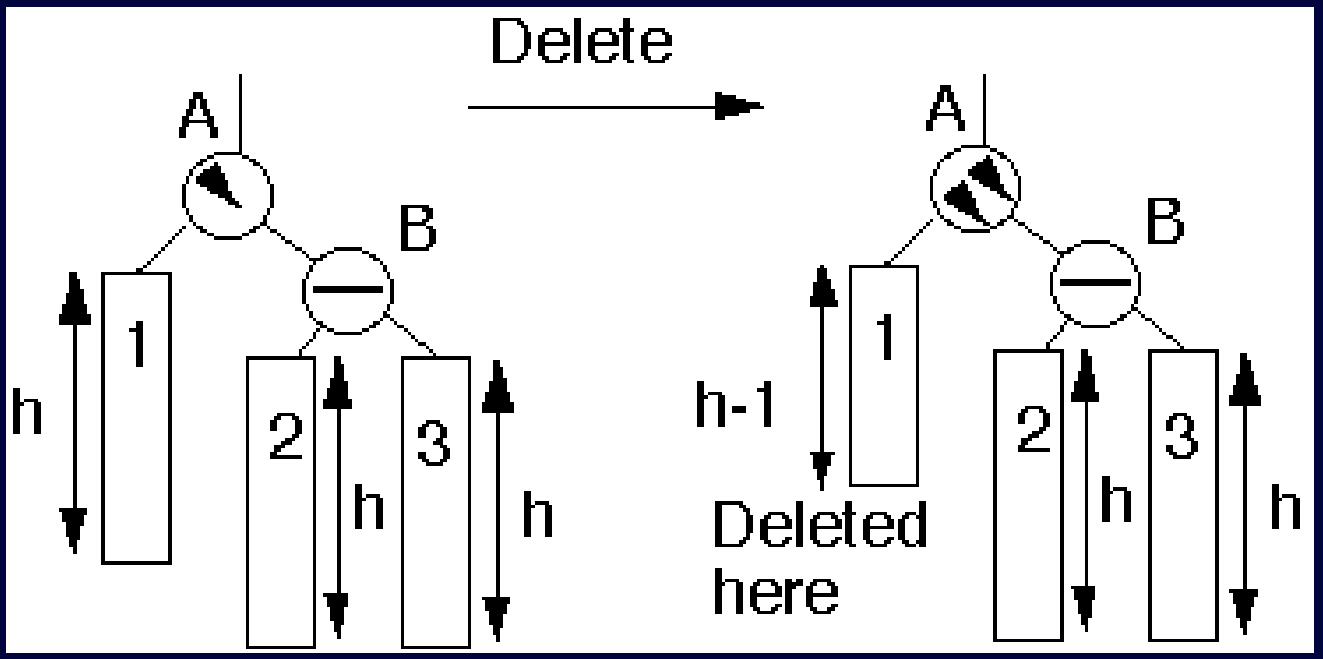
Case 2b: the parent of the deleted node had a balance of -1 and the node was deleted in the parent's *right* subtree.



Action: same as 2a: change the balance of the parent node. May have caused imbalance in higher nodes so continue up the tree.

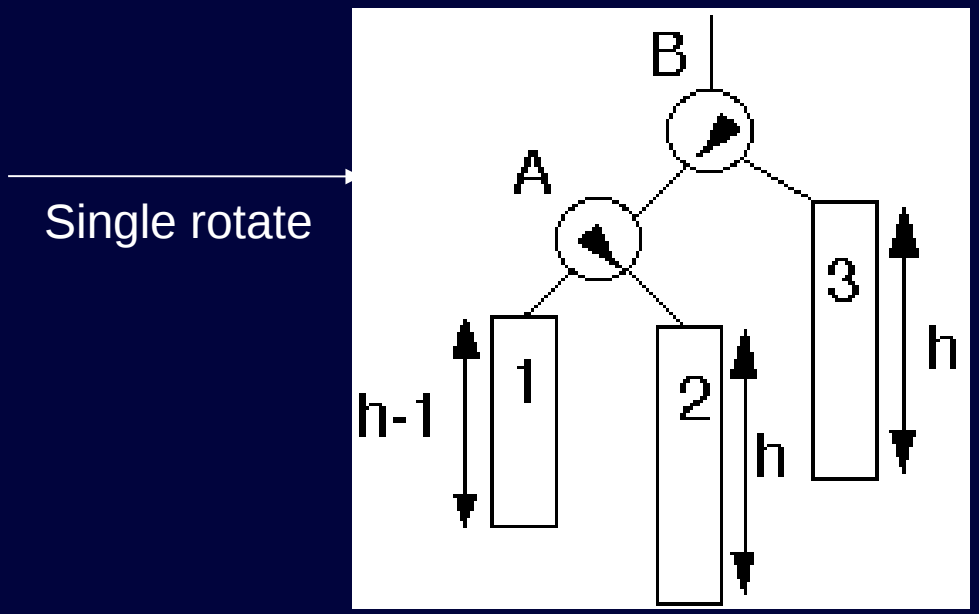
Deletion in AVL Tree

Case 3a: the parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was balanced.



Deletion in AVL Tree

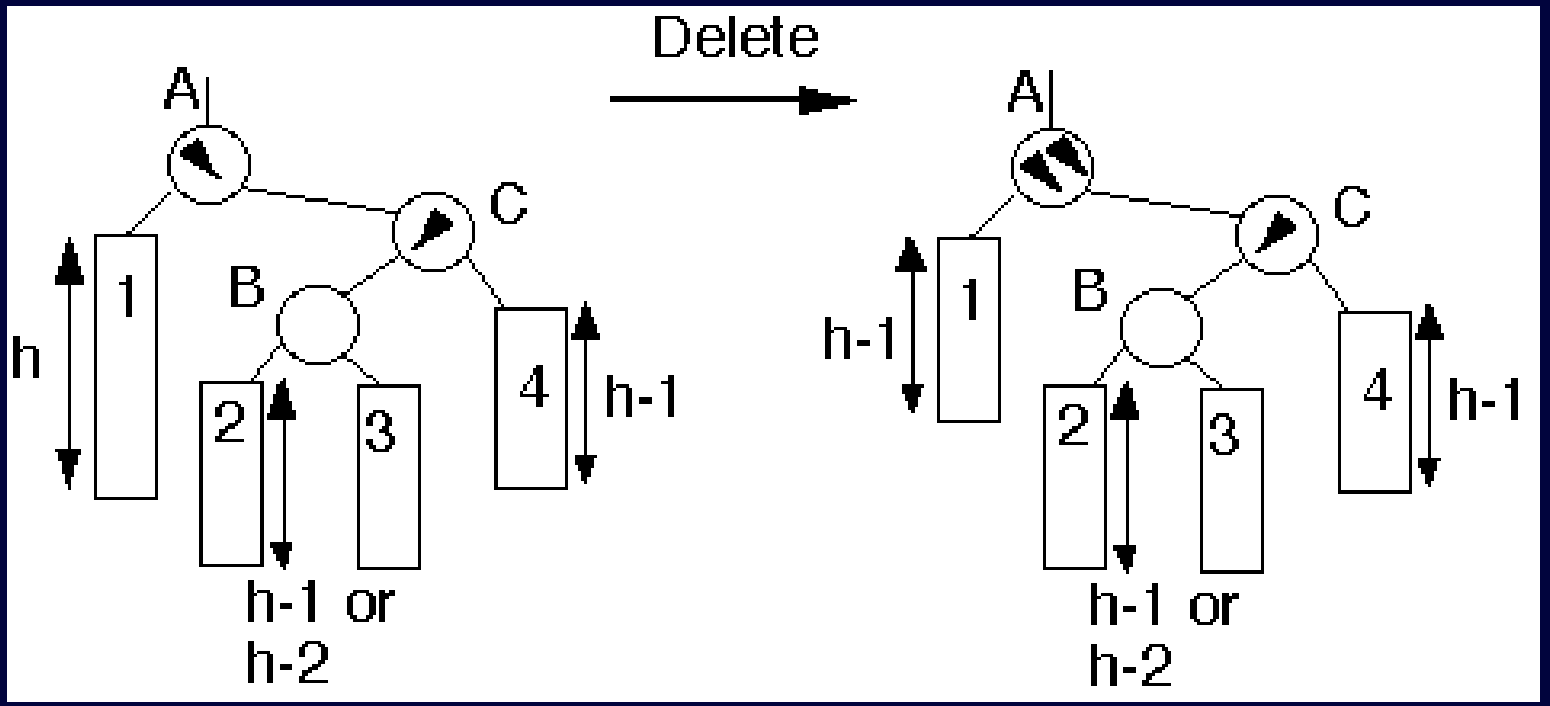
Case 3a: the parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was balanced.



Action: perform single rotation, adjust balance. No effect on balance of higher nodes so stop here.

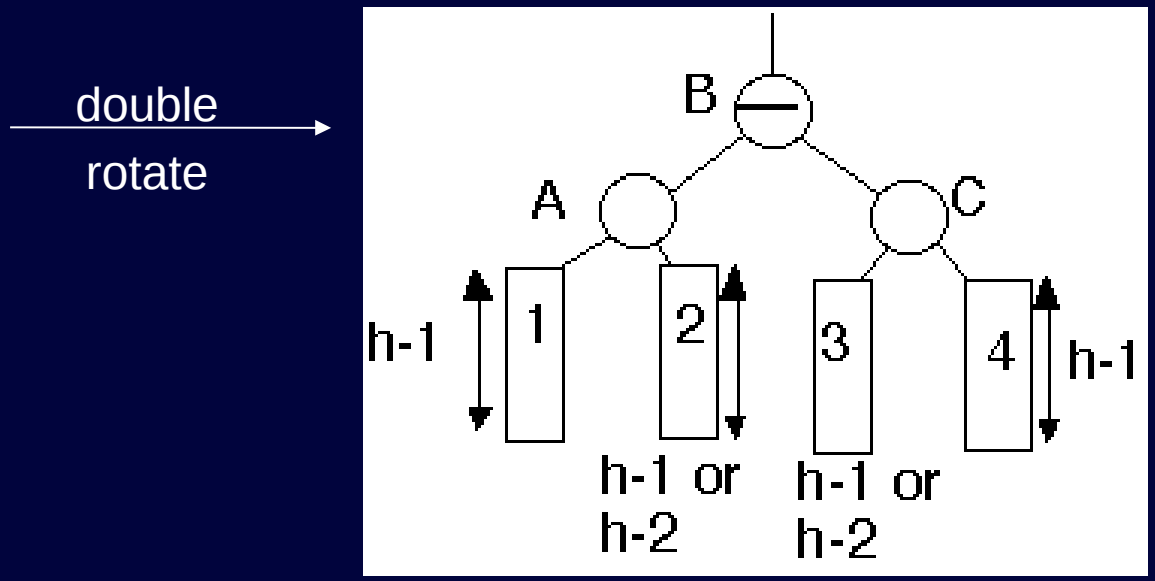
Deletion in AVL Tree

Case 4a: parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was unbalanced.



Deletion in AVL Tree

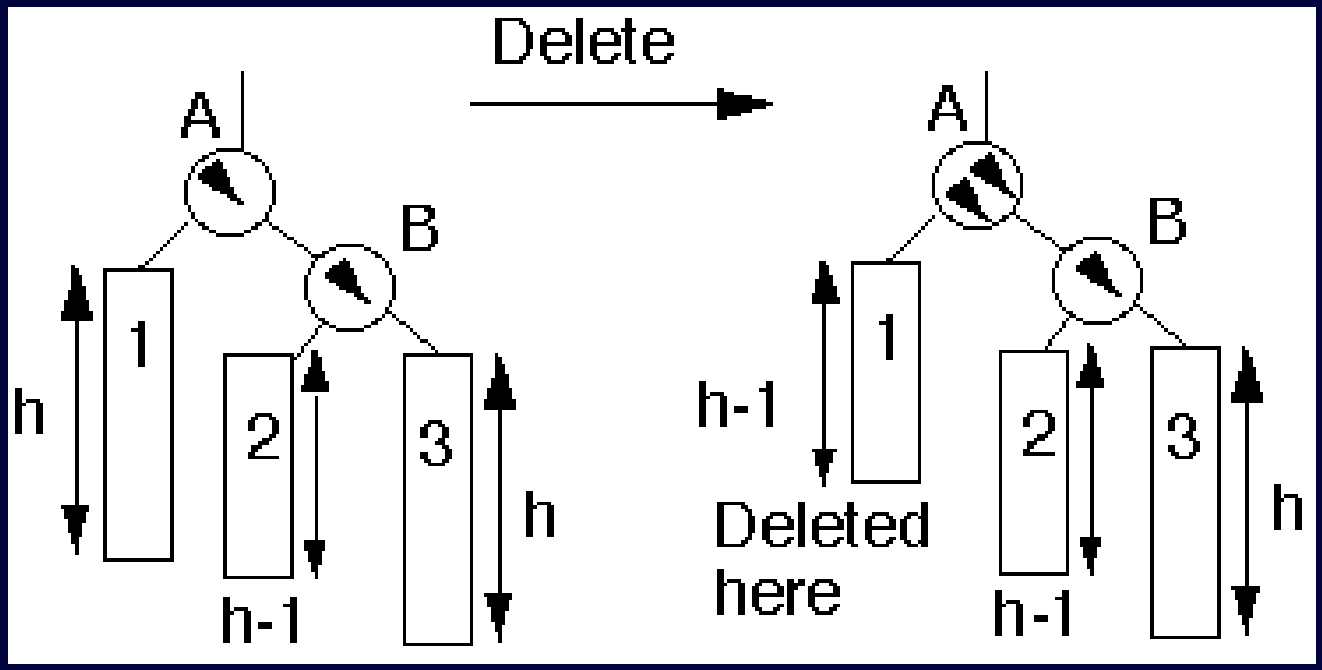
Case 4a: parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was unbalanced.



Action: Double rotation at B. May have effected the balance of higher nodes, so continue up the tree.

Deletion in AVL Tree

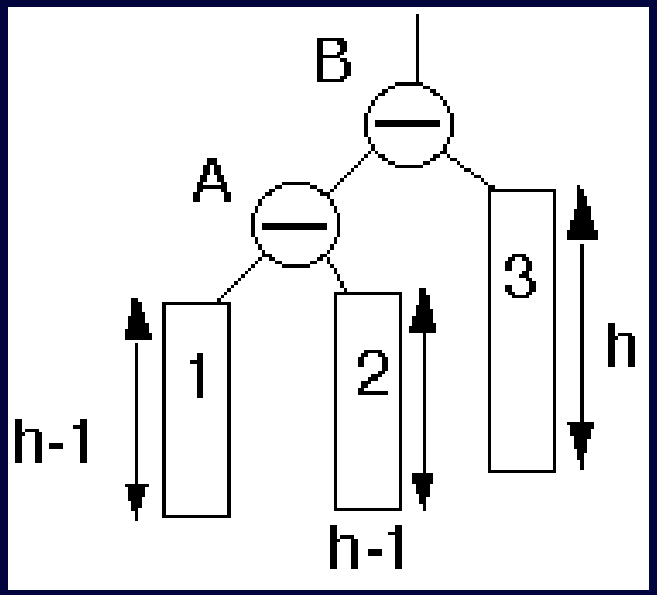
Case 5a: parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was unbalanced.



Deletion in AVL Tree

Case 5a: parent had balance of -1 and the node was deleted in the parent's *left* subtree, right subtree was unbalanced.

single rotate



Action: Single rotation at B. May have effected the balance of higher nodes, so continue up the tree.

Other Uses of Binary Trees

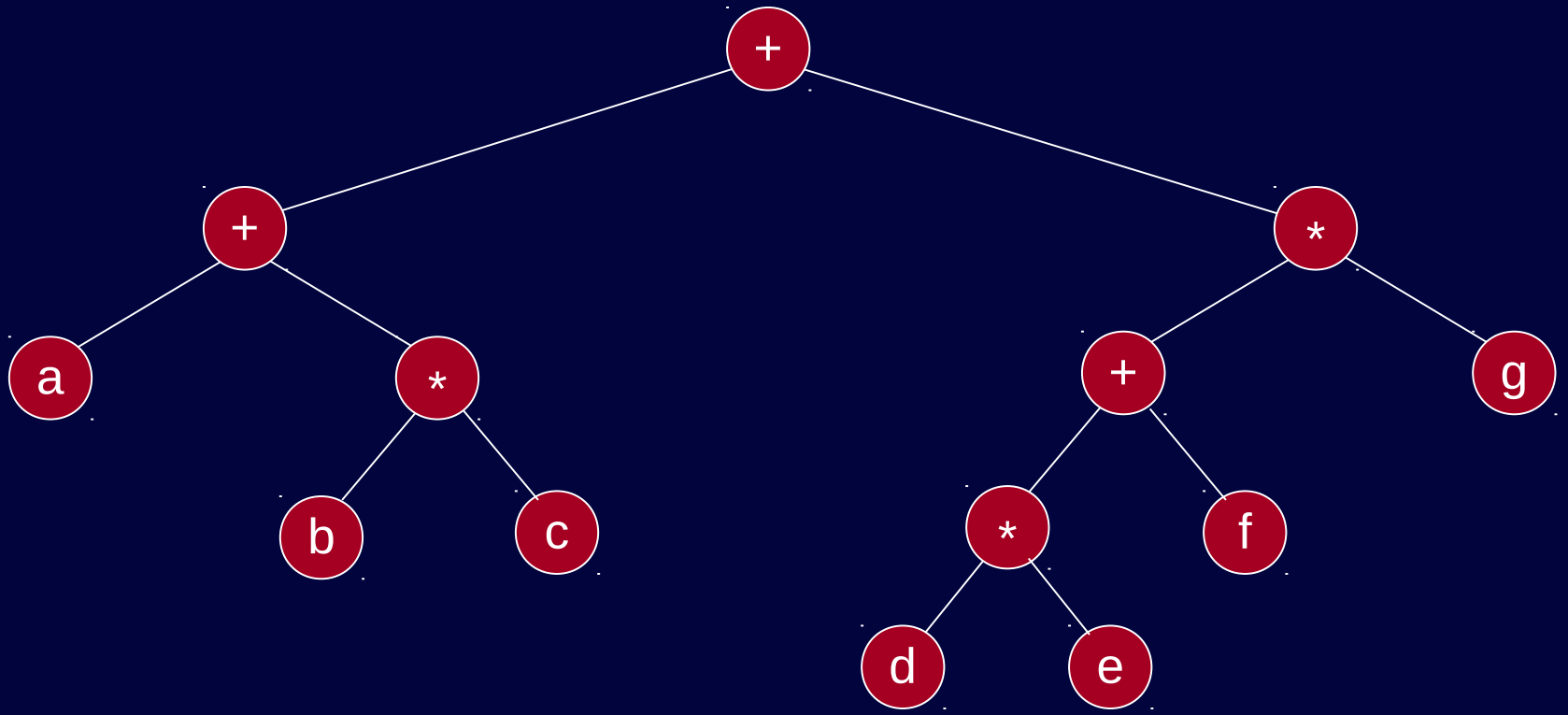
Expression Trees

Expression Trees

- *Expression trees*, and the more general parse trees and abstract syntax trees are significant components of compilers.
- Let us consider the expression tree.

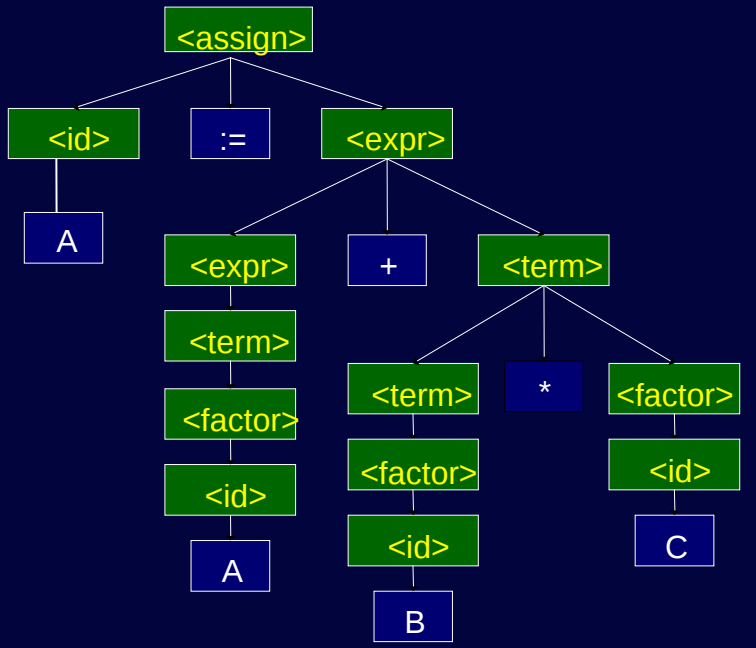
Expression Tree

(a+b*c)+((d*e+f)*g)



Parse Tree in Compiler

A := A + B * C



Expression grammar

```
<assign> → <id> := <expr>
<id>  → A | B | C
<expr>   → <expr> + <term> | <term>
<term>   → <term> * <factor> | <factor>
<factor> → ( <expr> ) | <id>
```

Parse Tree for an SQL query

Consider querying a movie database

Find the titles for movies with stars born in 1960

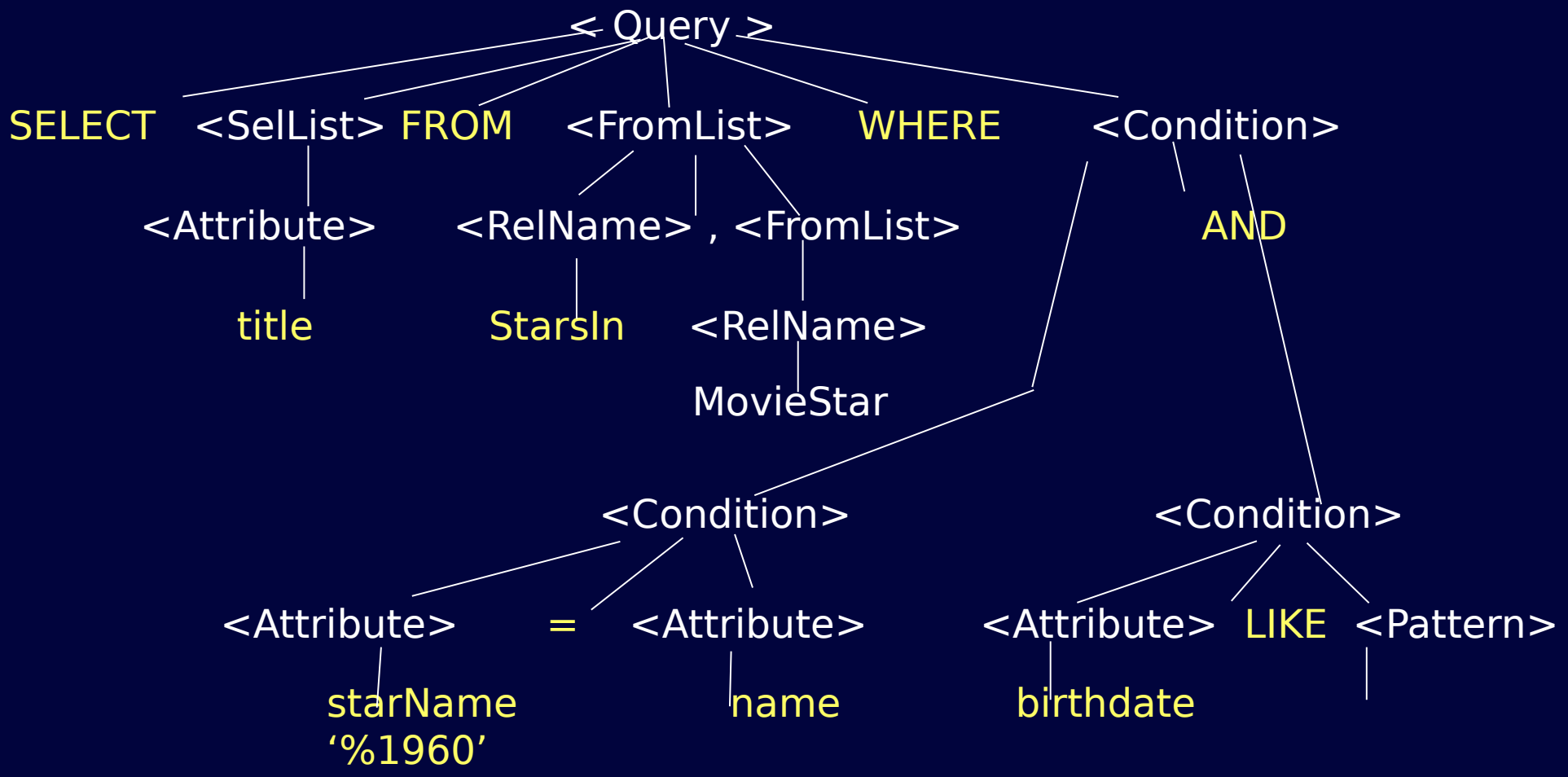
The database has tables

StarsIn(title, year, starName)

MovieStar(name, address, gender, birthdate)

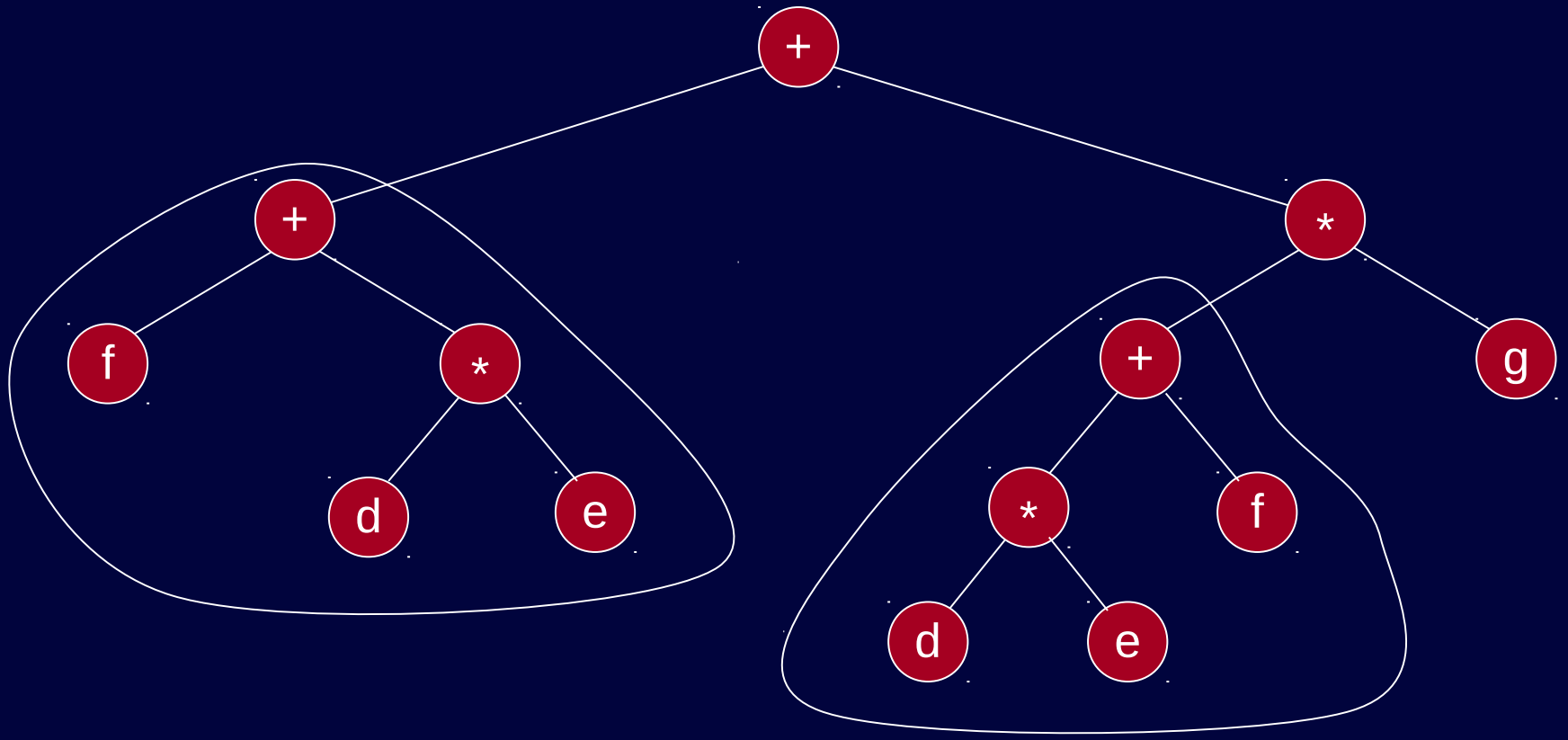
```
SELECT title
FROM    StarsIn, MovieStar
WHERE starName = name AND birthdate LIKE '%1960' ;
```

SQL Parse Tree



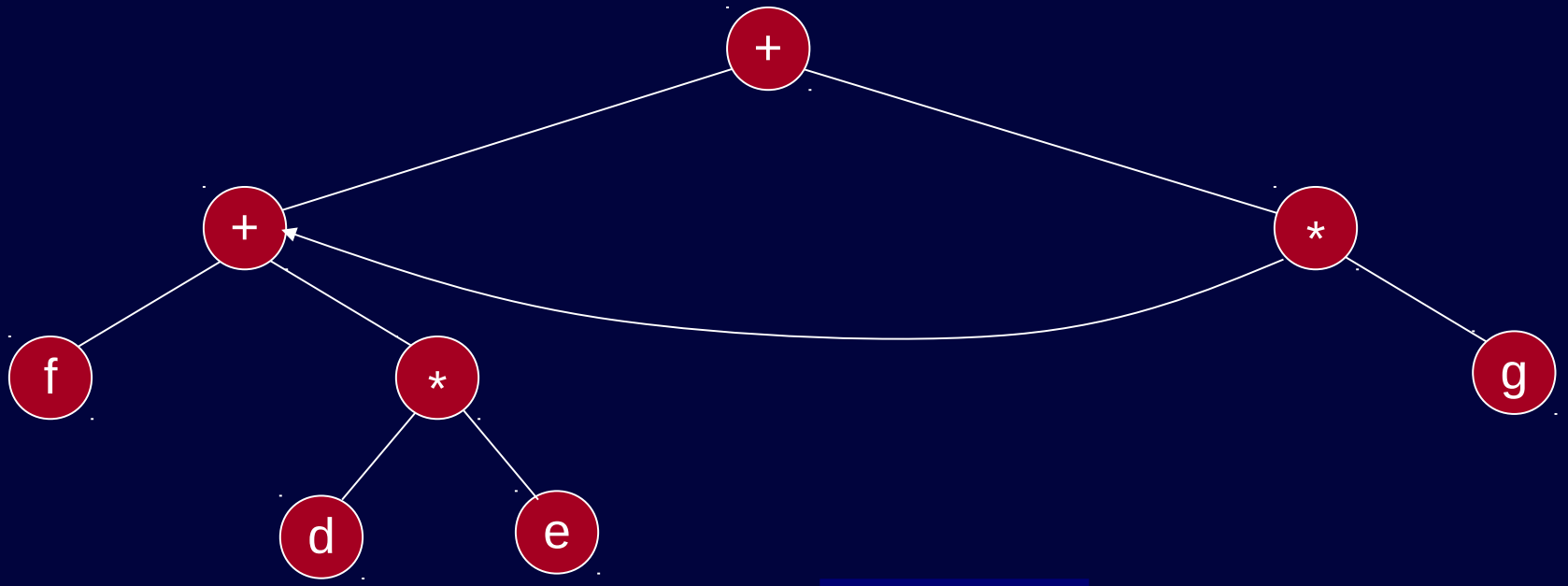
Compiler Optimization

Common subexpression:
(f+d*e)+((d*e+f)*g)



Compiler Optimization

(Common subexpression:
(f+d*e)+((d*e+f)*g)



Graph!