NANYANG TECHNOLOGICAL UNIVERSITY

CE1007/CZ1007 DATA STRUCTURES

Lecture 10: **Tree Balancing**

Dr. Owen Noel Newton Fernando

College of Engineering

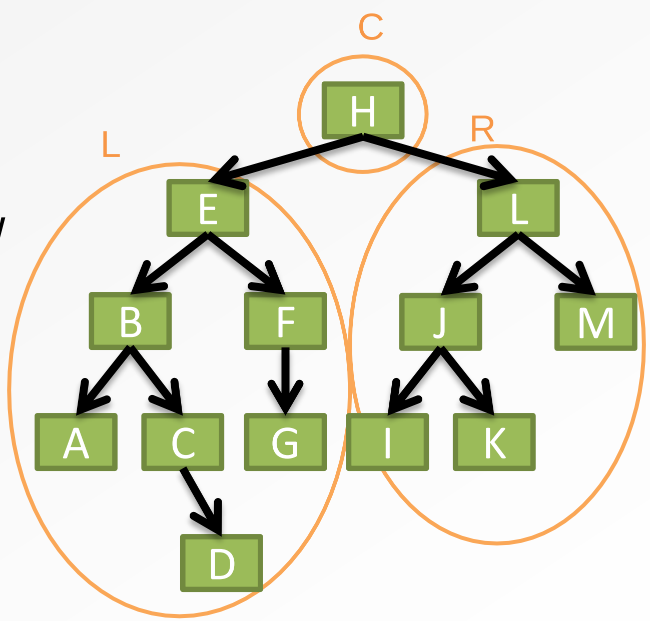
School of Computer Science and Engineering

OUTLINE

- **Importance of balance for BSTs**
- Tree Balancing

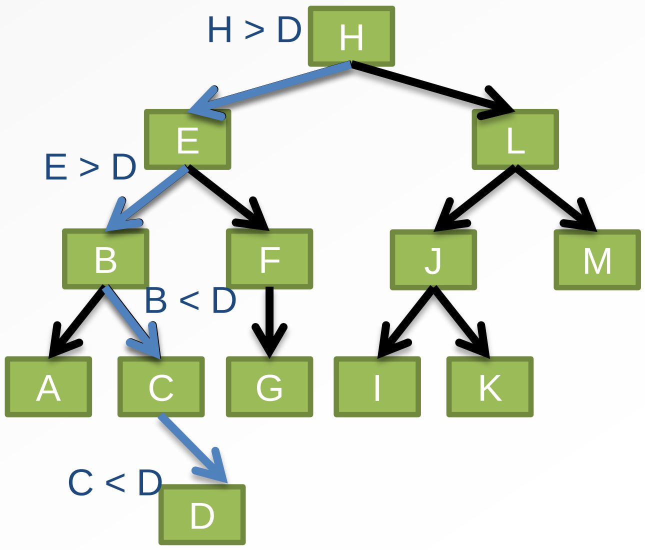
RECALL: WHY USE BSTs?

- BSTs are a special form of BT
- At every node, L < C < R
  - At every node, we always know whether to continue searching in the left or right subtree
  - If we continue searching in the left subtree, all nodes in the right subtree can be ignored



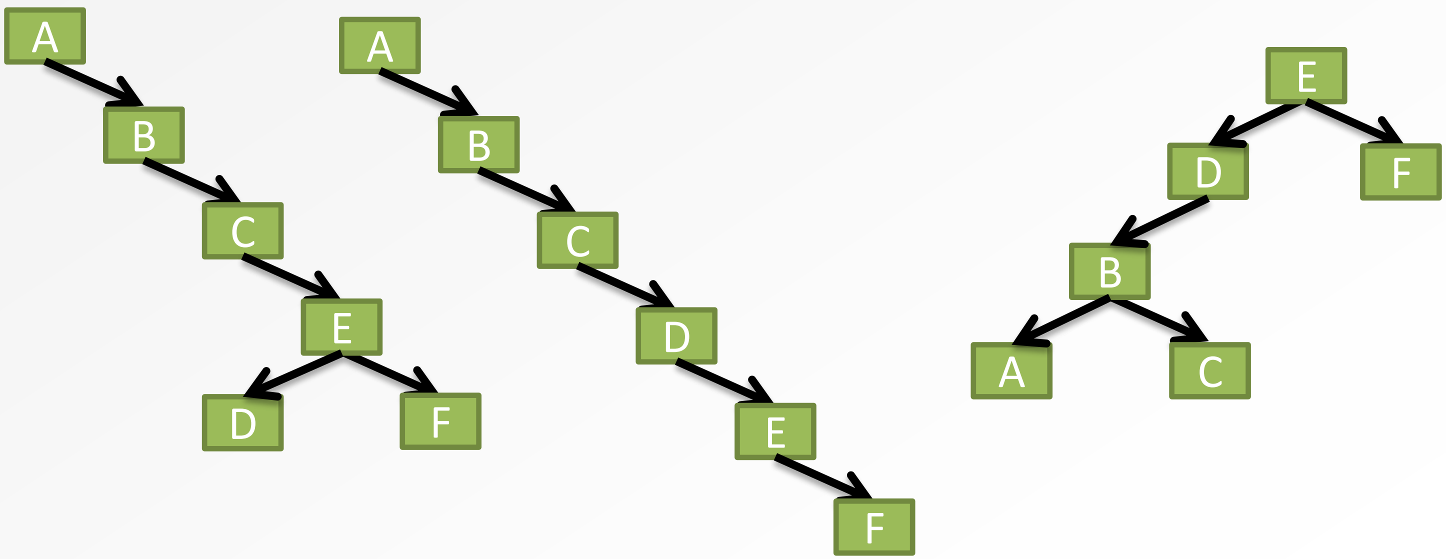
RECALL: EFFICIENT SEARCH WITH BSTs

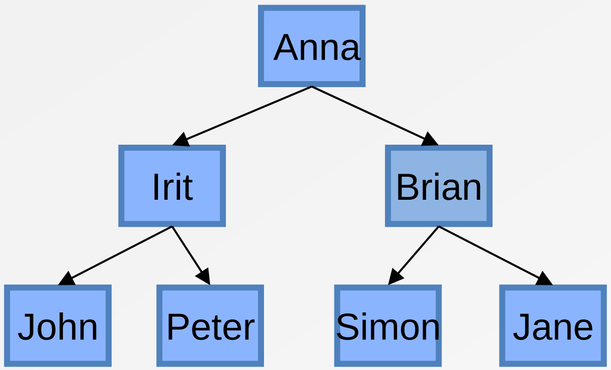
- Search is efficient because we traverse <u>one</u> external path
- # operations is proportional to <u>path length</u>
- Try to keep path length low
  - Ie, try to keep tree balanced

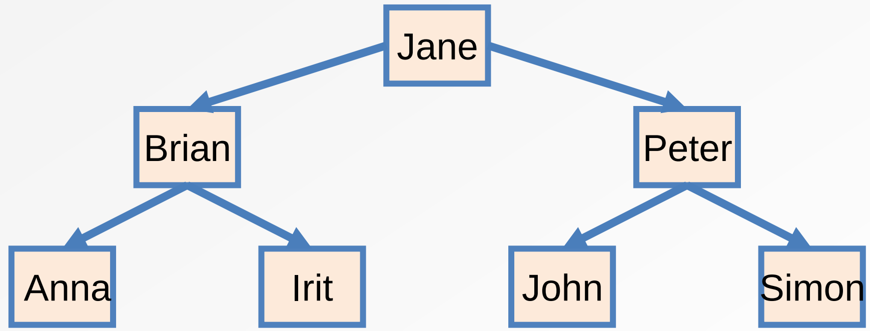


RECALL: EFFICIENT SEARCH WITH BSTs

- But an imbalanced BST starts to look more like a linked list!
- Path length is high







How many nodes are visited during search?
In general, for a BT with n nodes:
--best case: First node in traversal
--worst case: Last node in traversal, n

How many nodes are visited during search?
In general, for a BST with n nodes:
--best case: First node in traversal
--worst case:
leaf node: the height of the root + 1
Minimal height H = $\lfloor \log_2 n \rfloor$

Number of nodes visited is proportional to the height of the tree
Try to keep the height of tree short
Try to keep the tree balanced

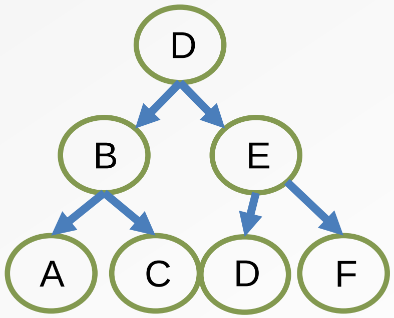
HOW DO WE GET MINIMAL H= $\lfloor \log_2 n \rfloor$

- For a tree with height *H*, we have:

  $$n \le 2^{H+1} - 1$$

  where *n* is the size of the tree.

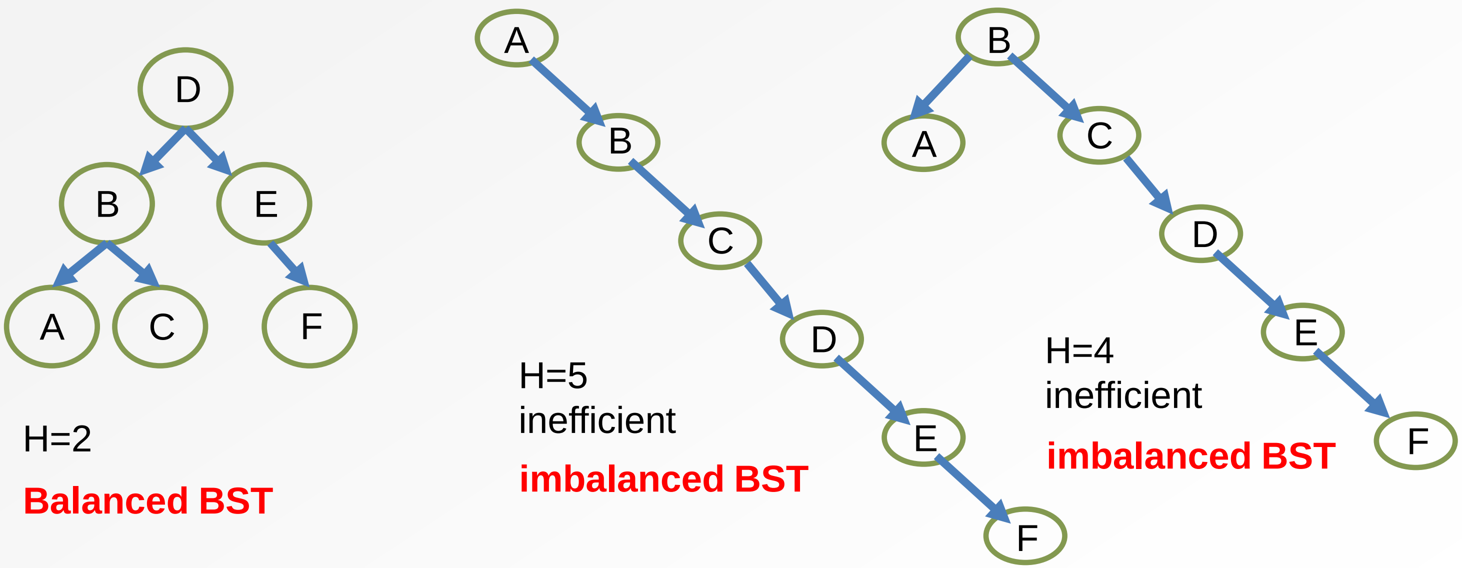
- **Tree Height ➔ H** $\ge \lfloor \log_2 n \rfloor$
- **Minimal Height =** $\lfloor \log_2 n \rfloor$
- **Height of a node** = number of links from that node to the deepest leaf node



Maximal size tree with H=2

RECALL: EFFICIENT SEARCH WITH BSTs

- What does a good/bad BST look like?
- Three possible BST representations of the list



an imbalanced BST looks more like a linked list!

OUTLINE

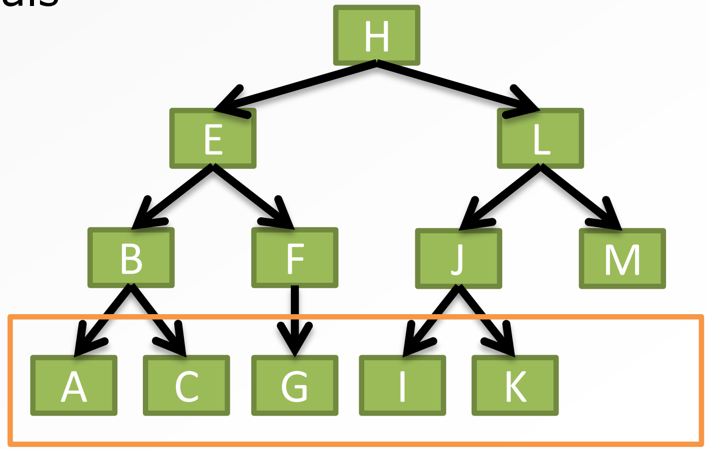
- Importance of balance for BSTs
- **Tree Balancing**

TREE BALANCING

- Goal: BST with the shortest height (short external paths, most efficient search)

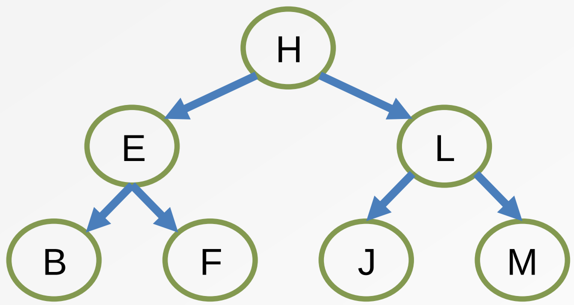
BALANCED TREES

- Ideal BST: Shortest height
  - Each tree node has exactly two child nodes except for the bottom 2 levels
  - This tree is "most balanced"
  - But, expensive to maintain this exact shape after multiple node insertions and removals



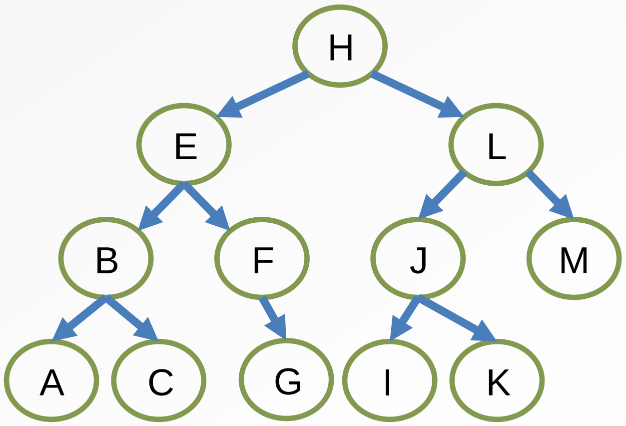
BALANCED TREES

- Ideal BST: Shortest height, $H = \lfloor \log_2 n \rfloor$
  - "perfectly balanced" tree, $n=2^{(H+1)}-1$
  - Try to fill nodes top-down, when $n< 2^{(H+1)}-1$



n= 7 **H=2**

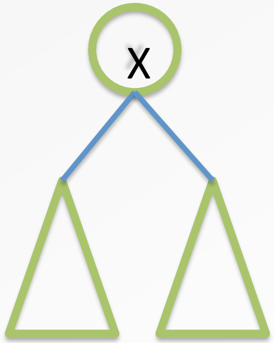
$n = 2^{2+1} -1$



n= 12 **H=3**

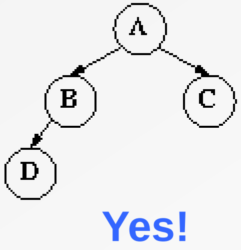
$n < 2^{3+1} -1 = 15$

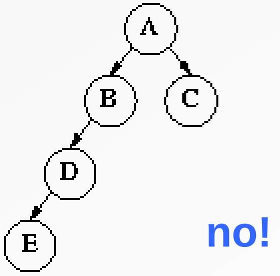
AVL BALANCED TREES

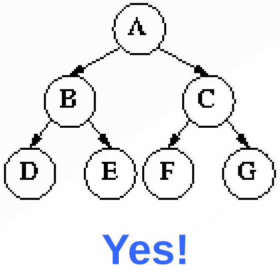


- AVL tree
  - First self-balancing binary tree invented
  - 1962: G.M. Adelson-Velskii and E.M. Landis
- Condition for every node in AVL tree:

  Heights of left vs right subtrees differ by at most 1



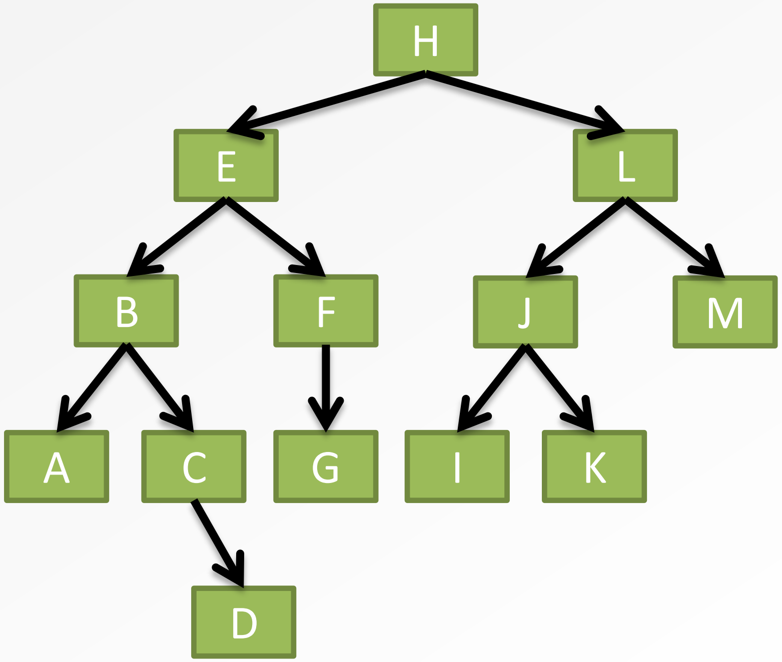




- **Height of a node** = number of links from that node to the deepest leaf node

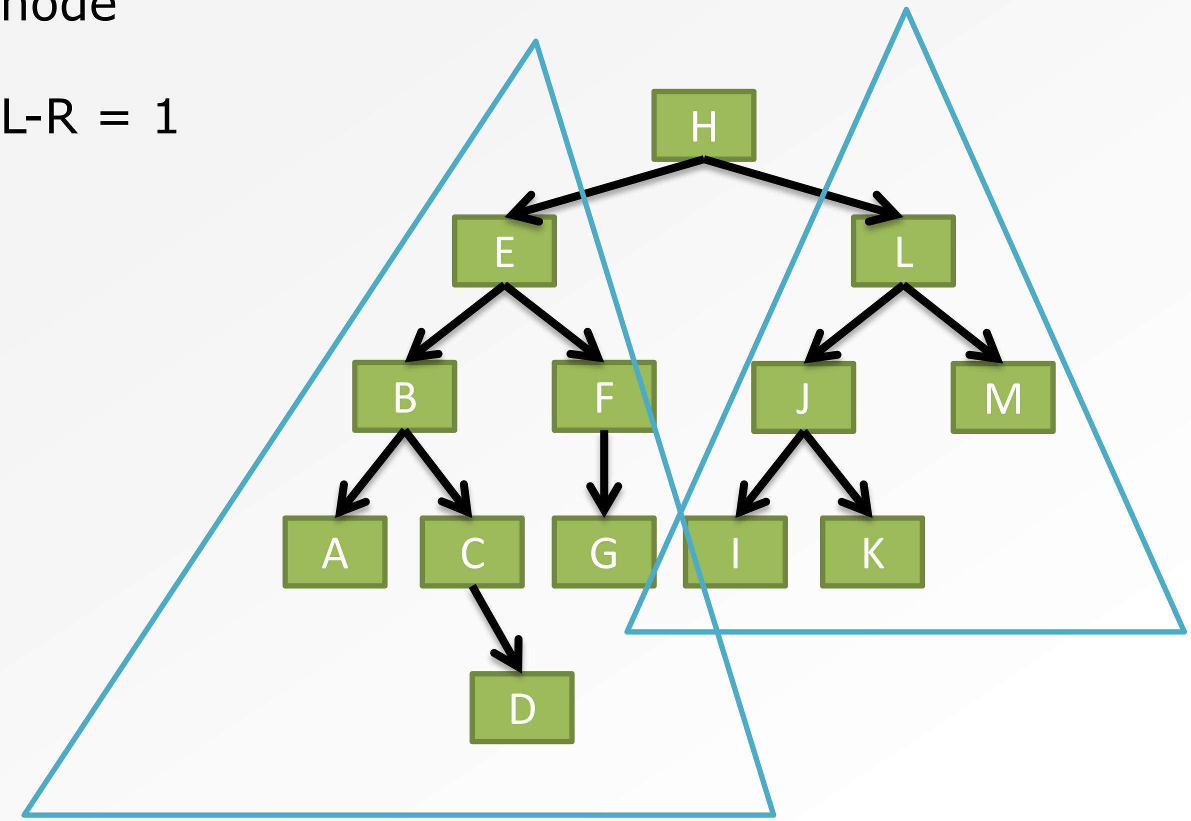
BALANCED TREES

- Check heights of the left and right subtrees at each node (L-R)



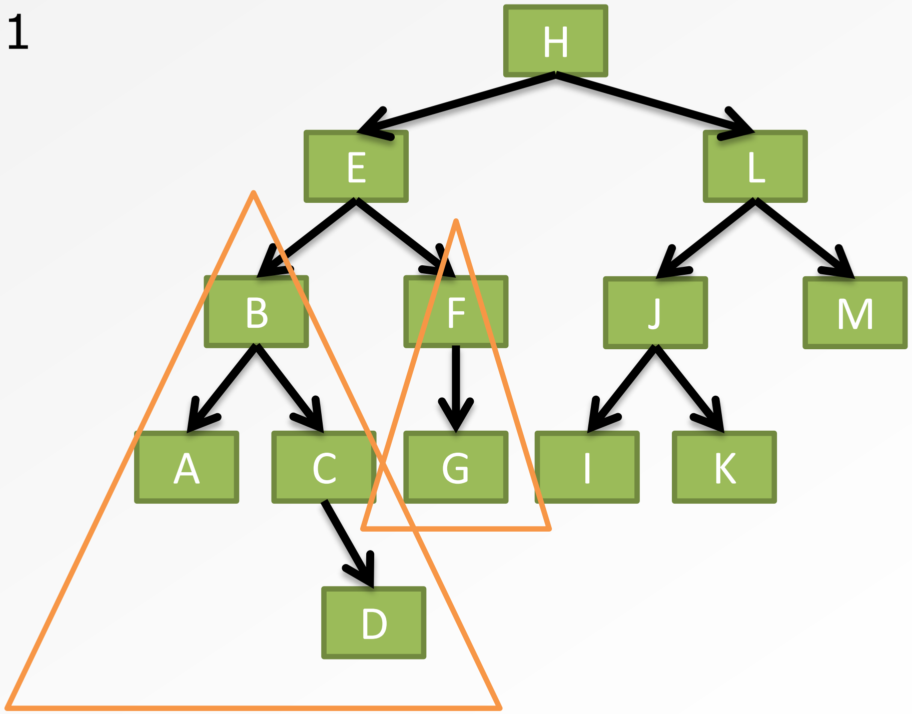
BALANCED TREES

- Check heights of the left and right subtrees at each node
- L-R = 1

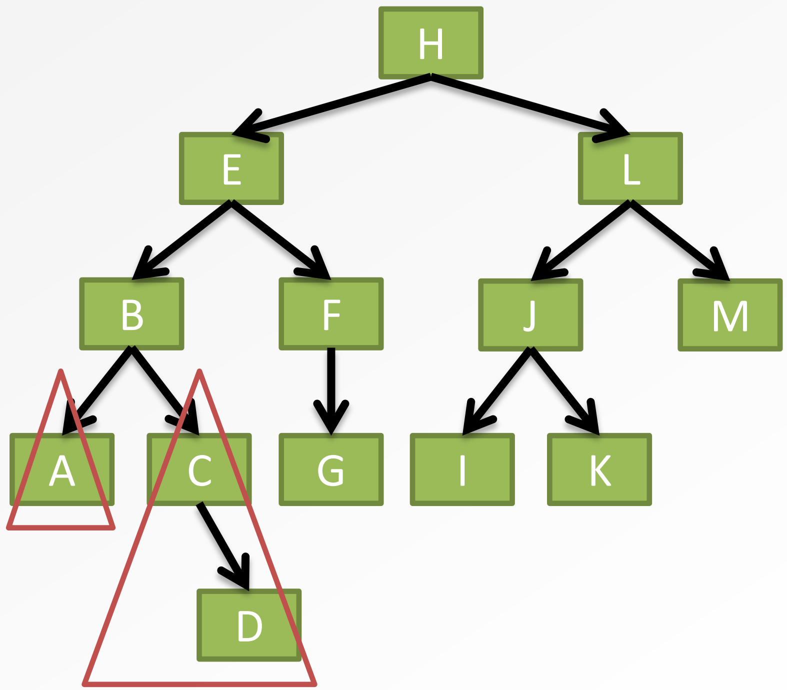


BALANCED TREES

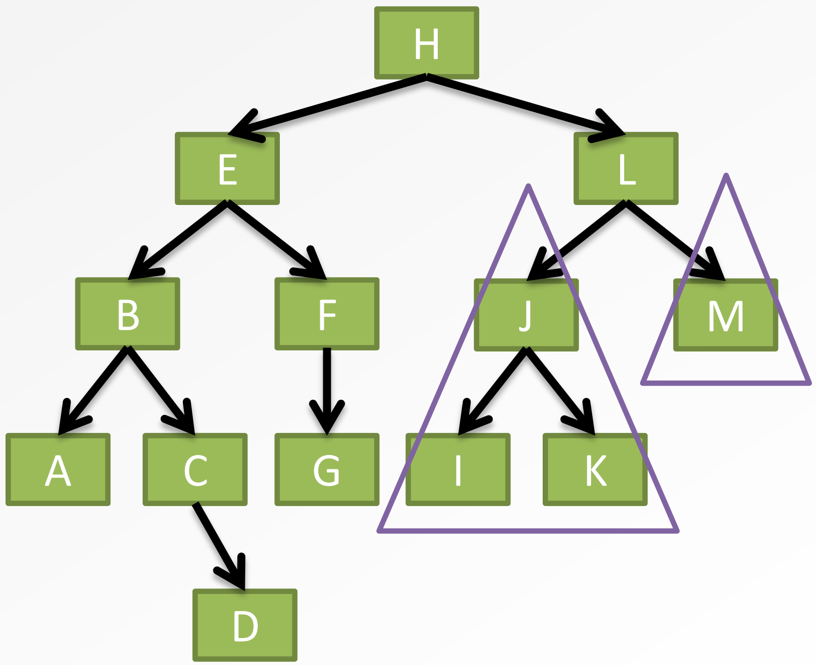
- Check heights of the left and right subtrees at each node
- L-R = 1



- Check heights of the left and right subtrees at each node
- L-R = -1

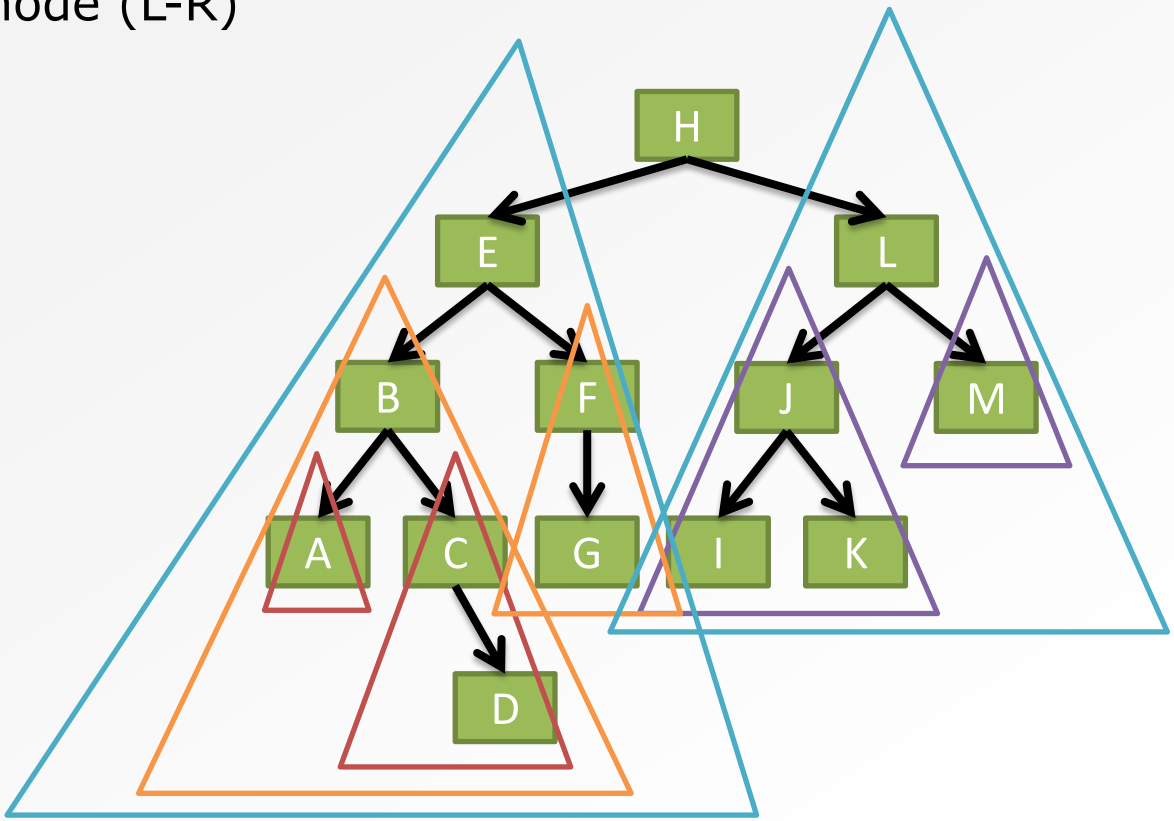


- Check heights of the left and right subtrees at each node
- L-R = 1



BALANCED TREES

- Check heights of the left and right subtrees at each node (L-R)



TREE BALANCING

- Given an imbalanced BST, we can apply some systematic sequence of operations to make it balanced: BST with the **shortest** height
- But to maintain a balanced BST after multiple node insertions and removals **is difficult/expensive**!

AVL TREES

- An AVL tree is a binary search tree with a *balance* condition.
- **AVL** is named for its inventors: **A**del'son-**V**el'skii and **L**andis
- AVL tree *approximates* the ideal tree (completely balanced tree).
- AVL Tree maintains a height close to the minimum.

Definition:

An AVL tree is a binary search tree such that for any node in the tree, the height of the left and right subtrees can differ by at most 1.