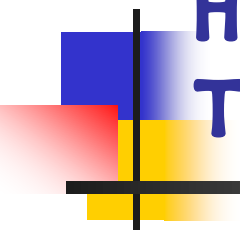
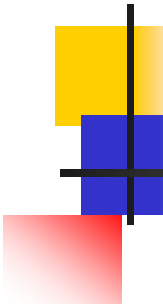


Data Structures:
Height-Balanced Search Trees:
T Tree

YoungWoon Cha
(Slide credits to Won Kim)
Spring 2022

T Tree



T Tree

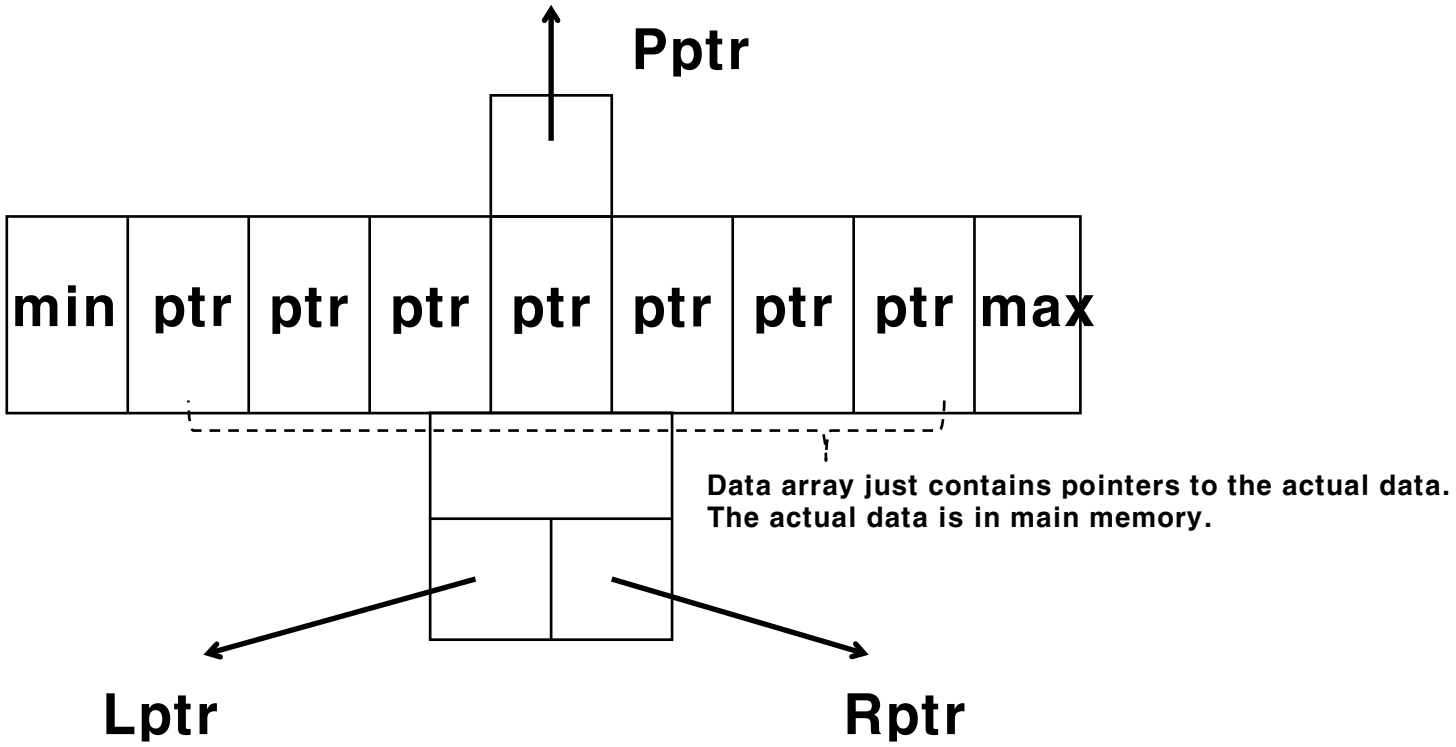
- Combination of AVL Tree and B Tree
  - (borrows from) AVL tree
    - tree rotations for height balancing
    - not perfectly balanced
  - (borrows from) B tree
    - N to 2N data in each node
- Important in main-memory database systems
  - Oracle, MySQL,…
- Reading
  - https://en.wikipedia.org/wiki/T-tree

Each T Tree Node (Implementation)

- An array of N to 2N data
  - (pointers to data in main memory)
  - “2N” is fixed at tree-creation time.
  - Underflow: < N (root node is an exception)
  - Overflow: > 2N
- Pointers to left subtree and right subtree
- Pointer to the parent
- Some control data

Visualization of Each T Tree Node

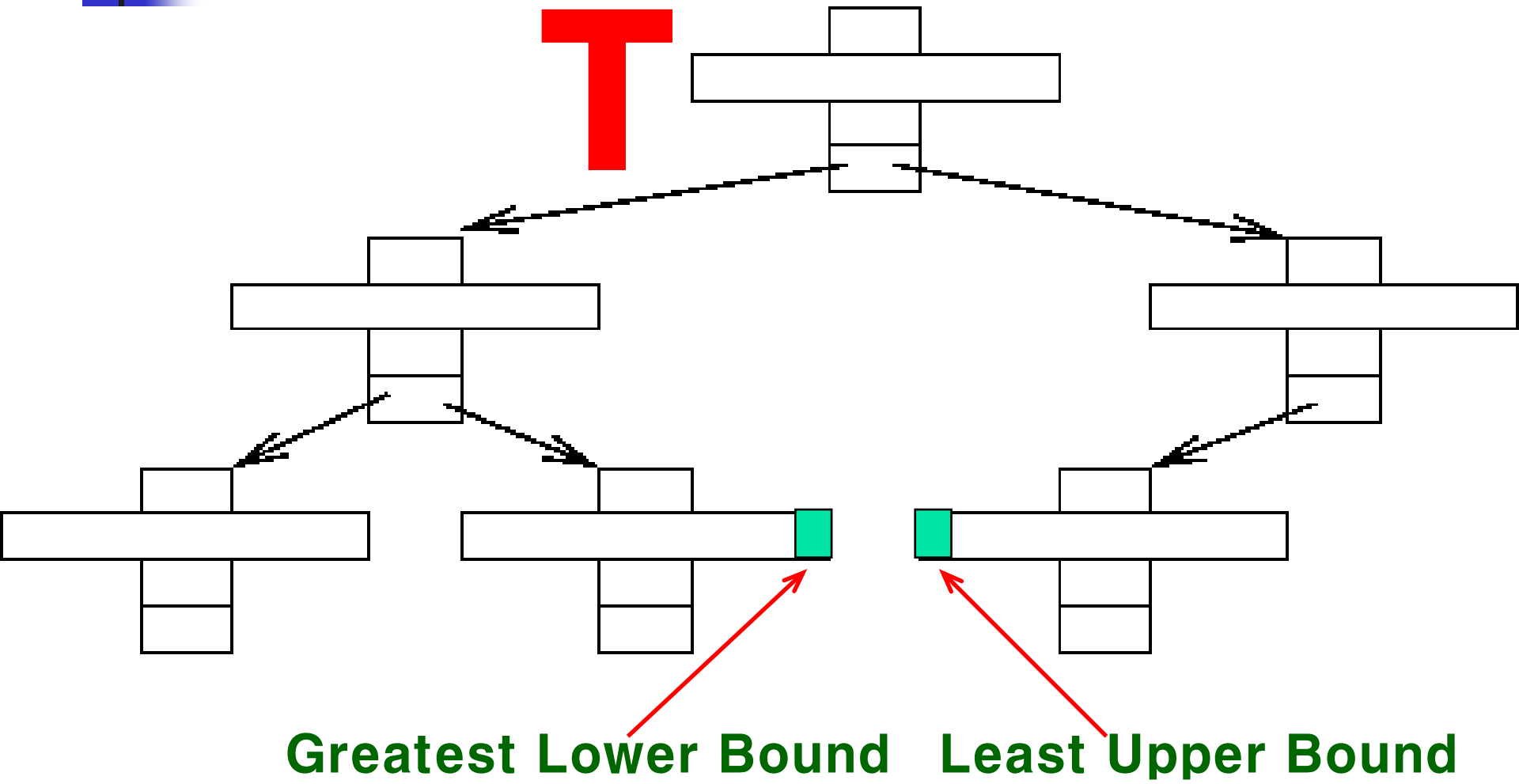
Node



T Tree Organization

- Root Node
- Interior Nodes
  - 2 subtrees
- Half-Leaf Nodes
  - 1 subtree
- Leaf Nodes
  - 0 subtree

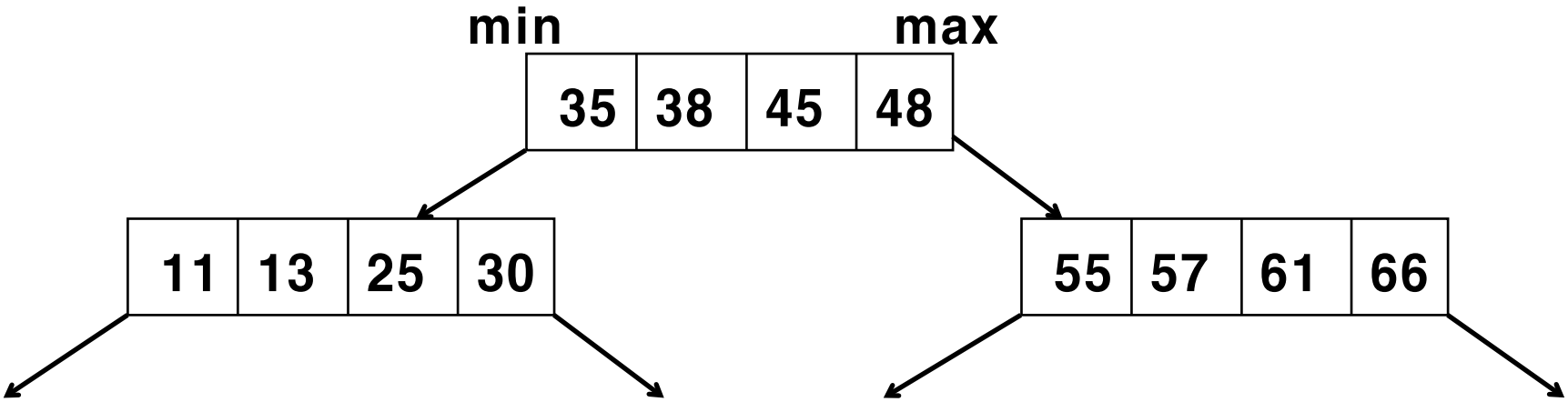
Visualization of a T Tree



Searching

- Search key X, starting at the root node.
- If X < the MIN of the node, search the left subtree.
- If X > the MAX of the node, search the right subtree.
- Otherwise, search the data array on the node.

Example



Leaf Node Overflow

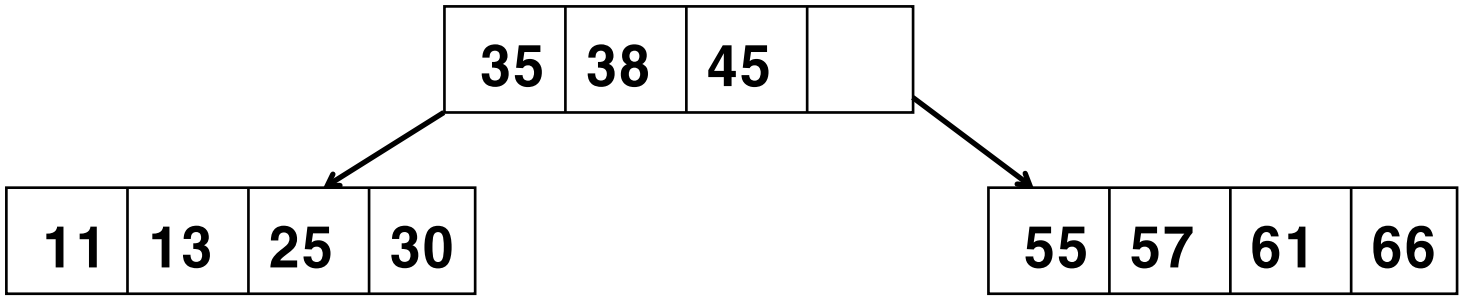
- If a leaf node has 2N + 1 data, split the node.
- Move Left
  - keep the largest N+1 data in the current node, and
  - move the smallest N data to a new left child leaf node
- Move Right
  - keep the smallest N+1 data in the current node, and
  - move the largest N data to a new right child leaf node

Inserting

- Search key X, starting at the root node.
- If X is found, finish.
- Insert in a node where search fails.
- If there is room, insert X. Finish.
- If there is no room,
- Either (move left)
  - if the current node is a leaf node, split the node. Finish.
  - else (remove the smallest data, and insert x in the node.
  - insert the removed data into the “greatest lower bound”
  - leaf node. If the leaf node overflows, split the leaf node. Finish.)
- Or (move right)
  - if the current node is a leaf node, split the node. Finish.
  - else (remove the largest data, and insert x in the node.
  - insert the removed data into the “least upper bound”
  - leaf node. If the leaf node overflows, split the leaf node. Finish.)
- If the tree is out of balance, perform tree rotations. Finish.

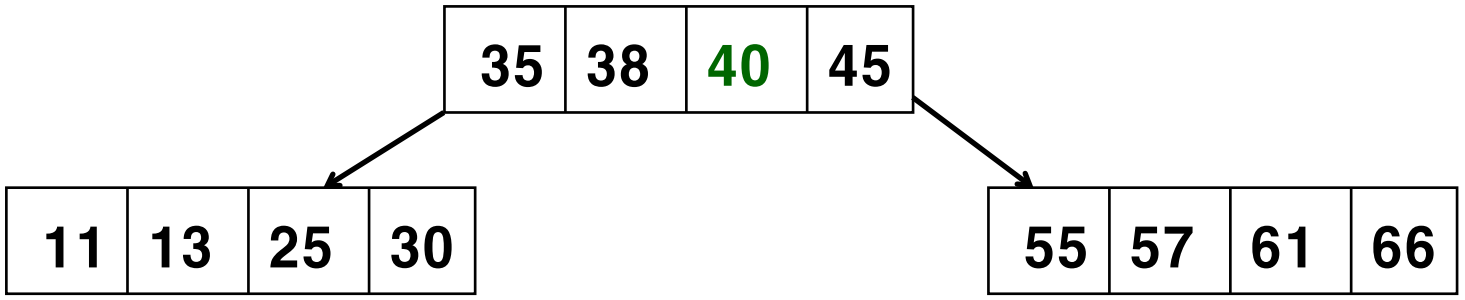
Example 1

insert 40



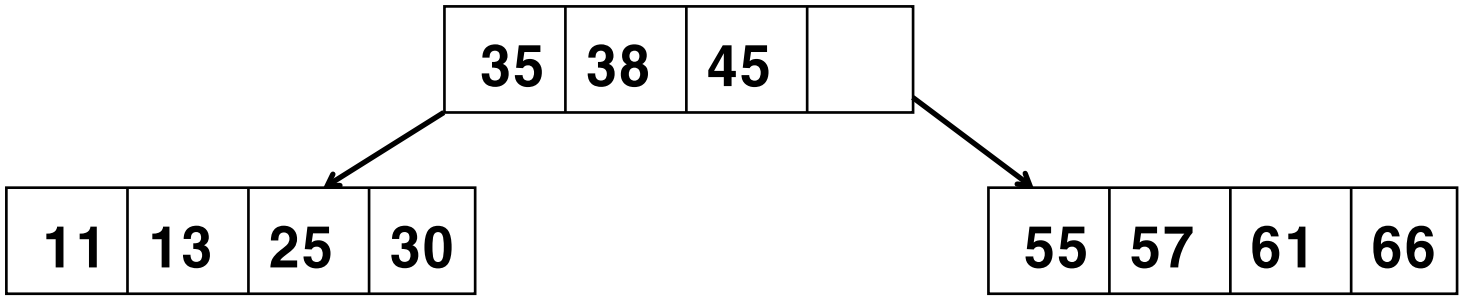
Example 1 (cont'd)

insert 40



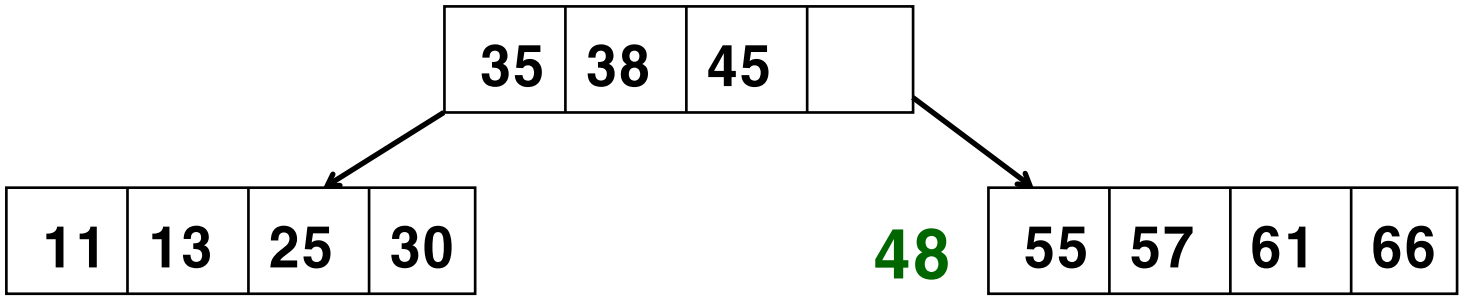
Example 2 (leaf node overflow)

insert 48



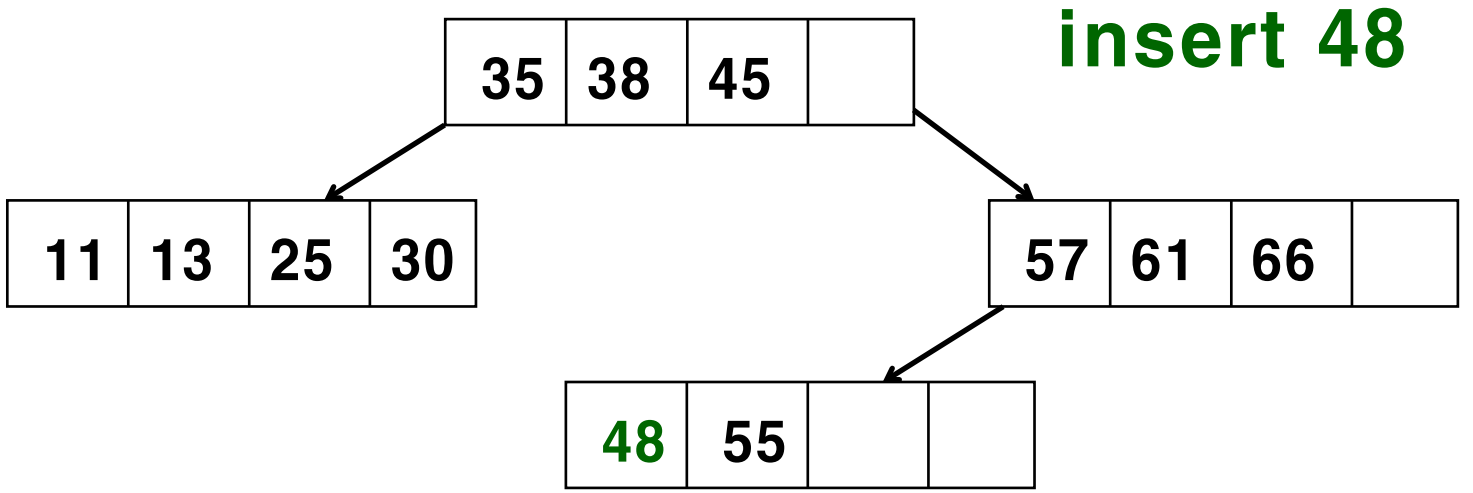
Example 2 (cont'd)

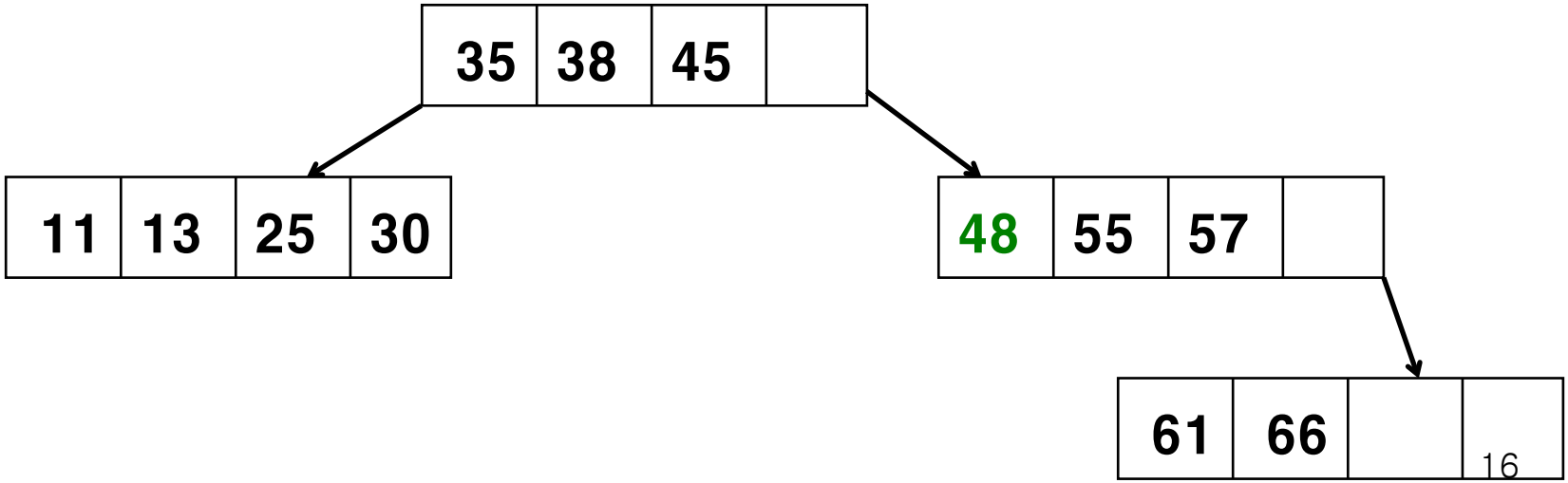
insert 48



The search failed here!

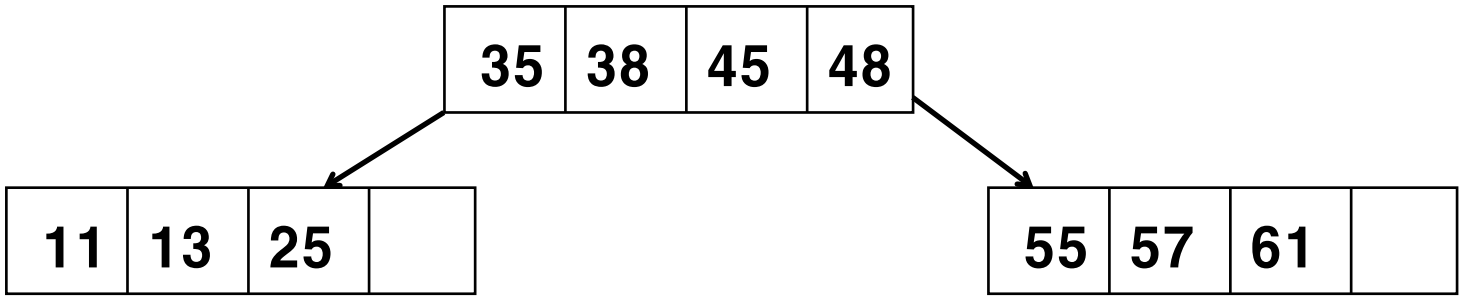
Example 2 (cont'd)



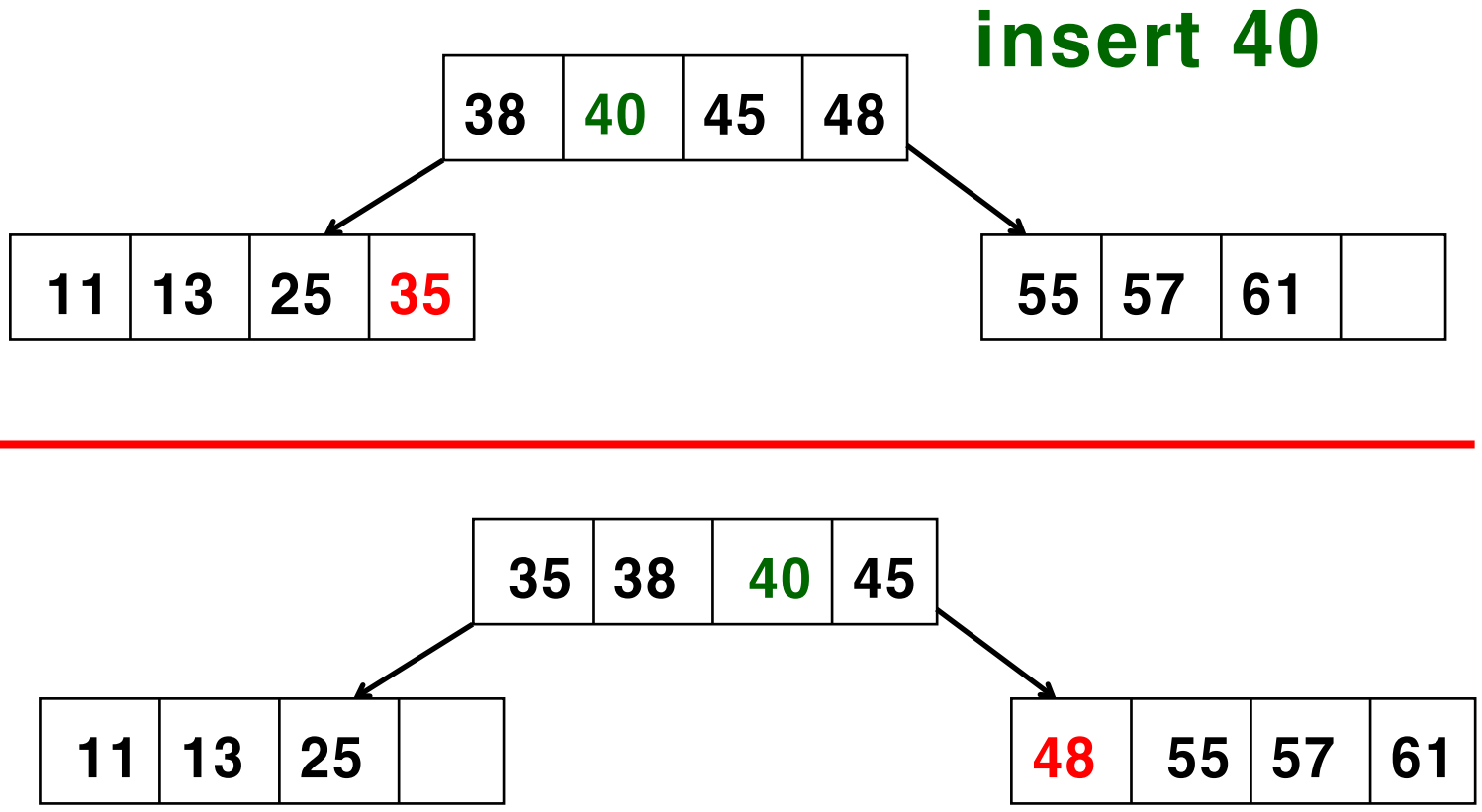


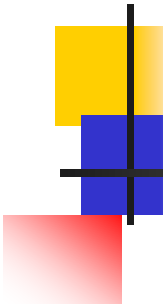
Example 3 (non-leaf node overflow)

insert 40



Example 3 (cont'd)





Leaf Node Underflow

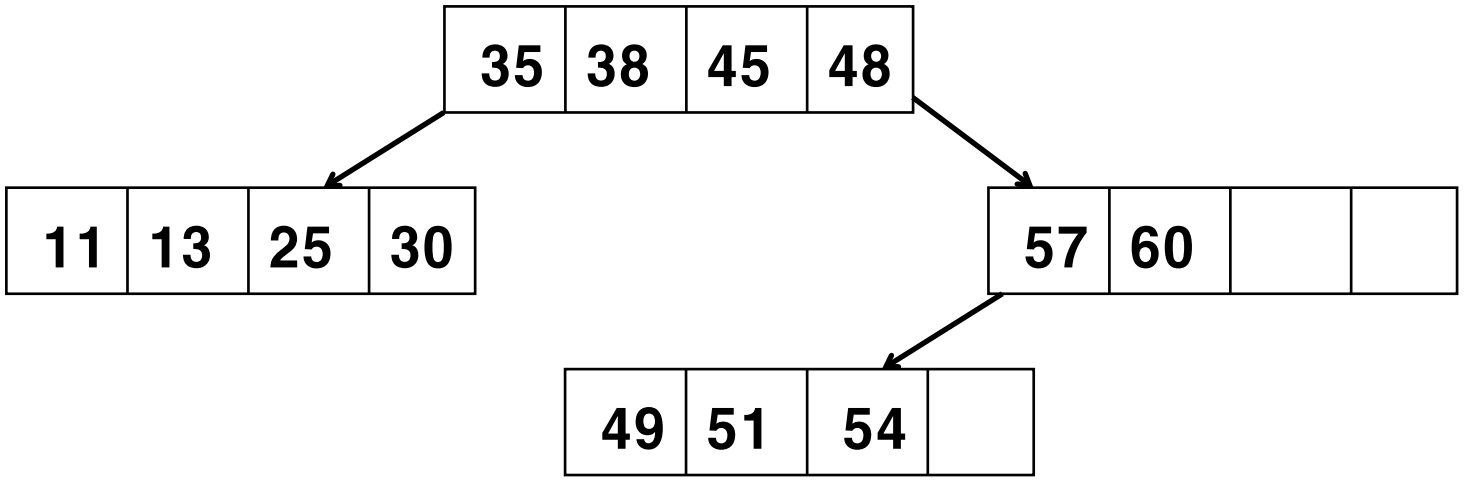
- If a leaf node has N-1 data, merge the node with its parent node.
- If the parent node overflows as a result, split the parent node.

Deleting

- Search key X, starting at the root node.
- If X is not found, finish.
- If X is found, delete it.
  - If X was in a leaf node, and the node underflows, merge the node with the parent node.
  - If X was in an interior node and the node underflows, replace X with the largest data from the left subtree, or the smallest data from the right subtree.
  - If the tree becomes out of balance (the balance factor of any node becomes +2 or -2), perform tree rotations.

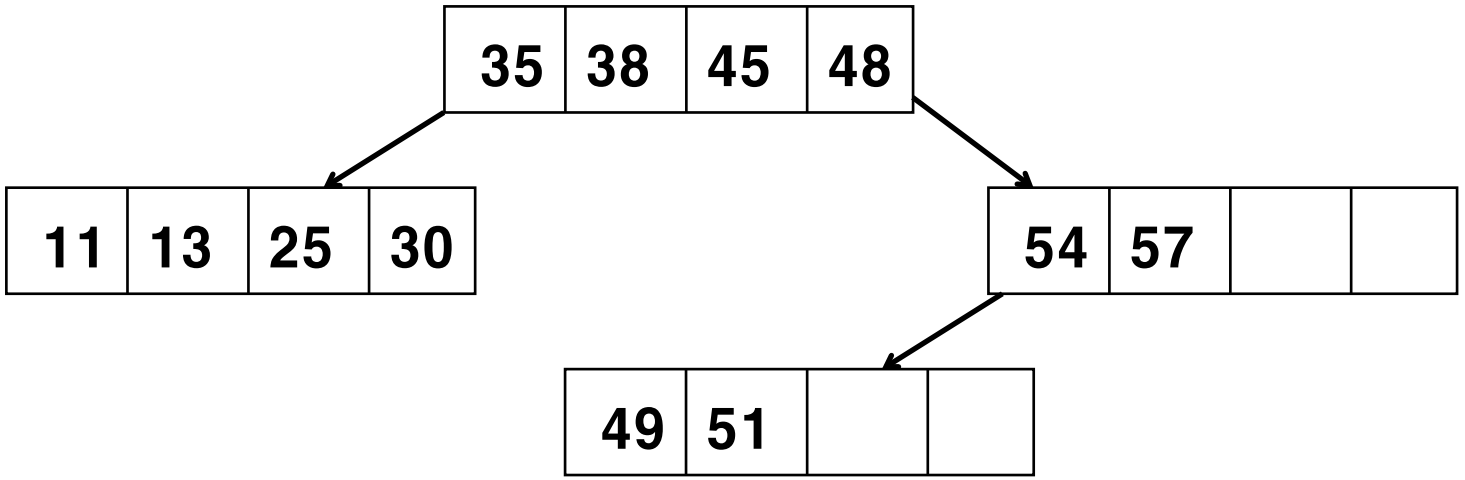
Example 1 (interior node underflow)

delete 60



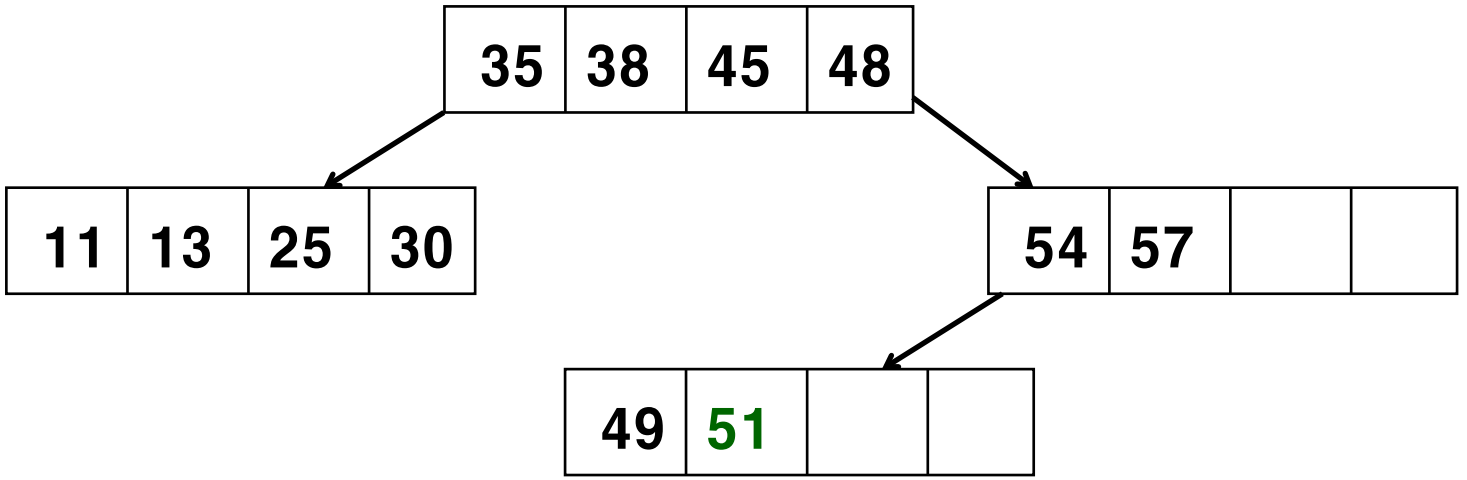
Example 1 (cont'd)

delete 60



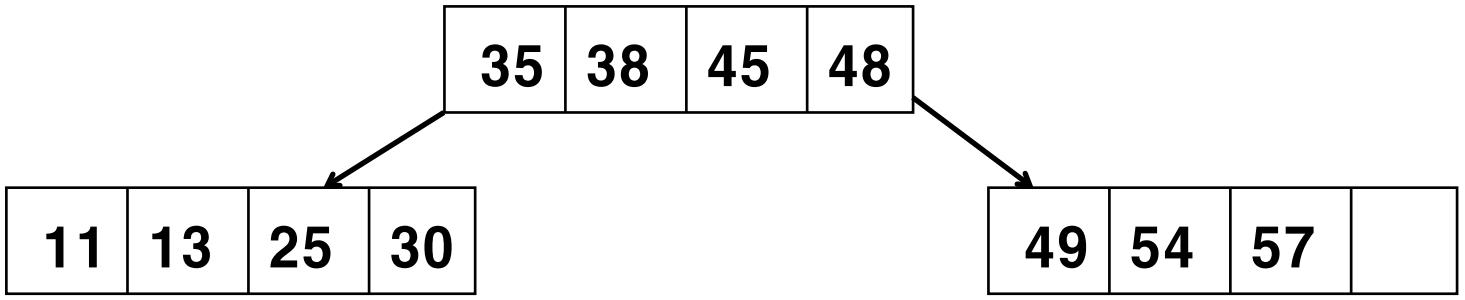
Example 2 (leaf node underflow)

delete 51



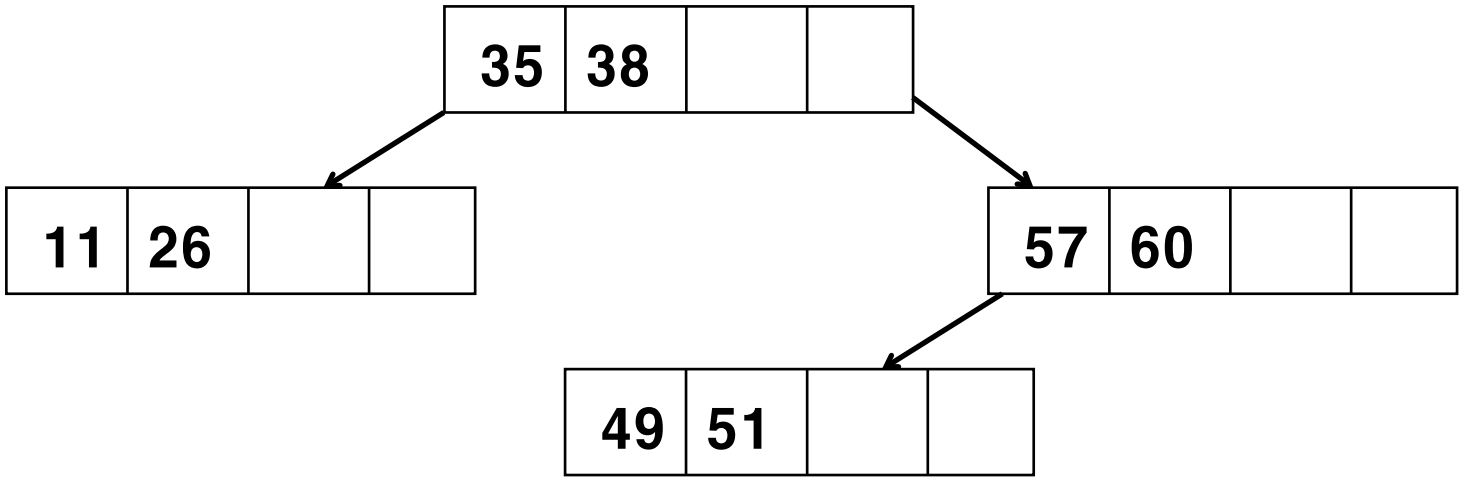
Example 2 (cont'd)

delete 51

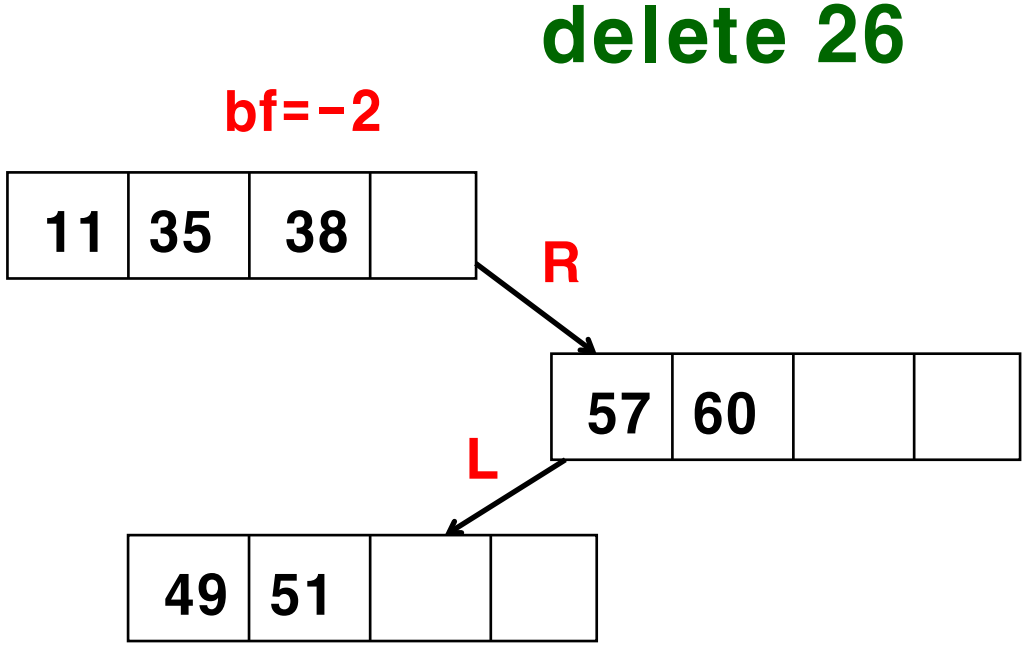


Example 3: Tree Rotation

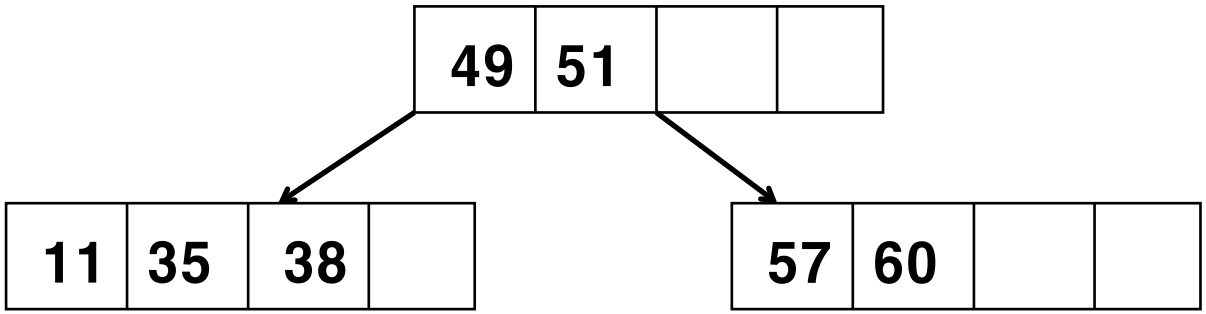
delete 26



Example 3: (cont'd)

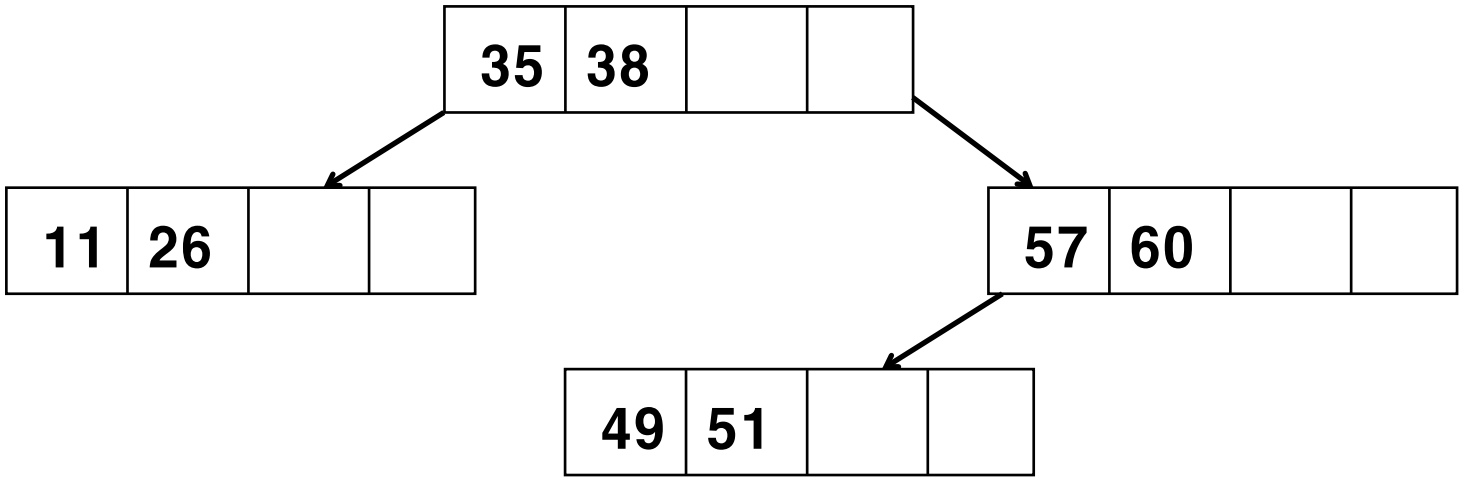


Example 3: (cont'd) RL Rotation

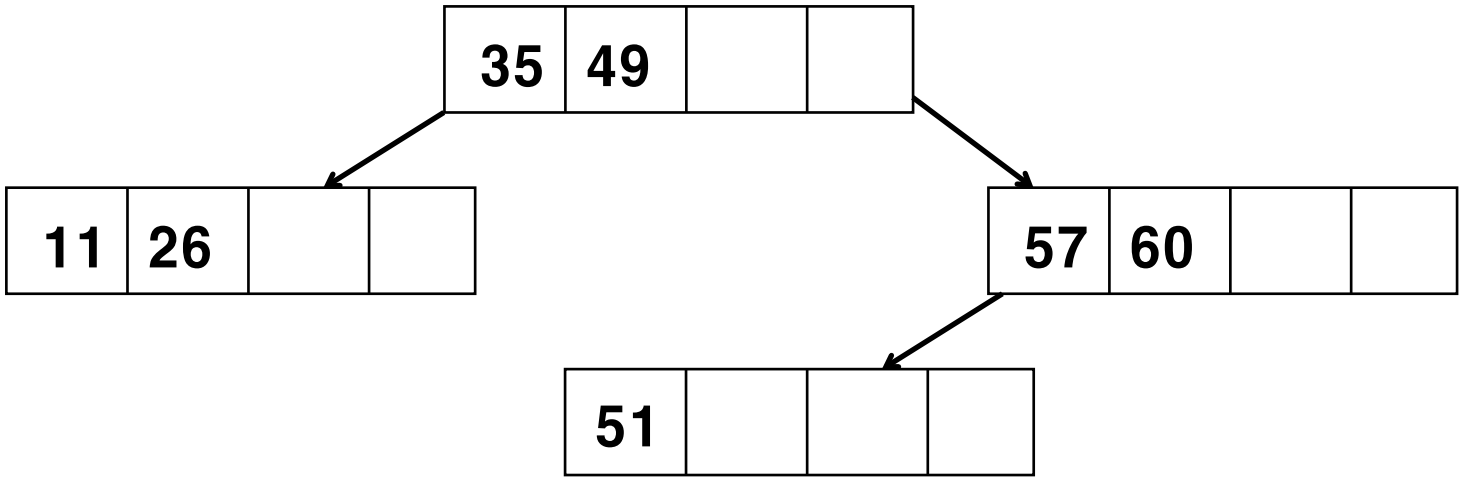


Exercise: Delete a Key From a T Tree

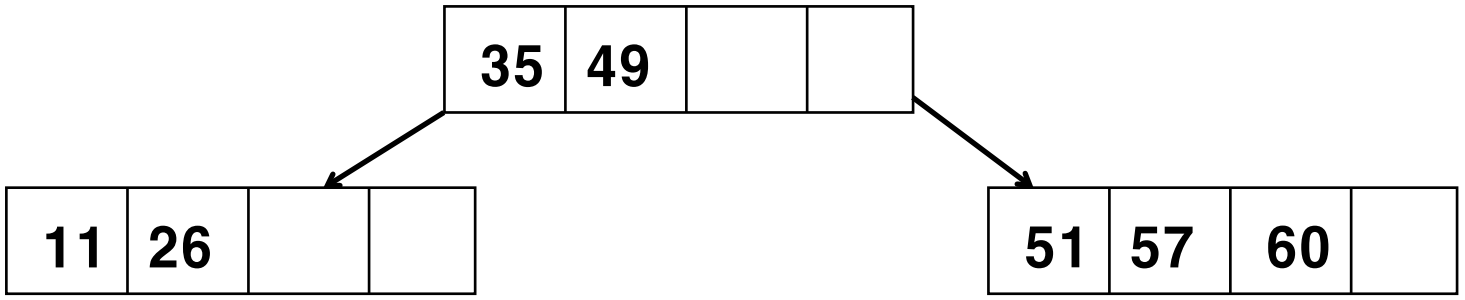
delete 38



Exercise: Delete a Key From a T Tree



Exercise: Delete a Key From a T Tree



Performance Properties of a T Tree

- Reduced tree height
  - $\log_2$ [N/M]
  - N = total number of keys, M = number of keys per node
- Node split and merge
- The usual problems of the array for the keys in each node
- Maintaining Min, Max key values in each node

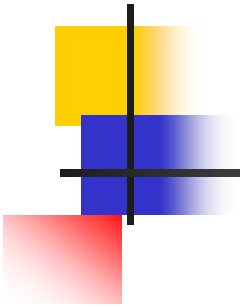
Assignment 7

HW 7-1

- Construct a 2-3 Tree with keys D, A1, T1, A2, S, T2, R1, U1, C, T3, U2, R2, E
  - in the given order, starting from an empty tree.
  - (you must show each insert and each node split)
- From the constructed 2-3 Tree, delete the nodes with keys A1, T1, T2, T3
  - in the given order.
  - (you must show each delete and each node merge)
- The keys are aligned in alphabetical order.

HW 7-2

- Construct a T-Tree (where M=2) with keys 20, 80, 60, 40, 15, 25, 30, 35
  - in the given order, starting from an empty tree.
  - (you must show each insert; node split and tree rotation)
- From the constructed T-Tree, delete the nodes with keys 20, 35, 60, 80
  - in the given order.
  - (you must show each delete; node merge and tree rotation)
- The keys are aligned in alphabetical order.



End of Lecture