B+, 2-3, Huffman Trees

Kuan-Yu Chen (陳冠宇)

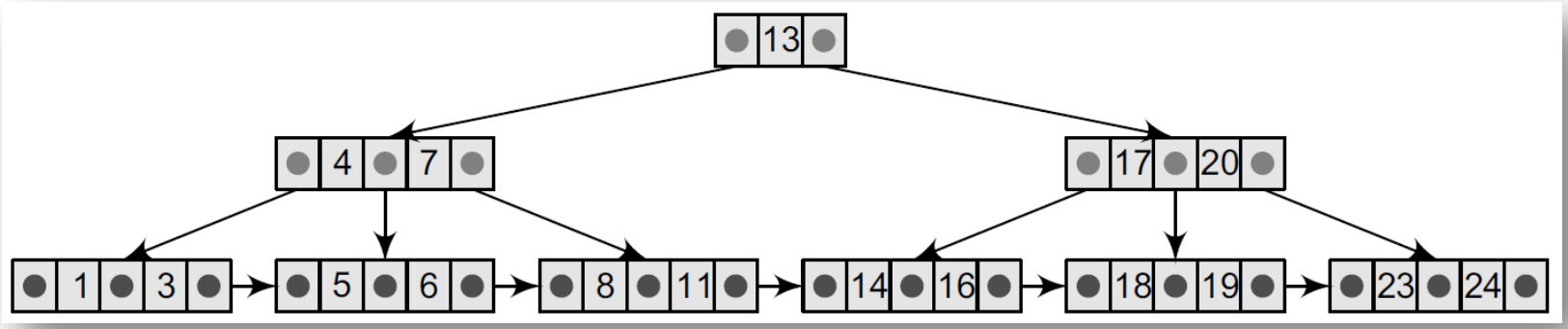
2020/11/04 @ TR-212, NTUST

Review

- A B tree of order m is a tree with all the properties of an M-way search tree and has additional properties
  - Every node in the B tree except the root node and leaf nodes has at least (minimum) $\left\lceil \frac{m}{2} \right\rceil$ children
    - Degree=4, at least 2 children, at least 1 key
    - Degree=5, at least 3 children, at least 2 key
  - The root node has at least two children
  - Each leaf node has at least $\left\lceil \frac{m}{2} \right\rceil$ NULL pointers

B+ Trees

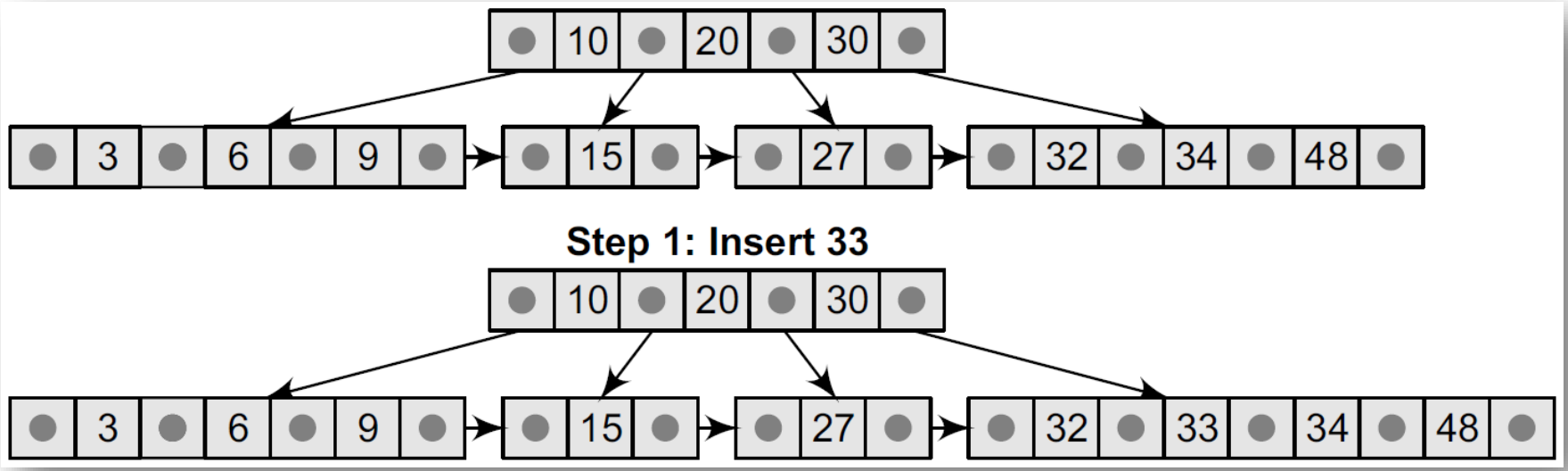
- A B+ tree is a variant of a B tree which stores sorted data in a way that allows for efficient insertion, retrieval, and removal of records, each of which is identified by a key
  - A B tree can store both keys and records in its interior nodes
  - A B+ tree stores all the records at the leaf level of the tree and keys are stored in the interior nodes
    - Typically, B+ trees are used to store large amounts of data that cannot be stored in the main memory
    - The leaf nodes of a B+ tree are often linked to one another in a linked list
    - All of the internal nodes are called index nodes or i-nodes



B+ Trees – Insertion.

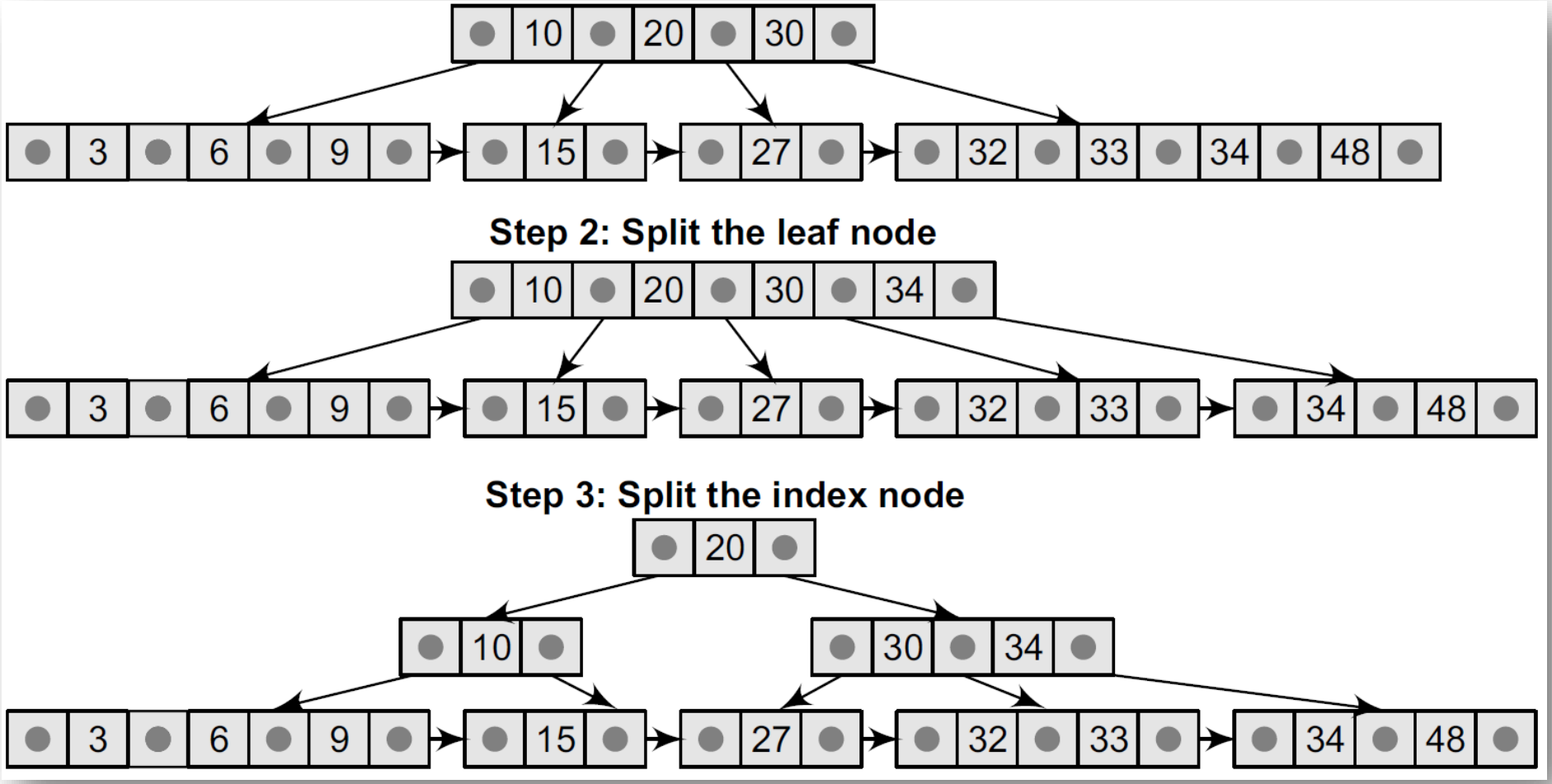
- For inserting a new element in a B+ tree
  - A new element is simply added in the leaf node if there is space for it
  - If the data node in the tree is full, then that node is split into two nodes

- For a given B+ tree of order 4, please insert 33 in the tree



B+ Trees – Insertion..

- For a given B+ tree of order 4, please insert 33 in the tree

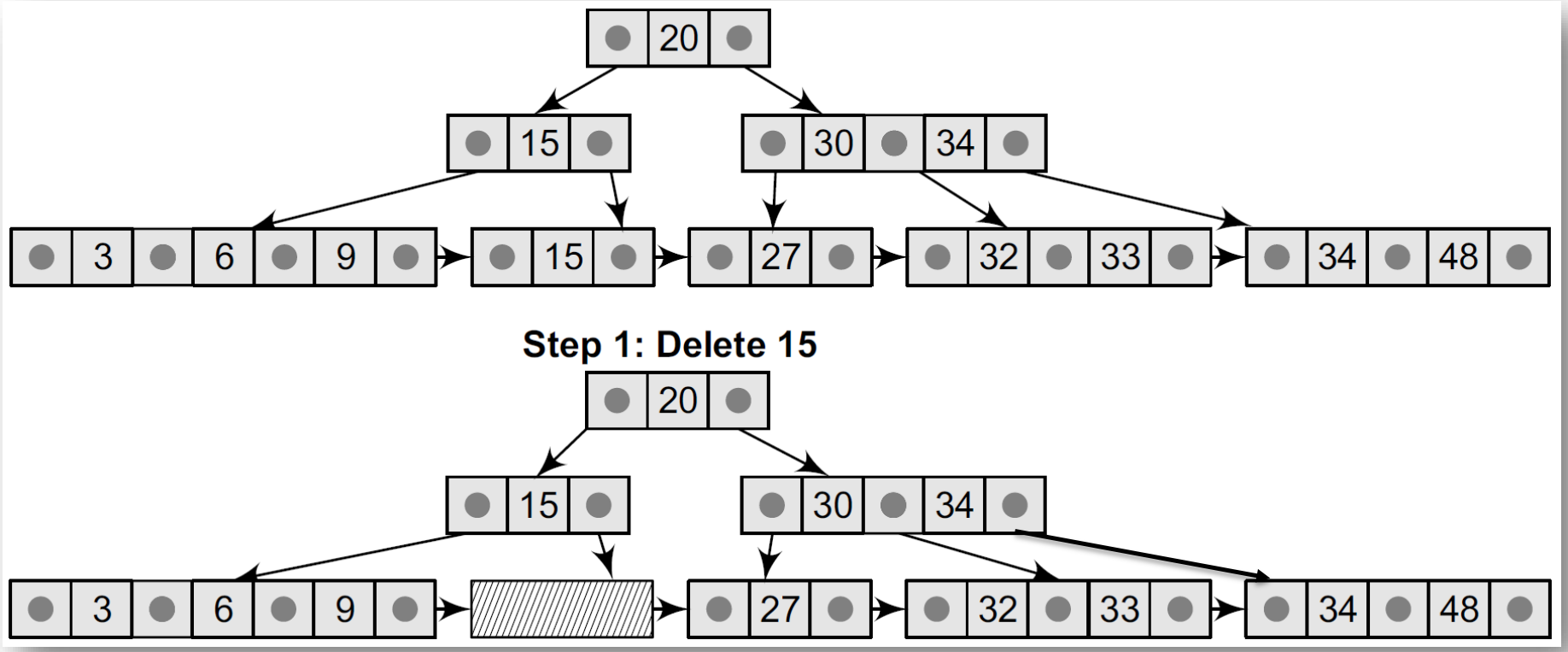


B+ Trees – Deletion.

- For a B+ tree, deletion is always done from a leaf node
  1. Delete the key and data from the leaves
  2. If the **leaf node underflows**, merge that node with the sibling and **delete** the key in between them
  3. If the **index node underflows**, merge that node with the sibling and **move down** the key in between them

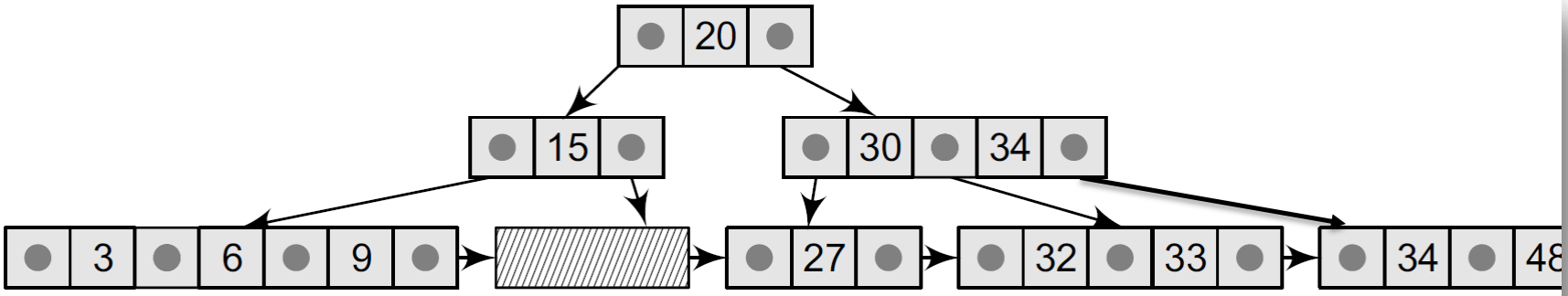
B+ Trees – Deletion..

- For a B+ tree of order 4, please delete node 15 from the tree

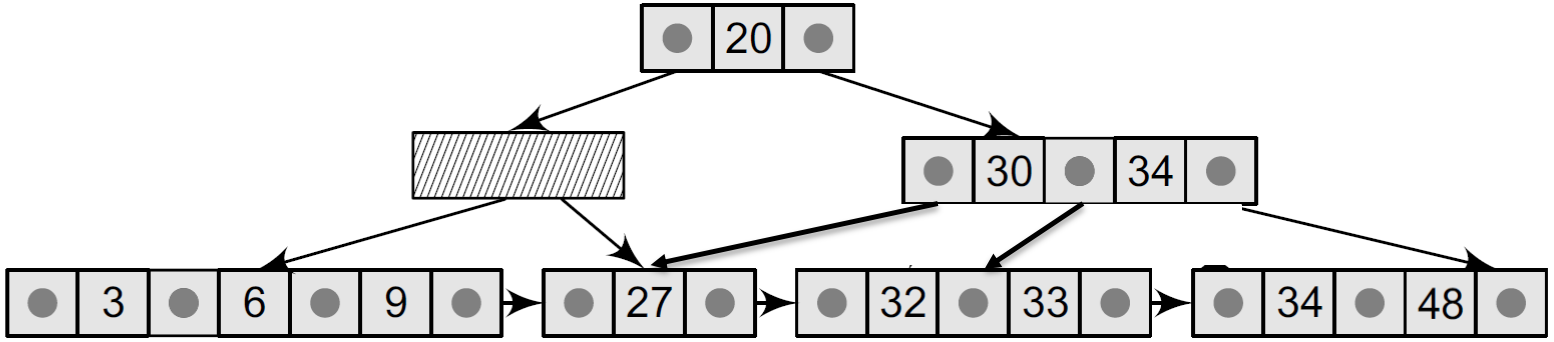


B+ Trees – Deletion…

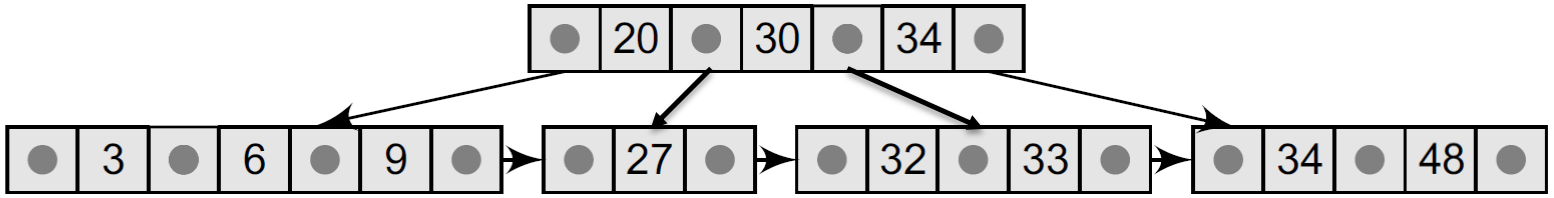
- For a B+ tree of order 4, please delete node 15 from the tree



Step 2: Leaf node underflows so merge with left sibling and remove key 15

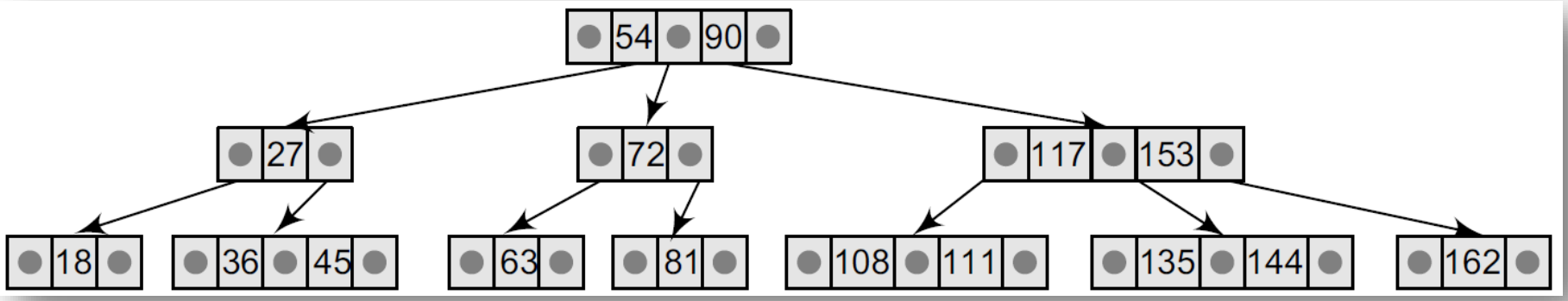


Step 3: Now index node underflows, so merge with sibling and delete the node



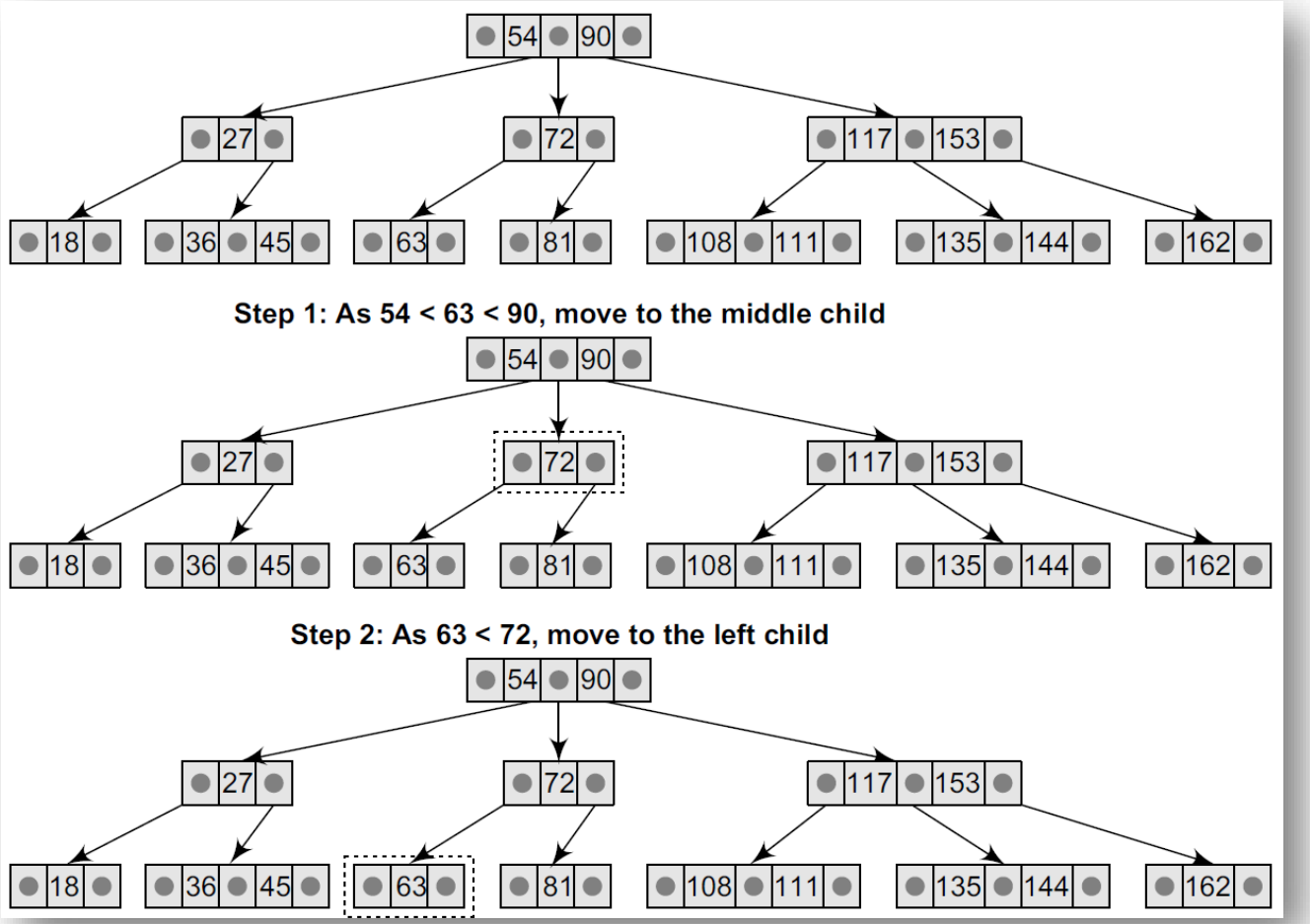
2-3 Trees

- In a 2-3 tree, proposed by John Hopcroft in 1970, each interior node has either two or three children
  - **2-3 tree is a B-tree of order 3**
  - Nodes with two children are called 2-nodes
    - The 2-nodes have one data value and two children
  - Nodes with three children are called 3-nodes
    - The 3-nodes have two data values and three children
  - All the leaf nodes are at the same level



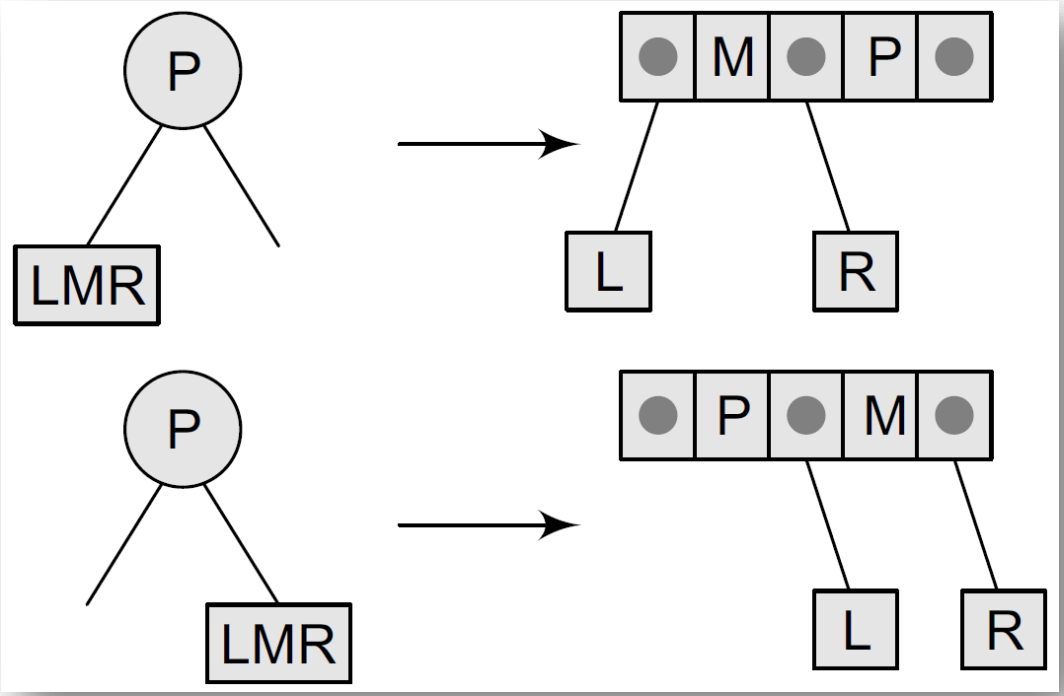
2-3 Trees – Searching

- The search operation is used to determine whether a data value is present in a 2-3 tree
  - Search for 63



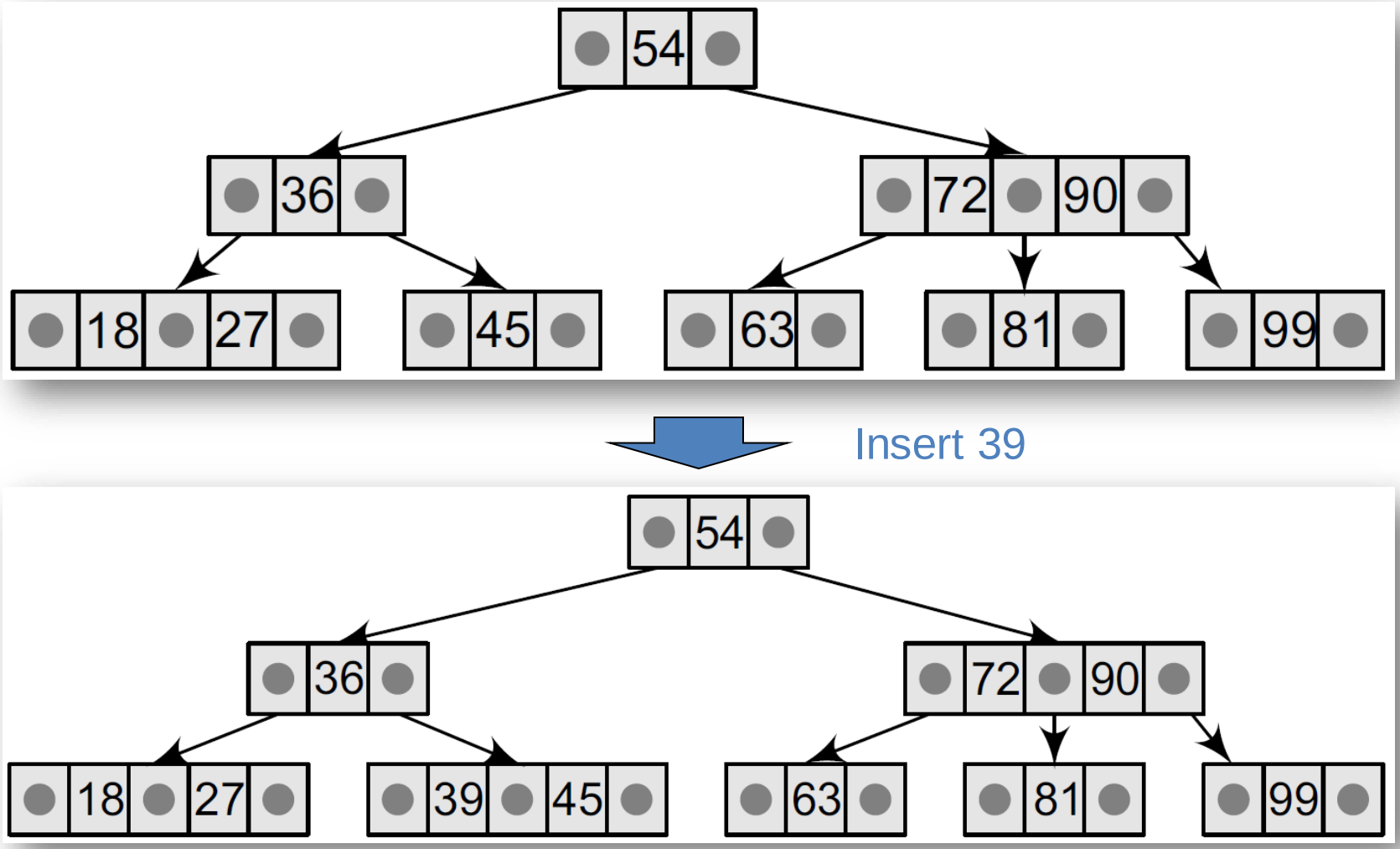
2-3 Trees – Insertion.

- To insert a new value in the 2-3 tree, an appropriate position of the value is located in one of the leaf nodes
  - If after insertion of the new value, the properties of the 2-3 tree do not get violated then insertion is over
  - If any property is violated then the violating node must be split
    - A node is split when it has three data values and four children



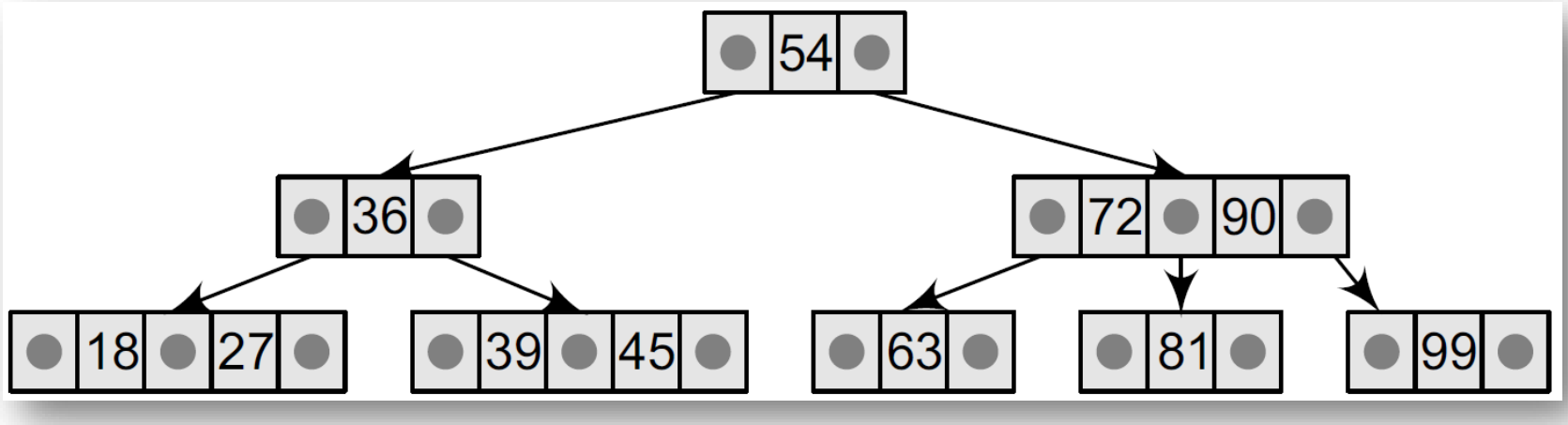
2-3 Trees – Insertion..

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree



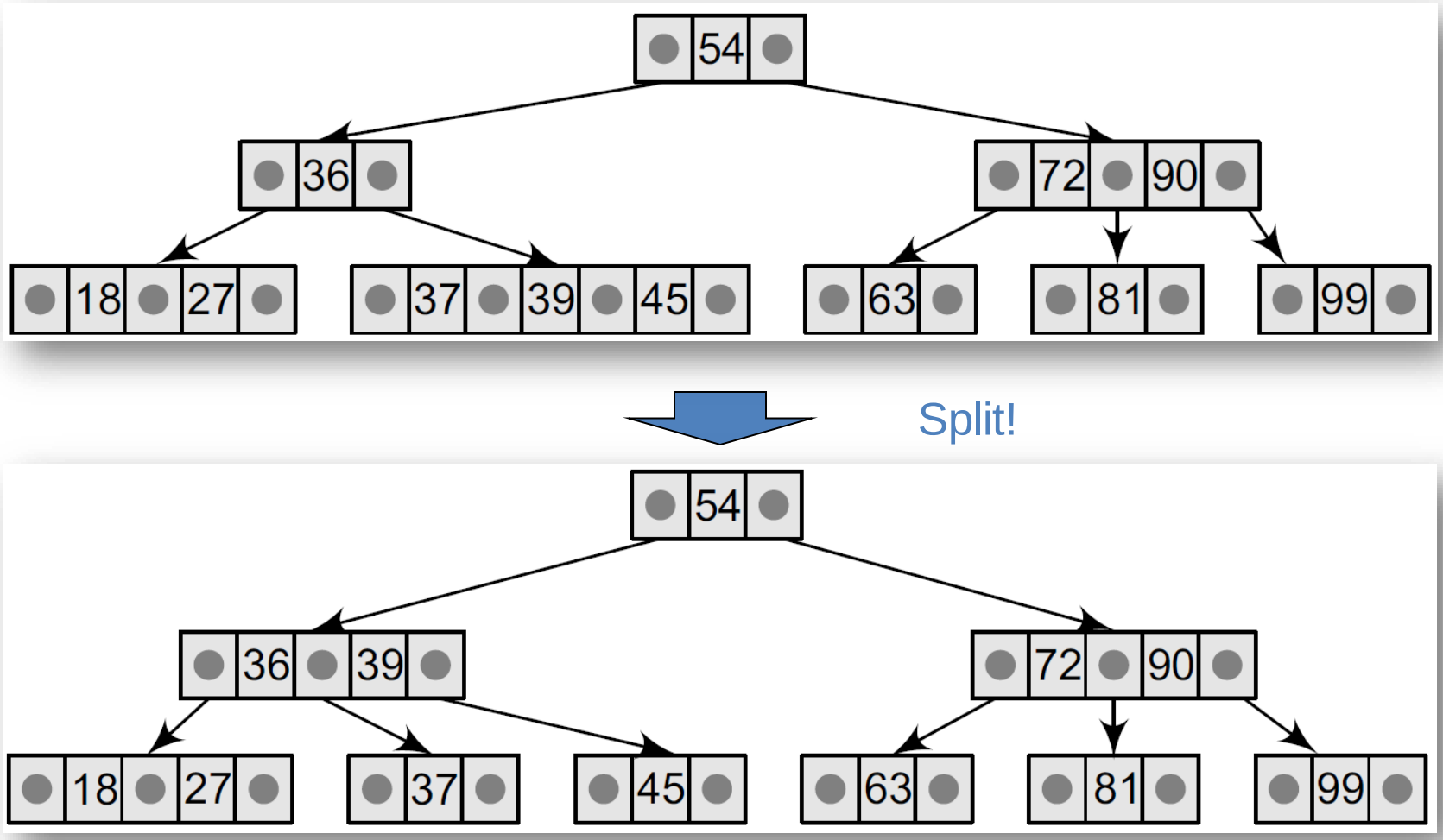
2-3 Trees – Insertion…

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree



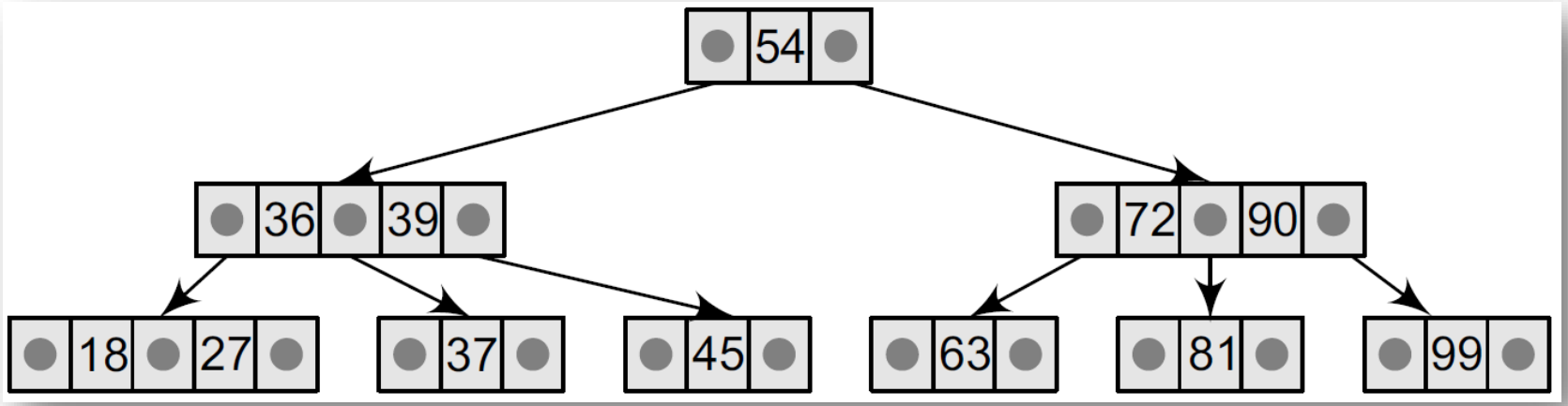
2-3 Trees – Insertion….

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree

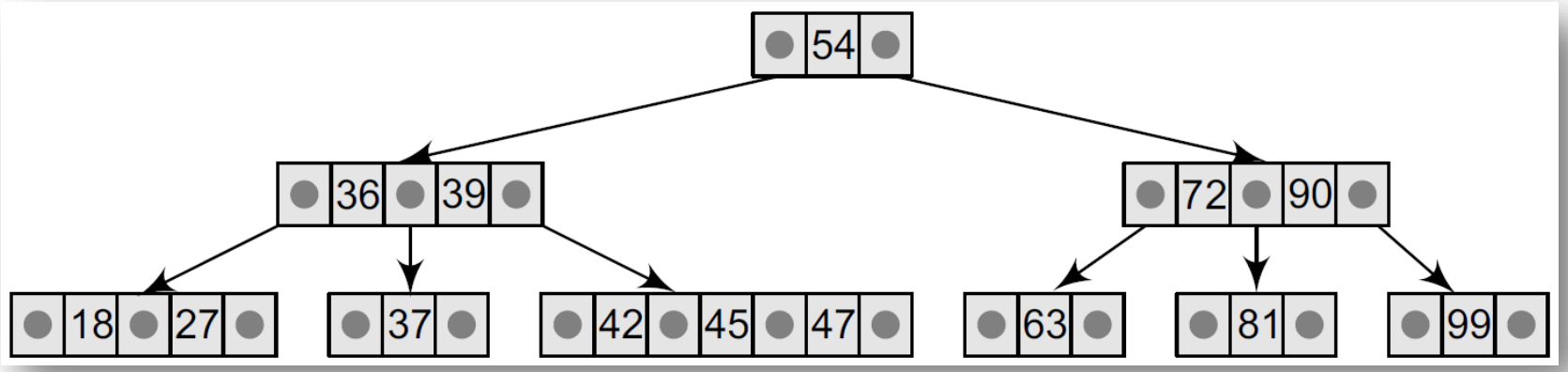


2-3 Trees – Insertion…..

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree

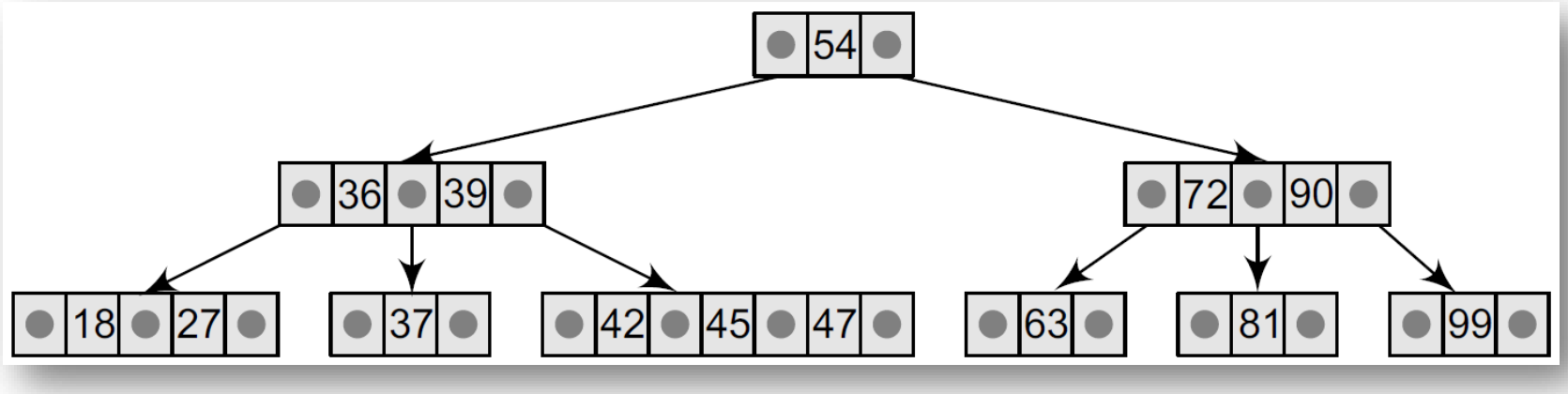


Insert 42 & 47

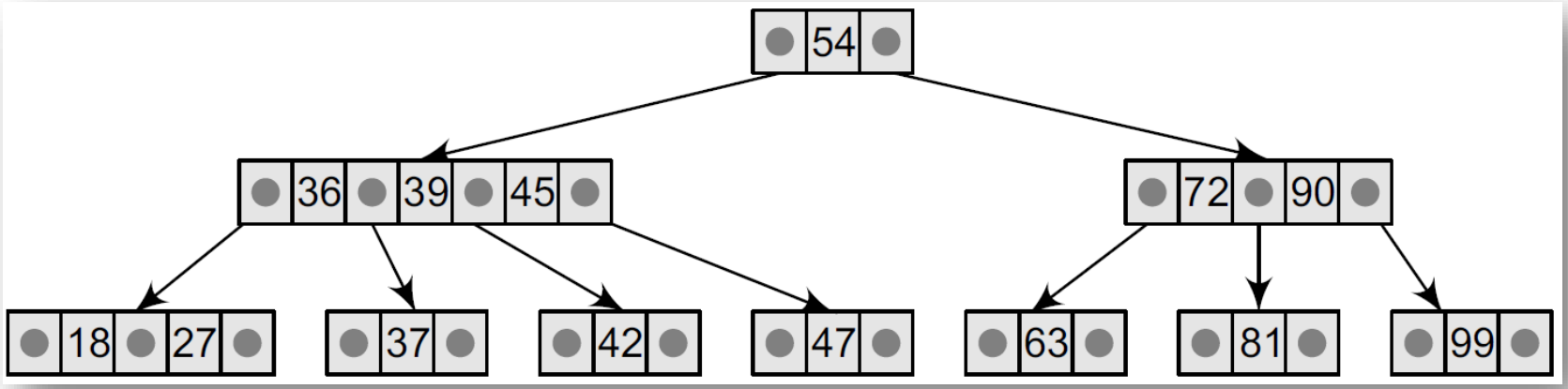


2-3 Trees – Insertion…..

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree

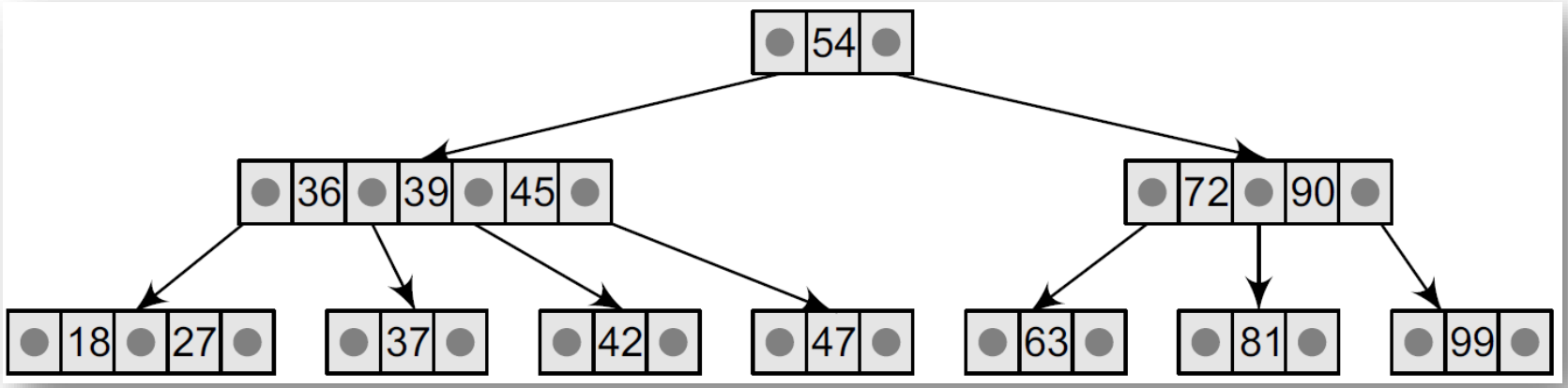


Split!

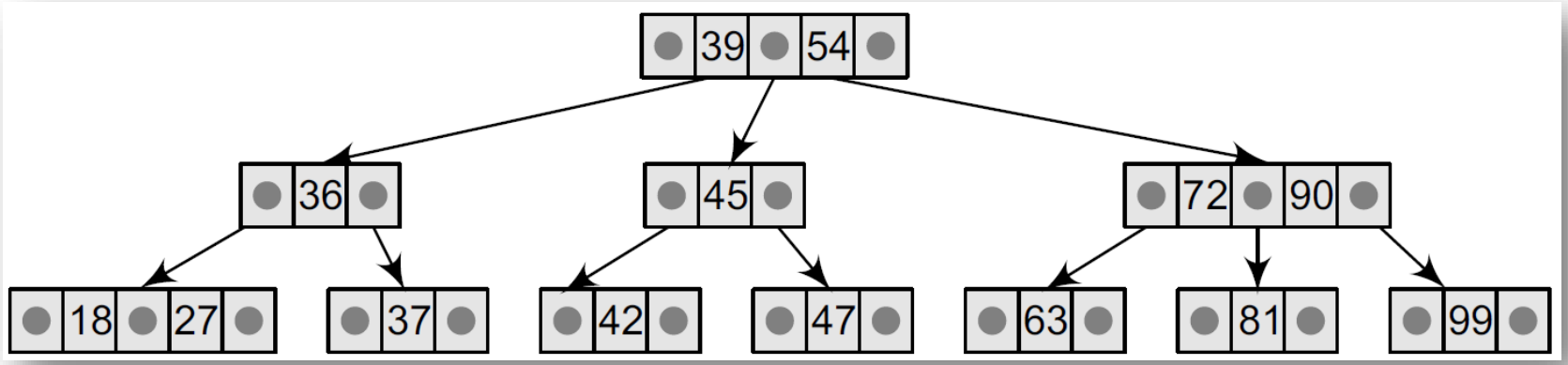


2-3 Trees – Insertion……

- Given a 2-3 tree, please insert the data values 39, 37, 42, 47 into the tree



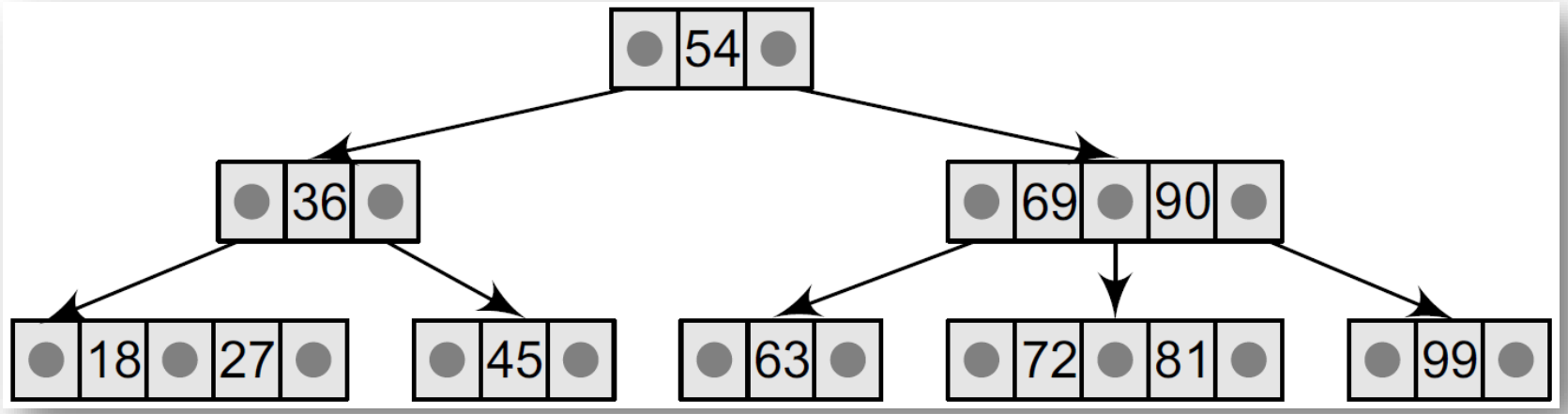
Split!



2-3 Trees – Deletion.

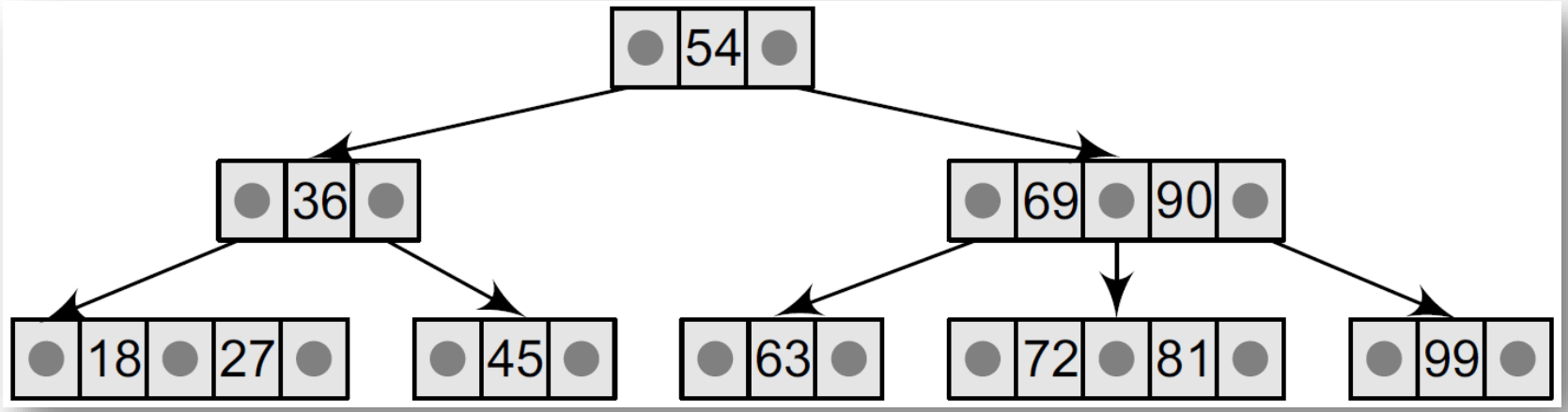
- To delete a value in internal node, it is replaced by its in-order successor and then removed
  - If a node becomes empty after deleting a value, it is then merged with another node to restore the property of the tree

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

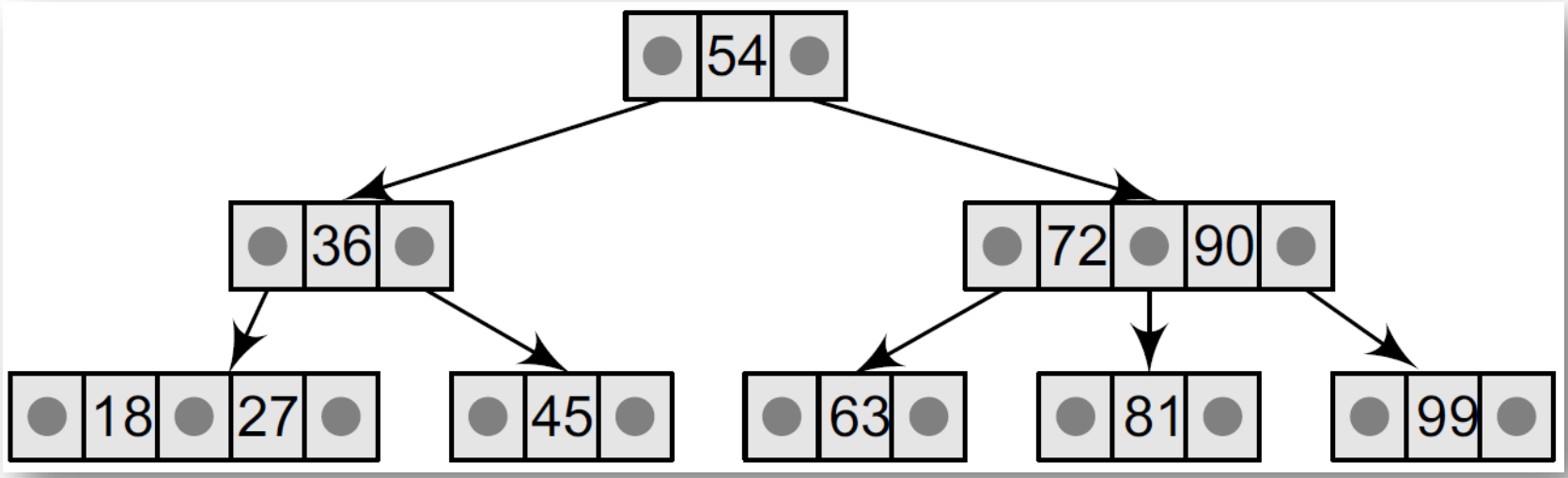


2-3 Trees – Deletion..

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

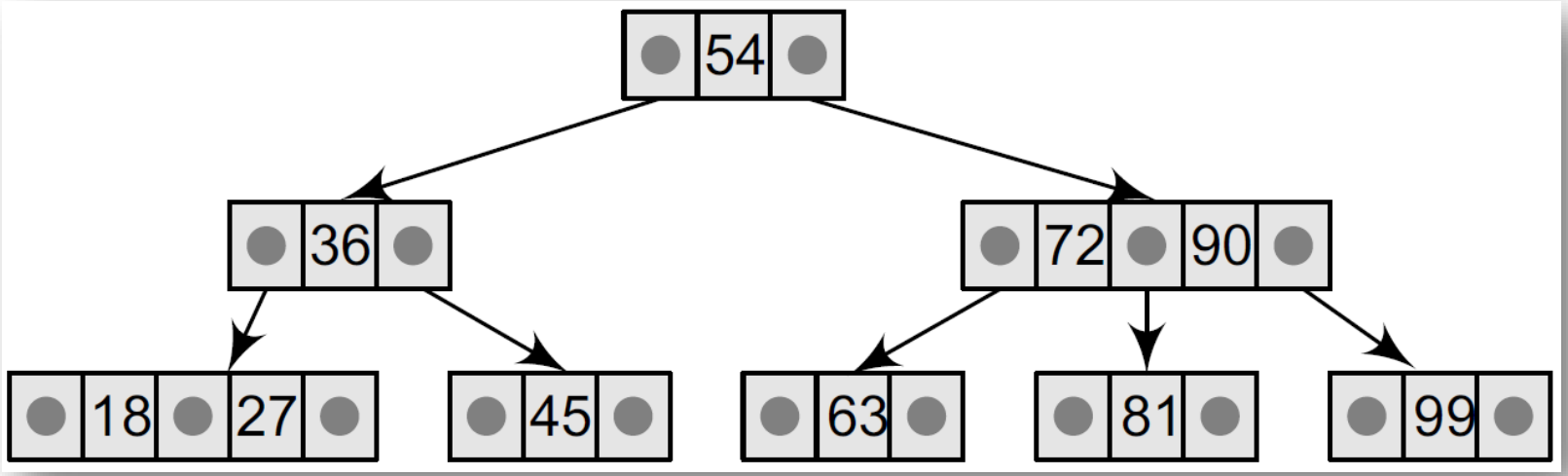


Delete 69

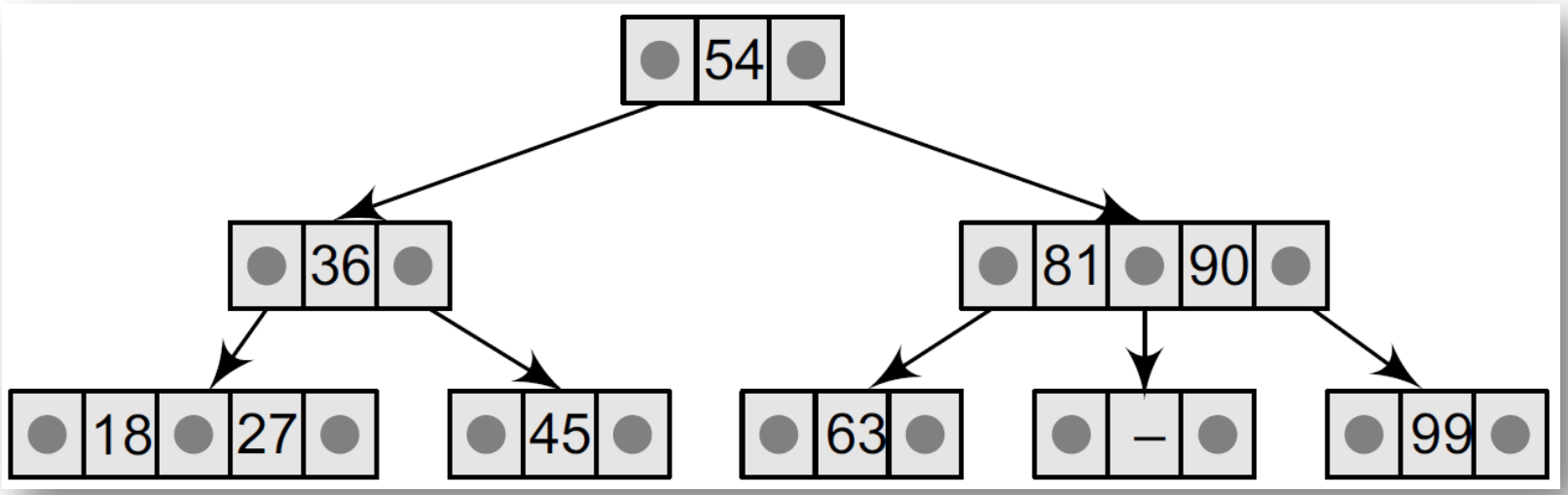


2-3 Trees – Deletion…

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

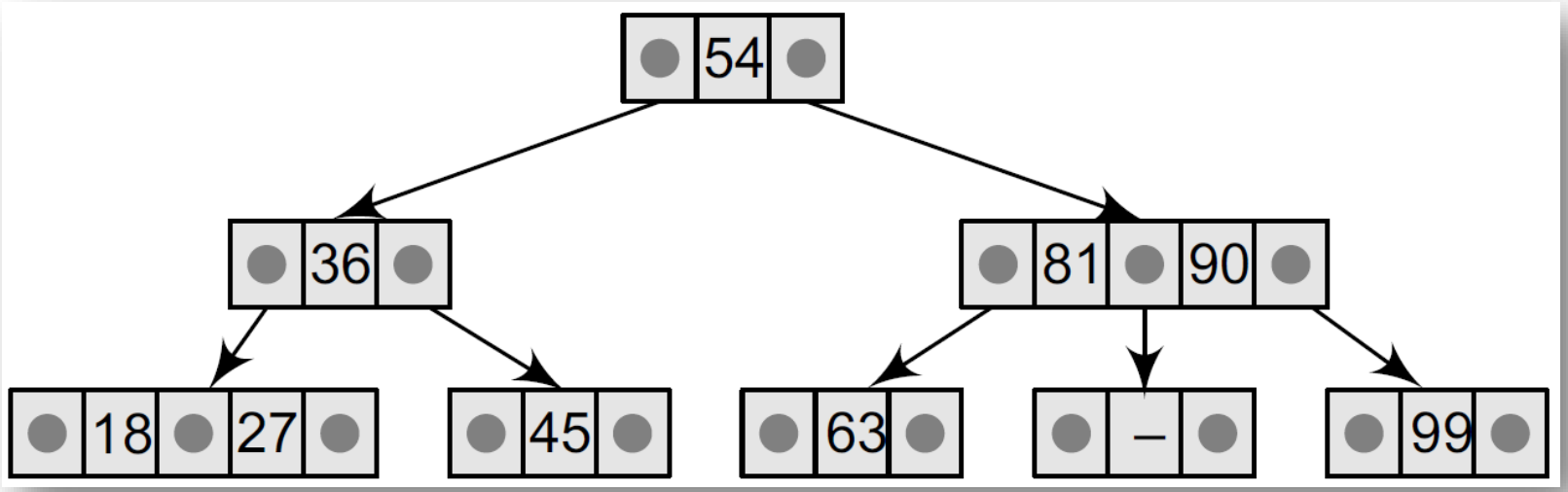


Delete 72

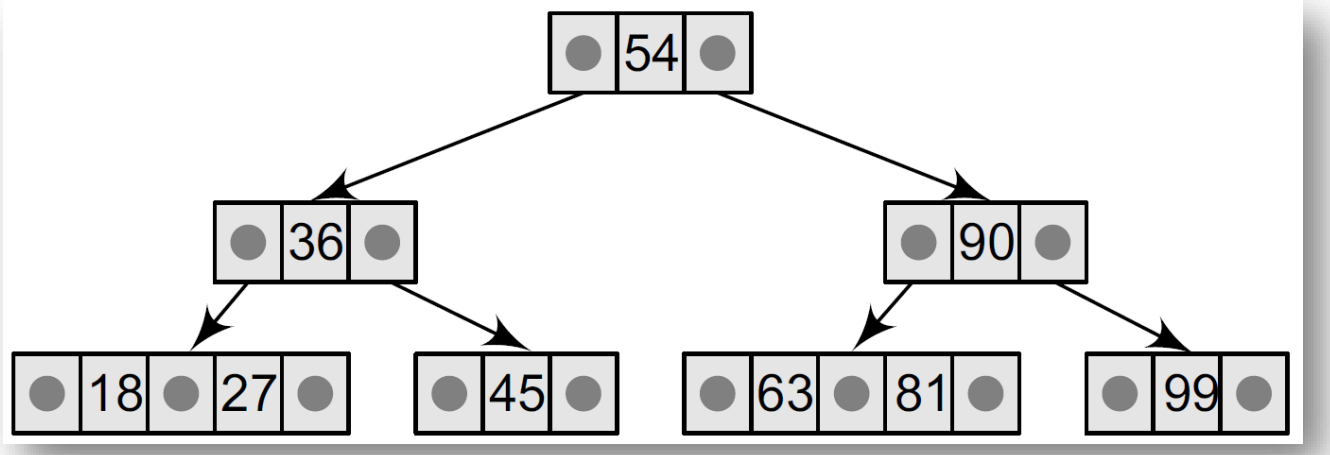


2-3 Trees – Deletion….

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

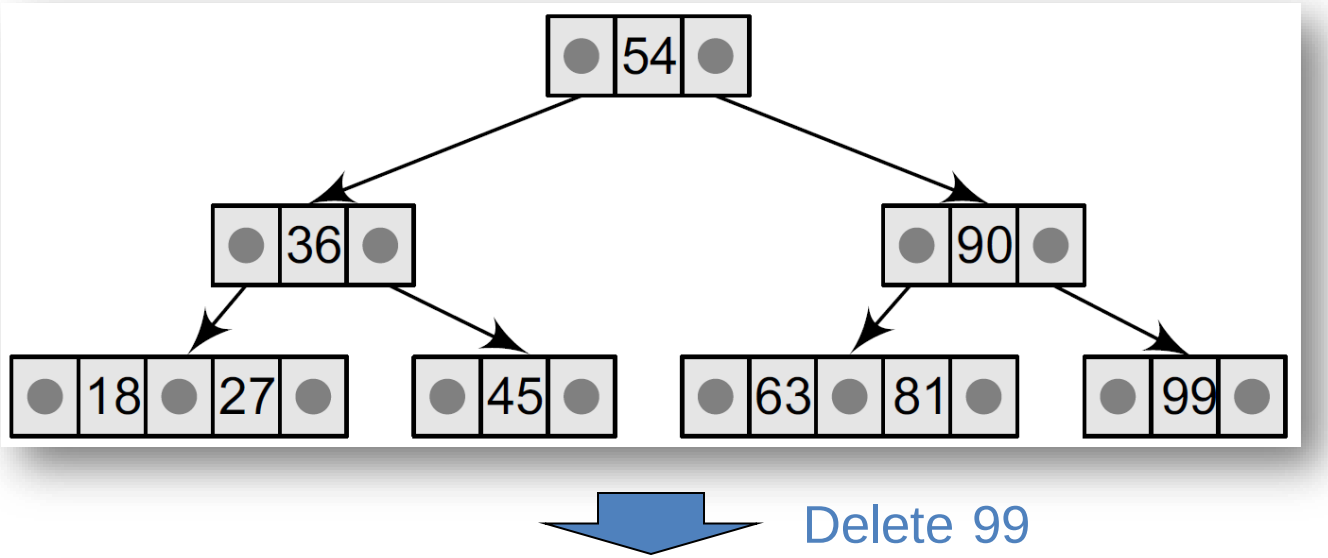


To merge the node, pull down the lowest data value in the parent's node and merge it with its left sibling



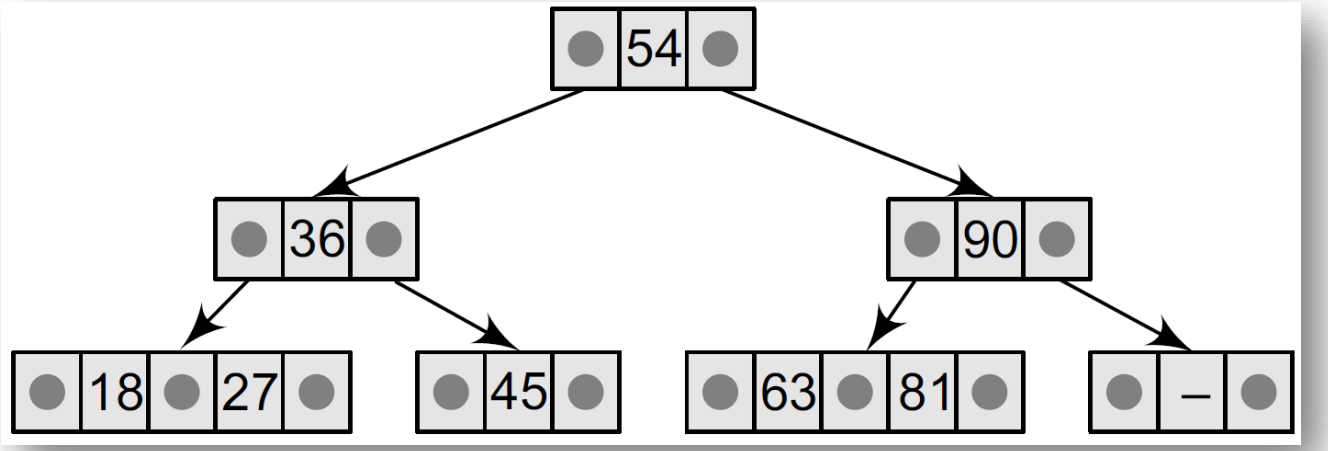
2-3 Trees – Deletion….

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

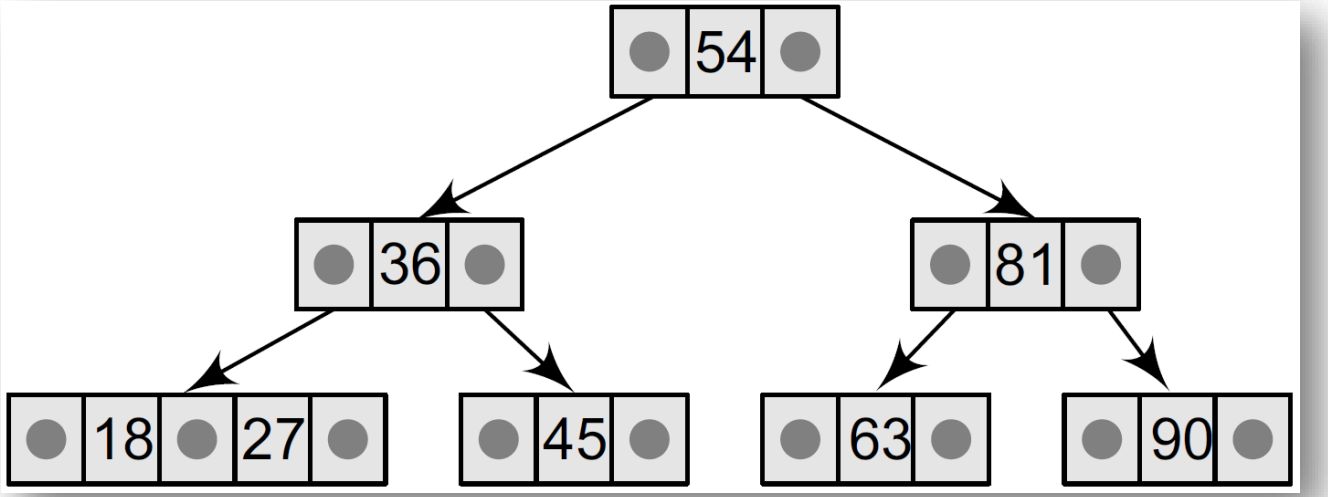


2-3 Trees – Deletion…..

- Given a 2-3 tree, please delete the values 69, 72, 99, 81

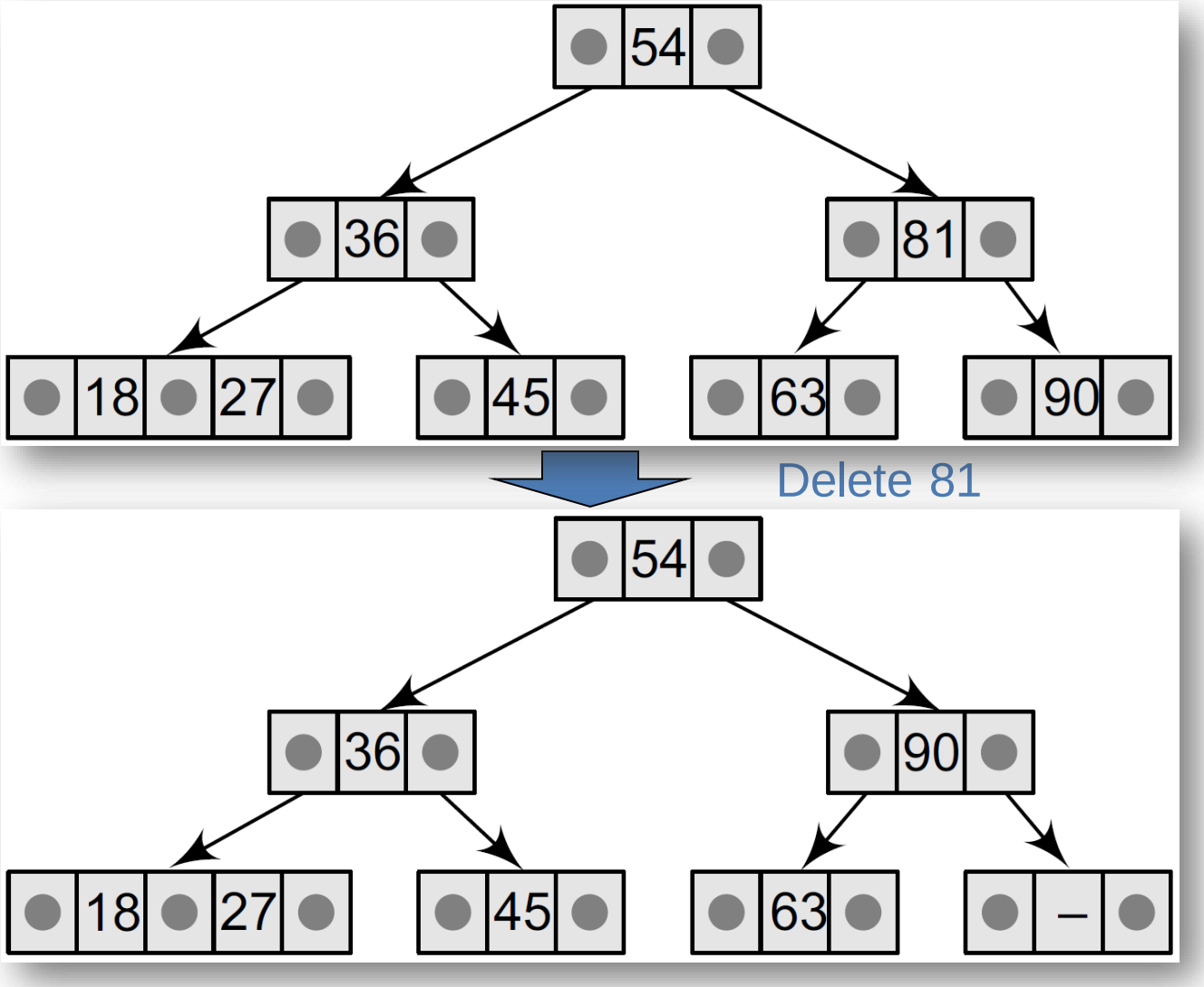


Merge & Split



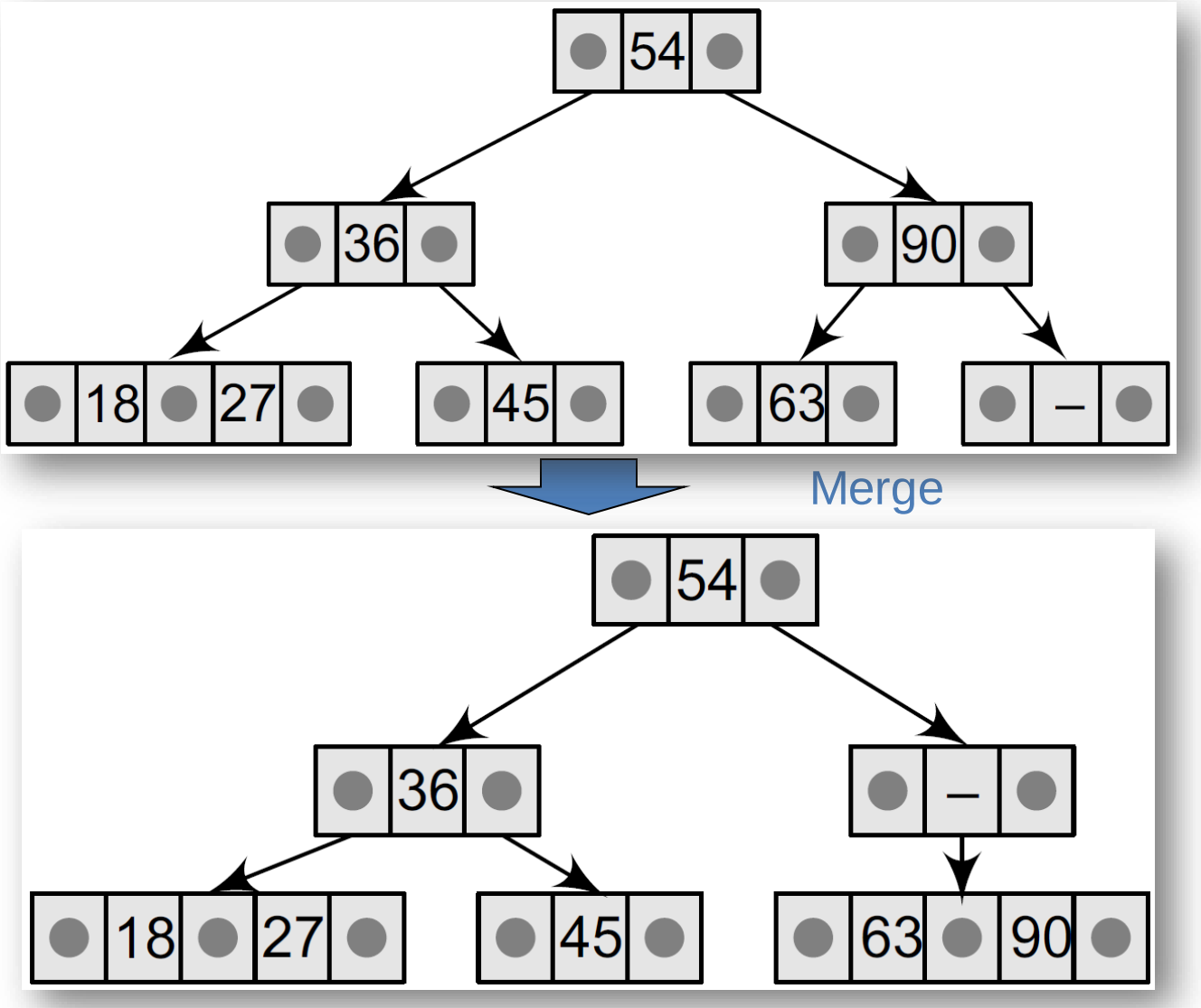
2-3 Trees – Deletion…..

- Given a 2-3 tree, please delete the values 69, 72, 99, 81



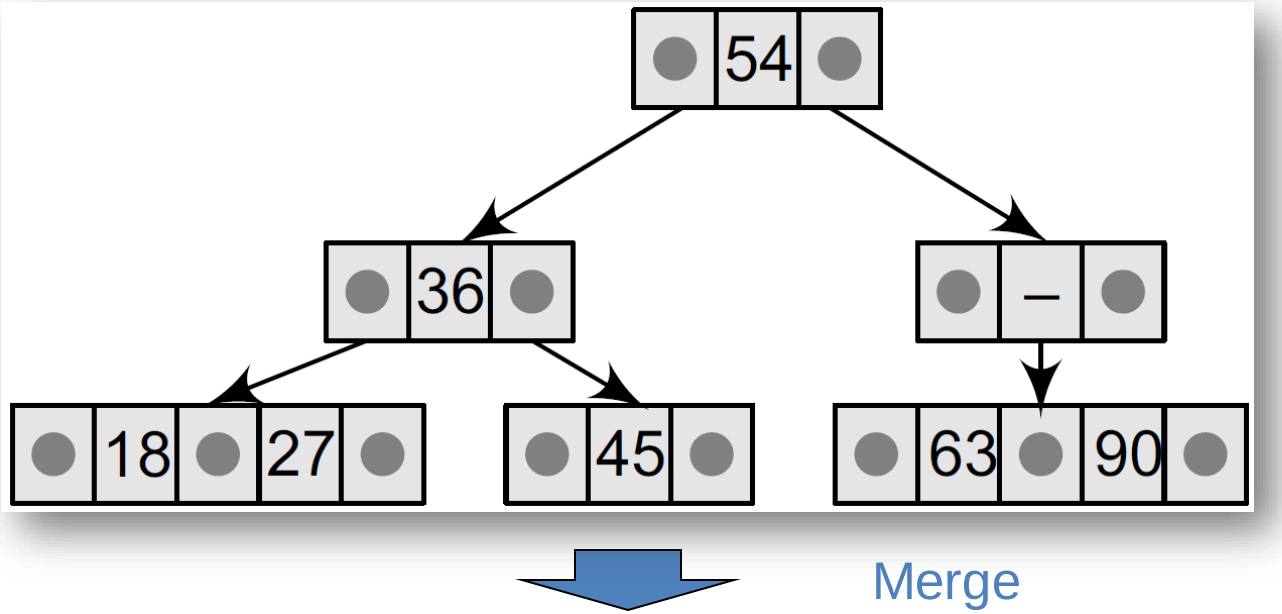
2-3 Trees – Deletion……

- Given a 2-3 tree, please delete the values 69, 72, 99, 81



2-3 Trees – Deletion……

- Given a 2-3 tree, please delete the values 69, 72, 99, 81



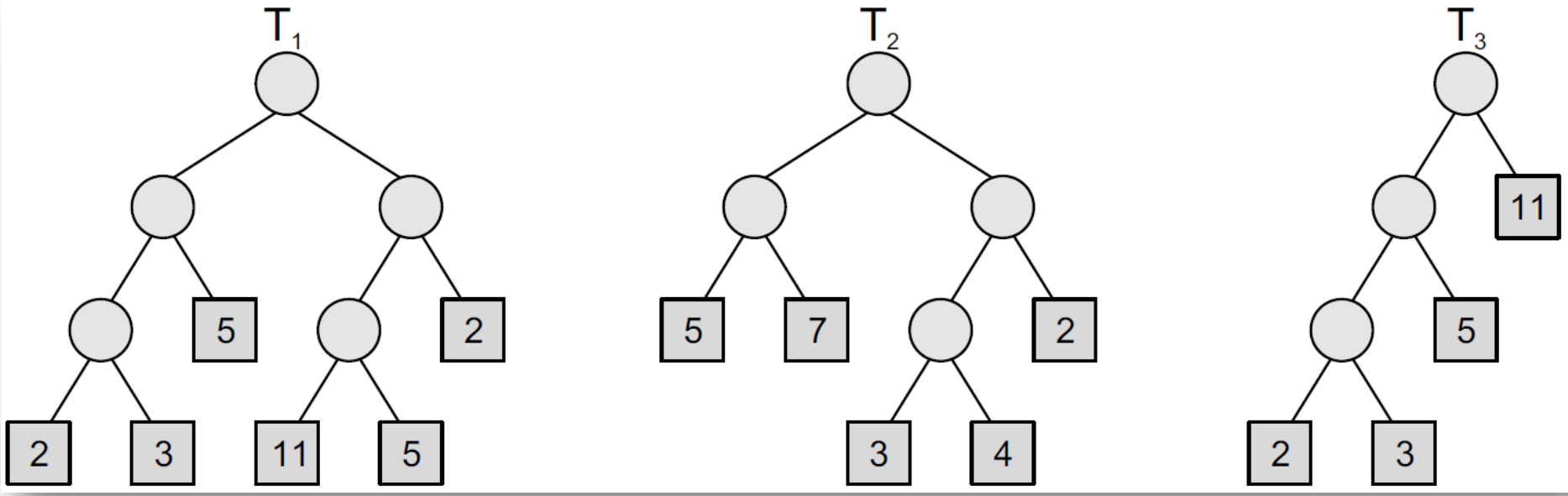
Huffman Trees

- Huffman coding is an entropy encoding algorithm developed by David A. Huffman in 1952 that is widely used as a lossless data compression technique

- The key idea behind Huffman algorithm is that it encodes the most common characters using shorter strings of bits than those used for less common source characters
  - The internal node is used to link to its child nodes
  - The external node contains the actual character and weight

Weighted External Path Length

- The weighted external path length
  - For T_1
    - $2 \times 3 + 3 \times 3 + 5 \times 2 + 11 \times 3 + 5 \times 3 + 2 \times 2 = 77$
  - For T_2
    - $5 \times 2 + 7 \times 2 + 3 \times 3 + 4 \times 3 + 2 \times 2 = 49$
  - For T_3
    - $2 \times 3 + 3 \times 3 + 5 \times 2 + 11 \times 1 = 36$

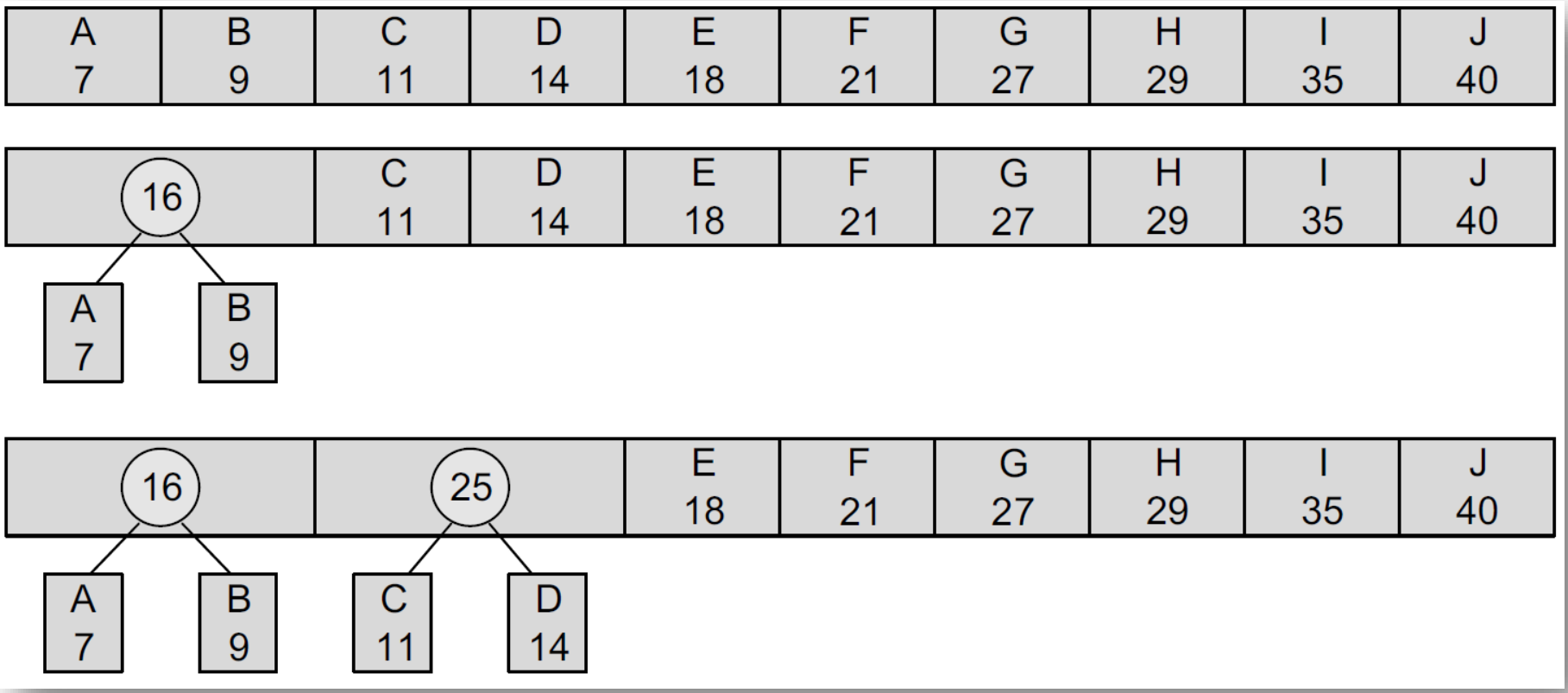


Creating a Huffman Tree

- Given n nodes and their weights, the Huffman algorithm is used to find a tree with a **minimum** weighted path length
  - Creating a new node whose children are the two nodes with the smallest weight
  - The weight of the new node is the sum of the two children
  - Repeat the process until the tree has only one node

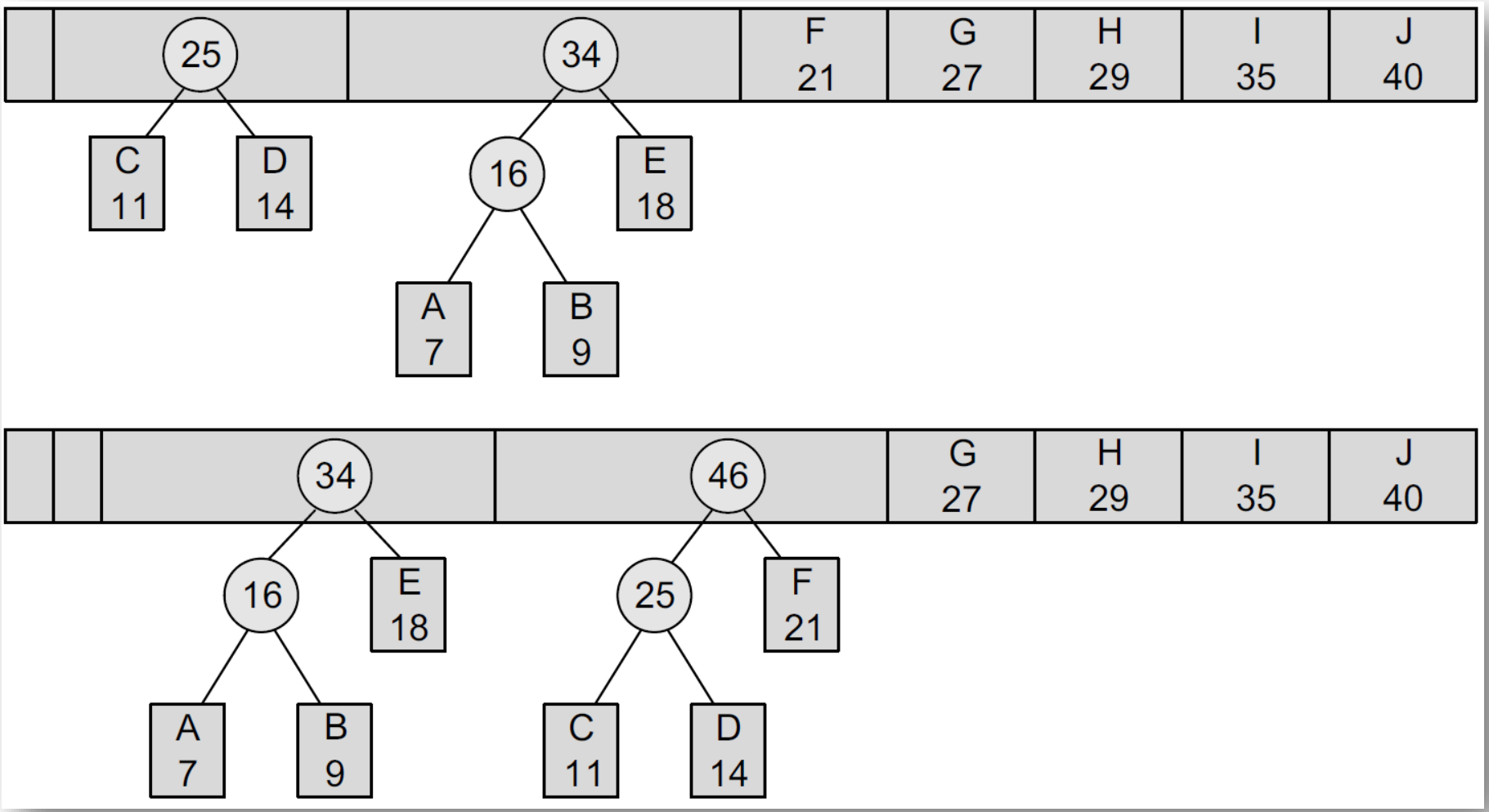
Example.

- Create a Huffman tree with the following sorted nodes



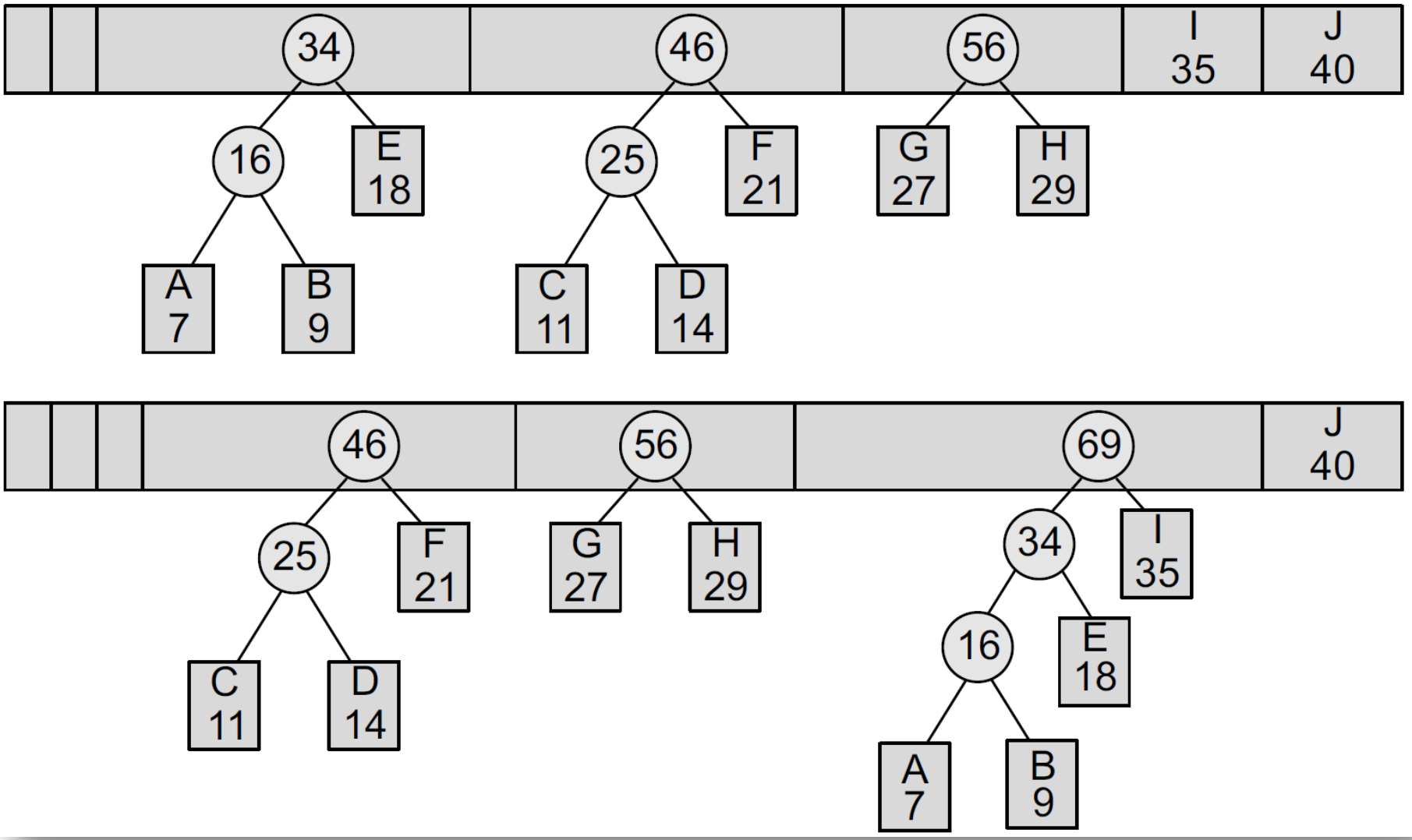
Example..

- Create a Huffman tree with the following sorted nodes



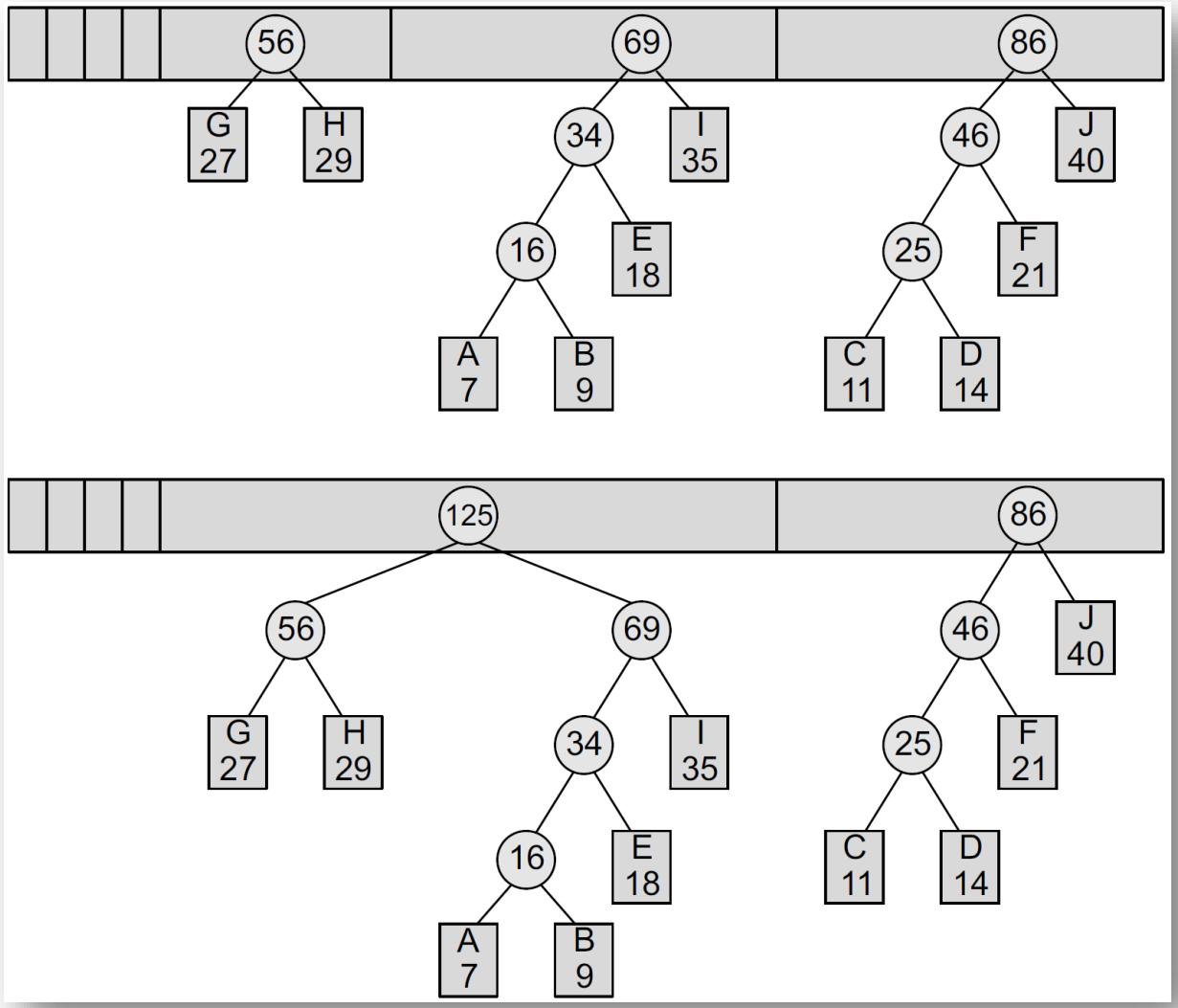
Example…

- Create a Huffman tree with the following sorted nodes



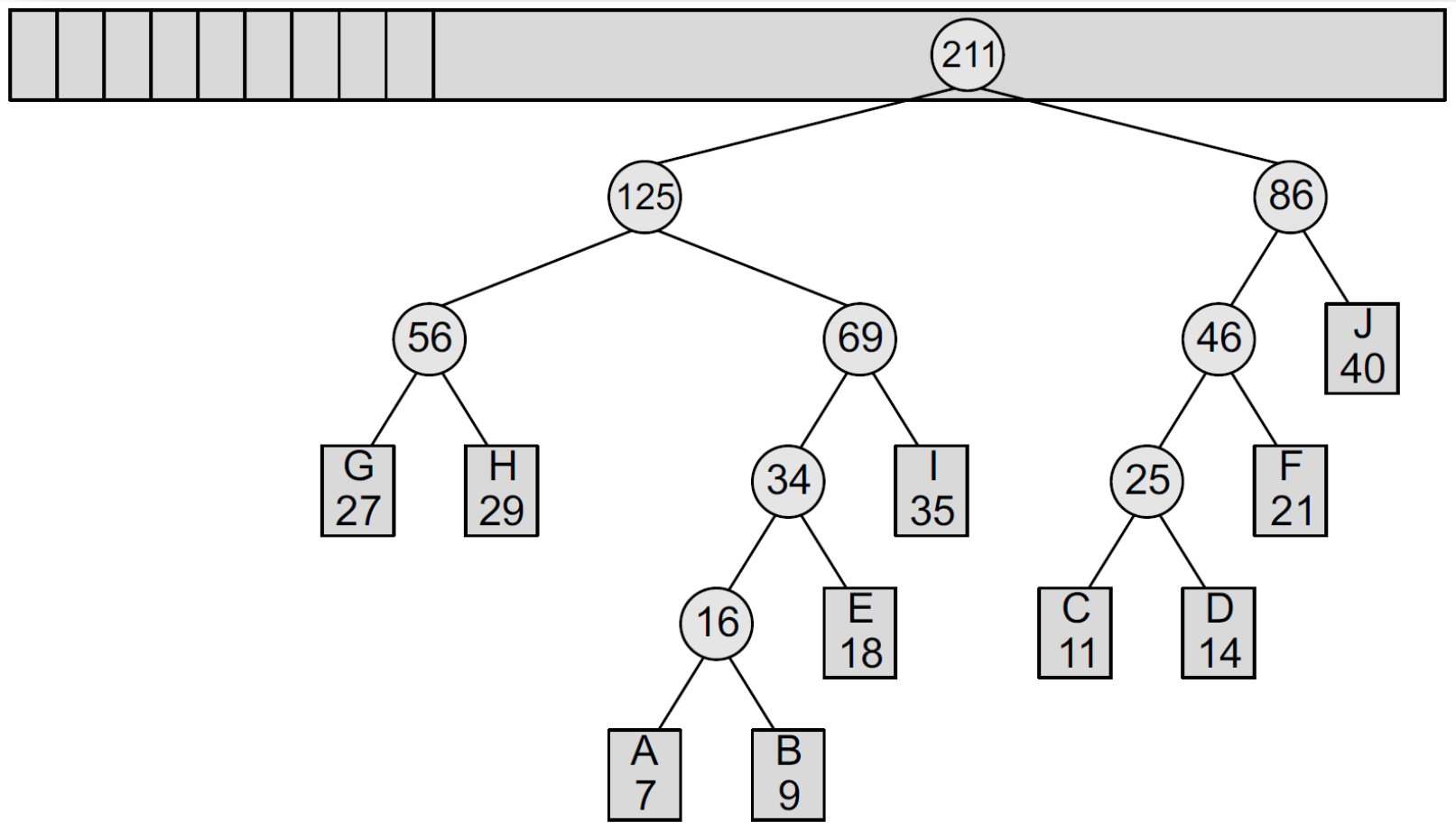
Example….

- Create a Huffman tree with the following sorted nodes



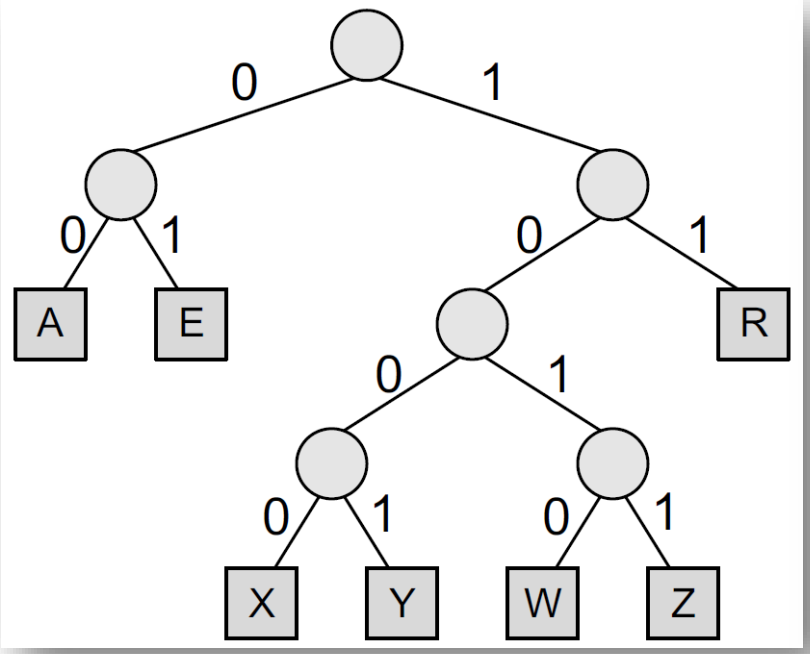
Example…..

- Create a Huffman tree with the following sorted nodes



Data Coding

- For a Huffman tree, every left branch is coded with 0 and every right branch is coded with 1
  - For a character sequence: AAERZ
    - By Huffman Coding Scheme: 000001111011
    - By Original Coding Scheme: 000000001010110



Character	Code	Original Coding
A	00	000
E	01	001
R	11	010
W	1010	011
X	1000	100
Y	1001	101
Z	1011	110

Summary

- We introduce a lots of variants of binary search tree, which is the fundamental
  - Huffman Tree is created in 1952
    - Data compression
  - Binary Search Tree, 1960
  - AVL Tree, 1962
    - Proposed by Georgy **A**delson-**V**elsky & Evgenii **L**andis
  - 2-3 Tree, 1970
  - Red-Black Tree, 1972
    - std::map in C++
      - ✓ The re-balance process is faster than AVL tree
  - B Tree, 1972
    - A B-tree of order 3 is a 2-3 tree
  - B+ Tree, 1973
    - NTFS uses B+ trees for directory and security-related metadata indexing
    - MySQL indexing
  - Splay Tree, 1985
    - For memory management algorithms

Questions?

kychen@mail.ntust.edu.tw