Unit 9 – B-Tree

CS 201 - Data Structures II

Spring 2023

Habib University

Syeda Saleha Raza

About Memory Management

- Memory Word (4, 8, 16 bytes)
- Memory addresses
- Blocks
- Fragmentation
- Garbage Collection

Memory Hierarchy

- Registers
- Cache
- Internal Memory (Main memory)
- External Memory (Disks, drives, DVD, Tape)

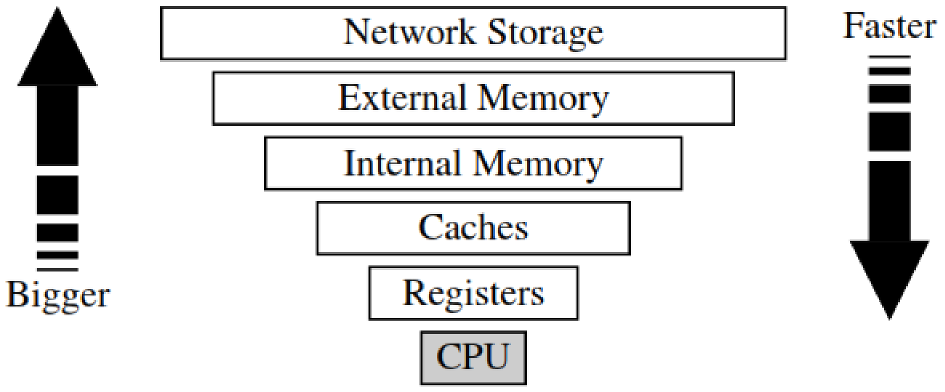


Figure 15.1: The memory hierarchy.

Locality of reference properties

- **Temporal locality:** If a program accesses a certain memory location, then there is increased likelihood that it accesses that same location again in the near future. For example, it is common to use the value of a counter variable in several different expressions, including one to increment the counter's value. In fact, a common adage among computer architects is that a program spends 90 percent of its time in 10 percent of its code.
- **Spatial locality:** If a program accesses a certain memory location, then there is increased likelihood that it soon accesses other locations that are near this one. For example, a program using an array may be likely to access the locations of this array in a sequential or near-sequential manner.

External Searching

- Dealing with large data files
- Disk Transfers
- I/O Complexity

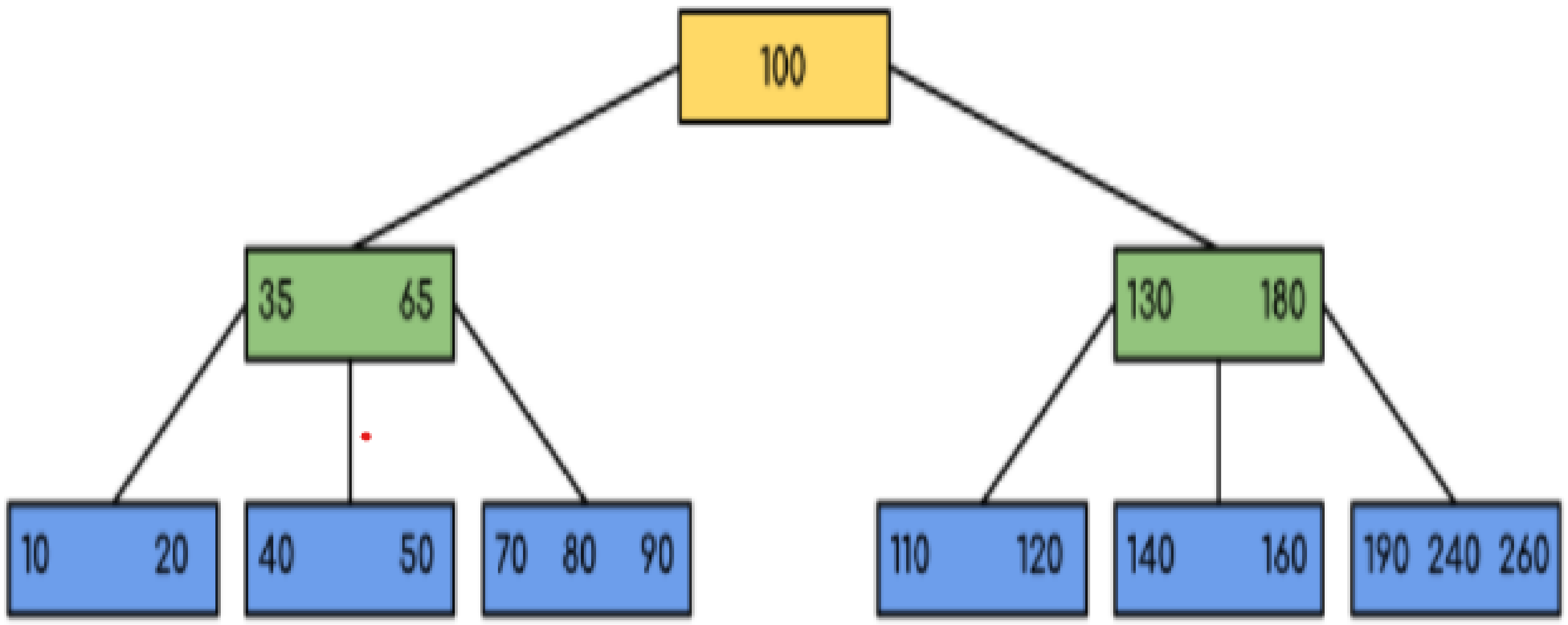
B-Tree

- A B-tree is the balanced Multiway tree and also known as the balanced sort tree. It is similar to binary search tree where the nodes are arranged on the basis of in-order traversal.

B-Tree

- There are certain conditions that must be true for a B-tree:
  - The height of the tree must remain as minimum as possible.
  - Above the leaves of the tree, there should not be any empty subtrees.
  - The leaves of the tree must occur at the same level.
  - All nodes must have some minimum number of children except leaf nodes.

Example



B-Tree

- All leaf nodes must be present at the same level.
- There can be at maximum n – 1 key for all nodes.
- There are at least two children for the root.
- n is the maximum number of children, and k is the number of keys that each node can contain.
- There are at least n/2 children and at most n non-empty children.
- The tree gets divided so that values in the left subtree shall be less than the value in the right subtree.

Why use B-Tree?

- Reduces the number of reads made on the disk
- B Trees can be easily optimized to adjust its size (that is the number of child nodes) according to the disk size
- It is a specially designed technique for handling a bulky amount of data.
- It is a useful algorithm for databases and file systems.
- A good choice to opt when it comes to reading and writing large blocks of data

2-3-4 OR 2-4 Tree

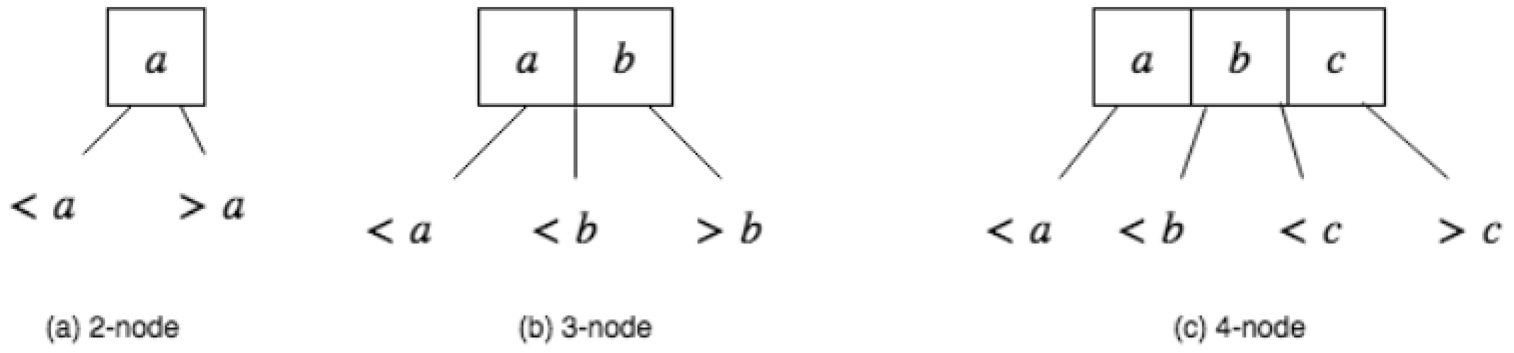


Figure 1: Illustrating node types

[2-3-4 Trees | Algorithm Tutor]

2-4 Tree

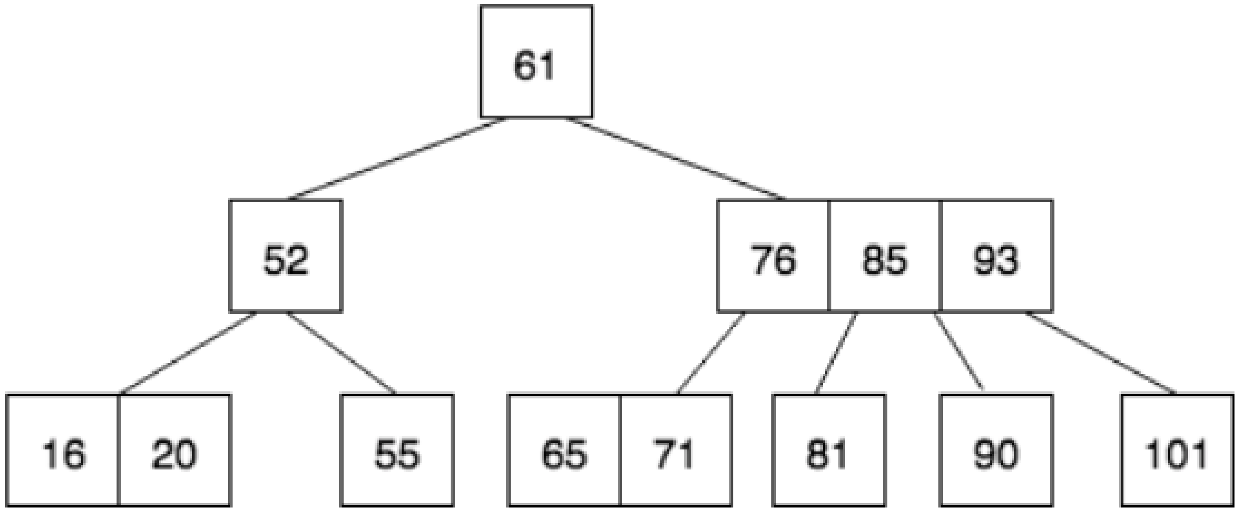


Figure 2: An example of a 2-3-4 tree

2-3-4 Trees | Algorithm Tutor

Splitting

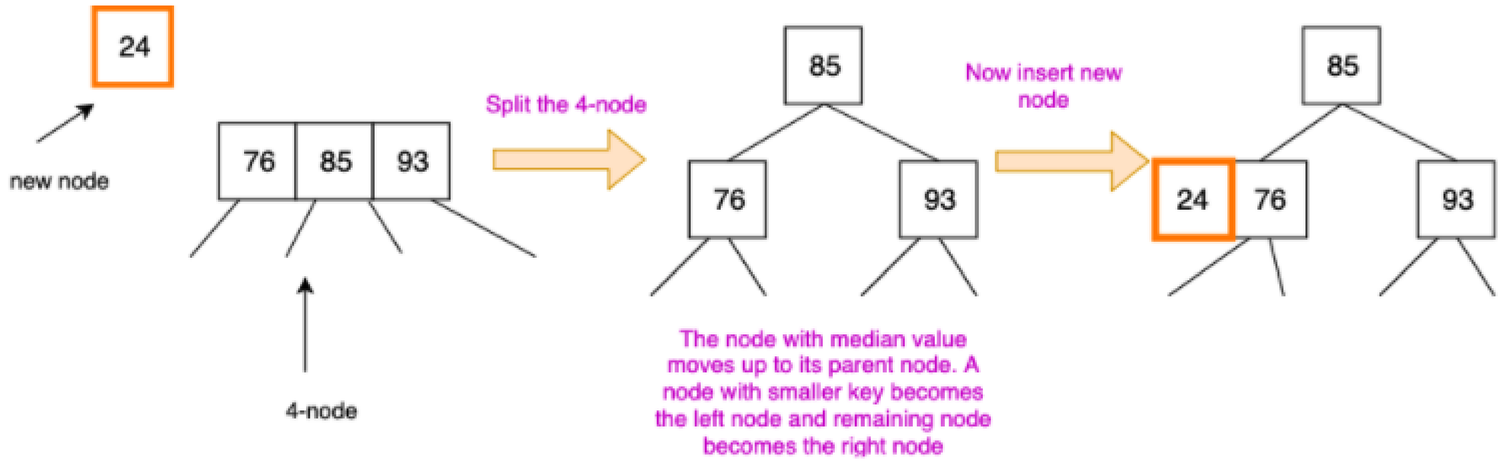
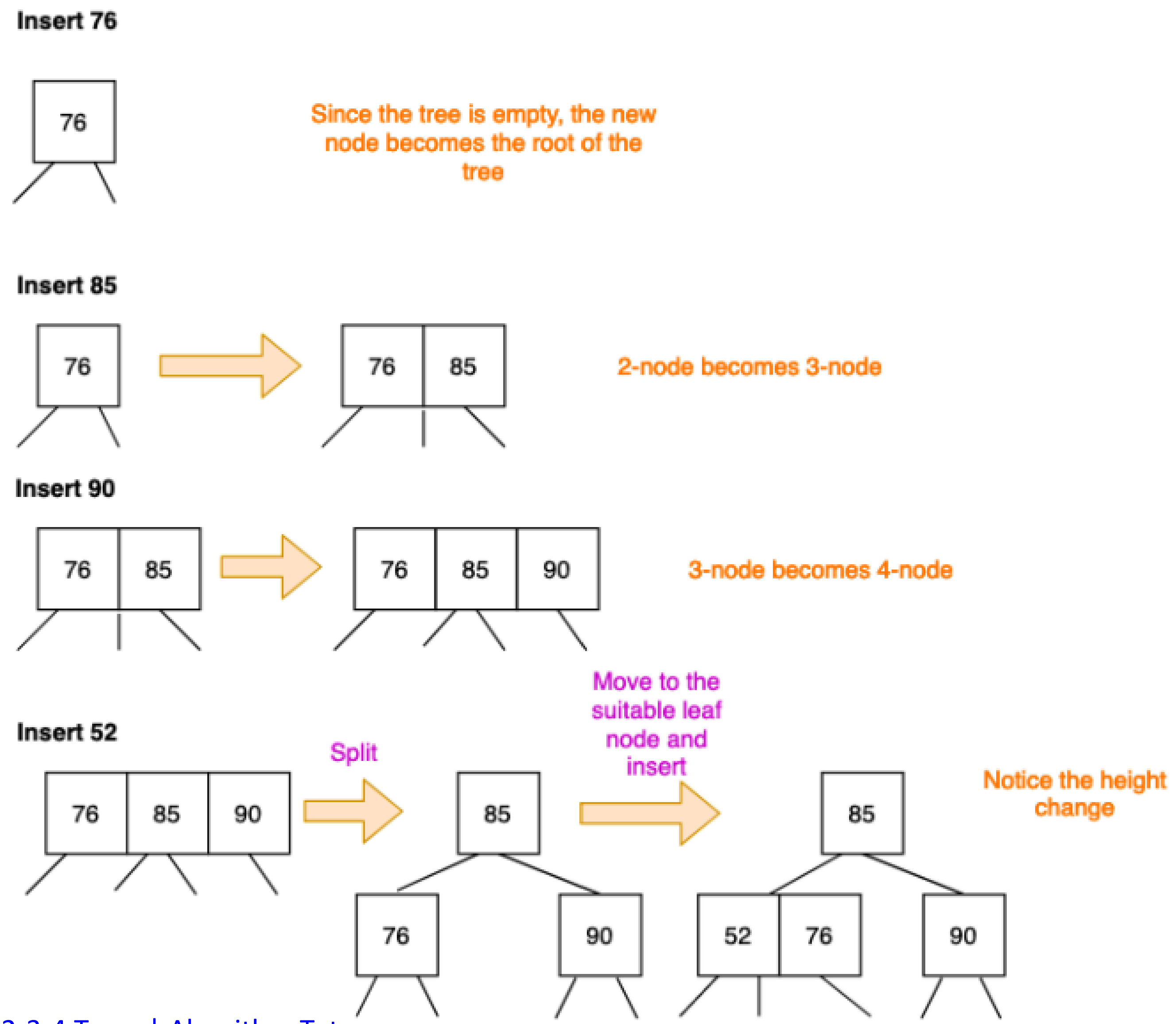

24
new node
76
85
93
4-node
Split the 4-node
85
76
93
The node with median value moves up to its parent node. A node with smaller key becomes the left node and remaining node becomes the right node
Now insert new node
85
24
76
93

Figure 3: Splitting of a 4-node before inserting a new node.

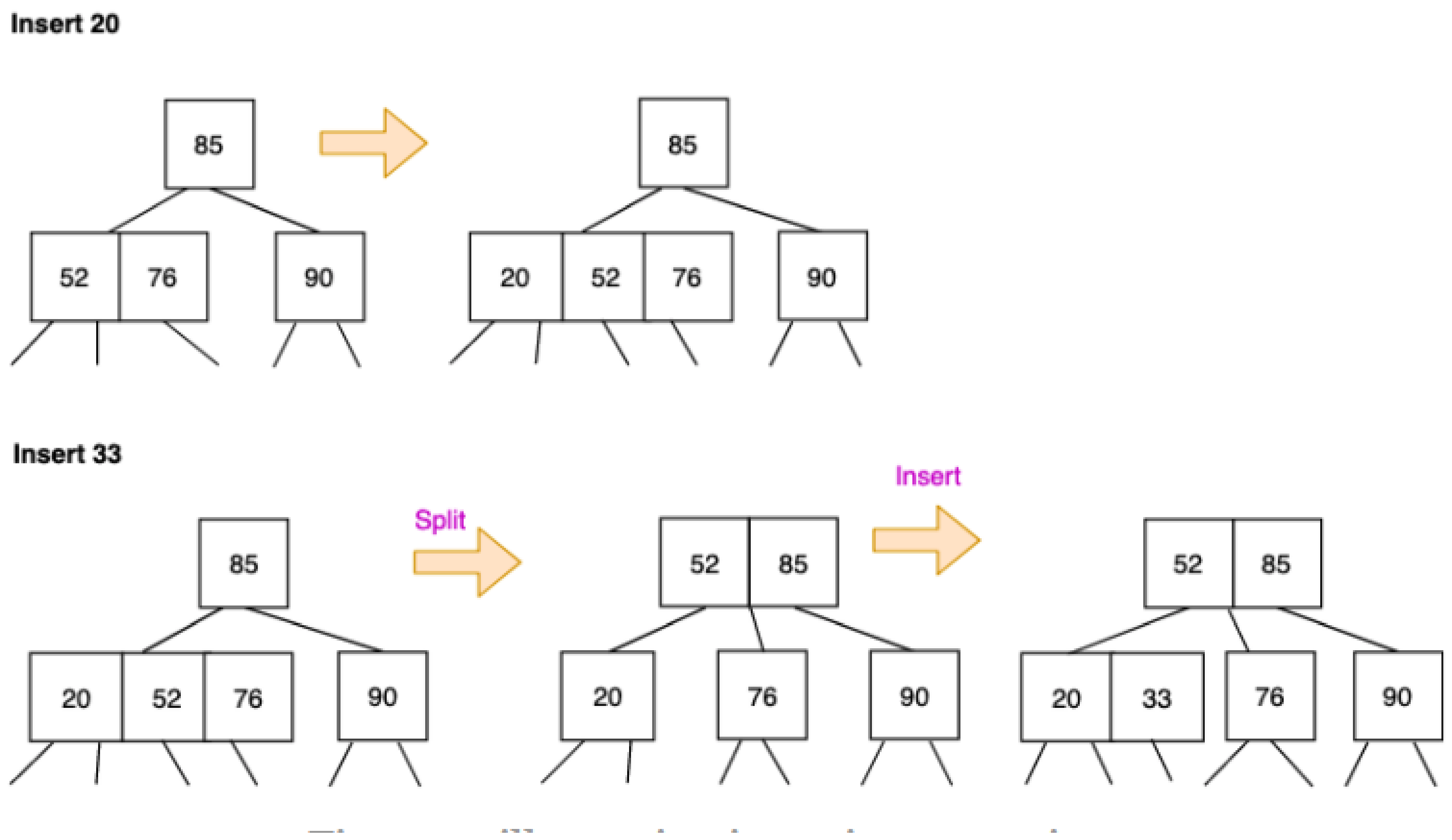
2-3-4 Trees | Algorithm Tutor

Sequence of Inserts



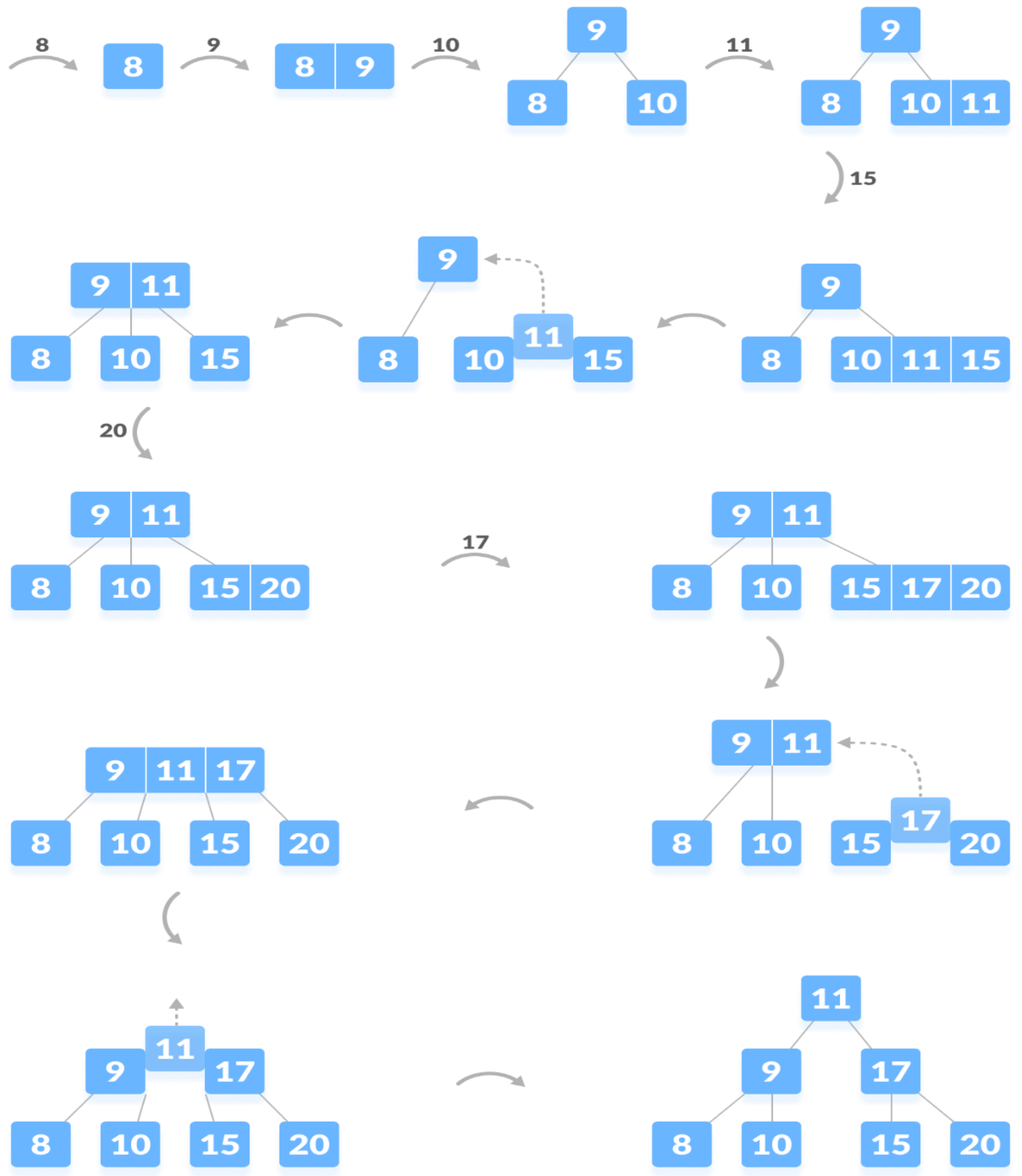
[2-3-4 Trees | Algorithm Tutor](https://algorithmtutor.com)

Sequence of Inserts



[2-3-4 Trees | Algorithm Tutor]

Insertion – Another Example



https://www.programiz.com/dsa/insertion-into-a-b-tree

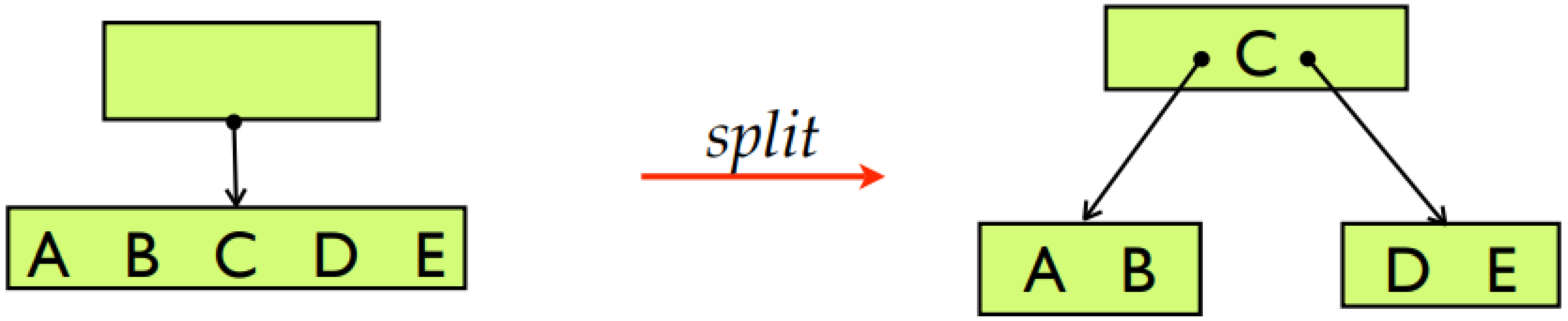
Insertion - Exercise

- Construct a 2-4 tree with the following values:
  - – 5, 3, 21,9,1,13,2,7,10,12,4,8

Insertion - Exercise

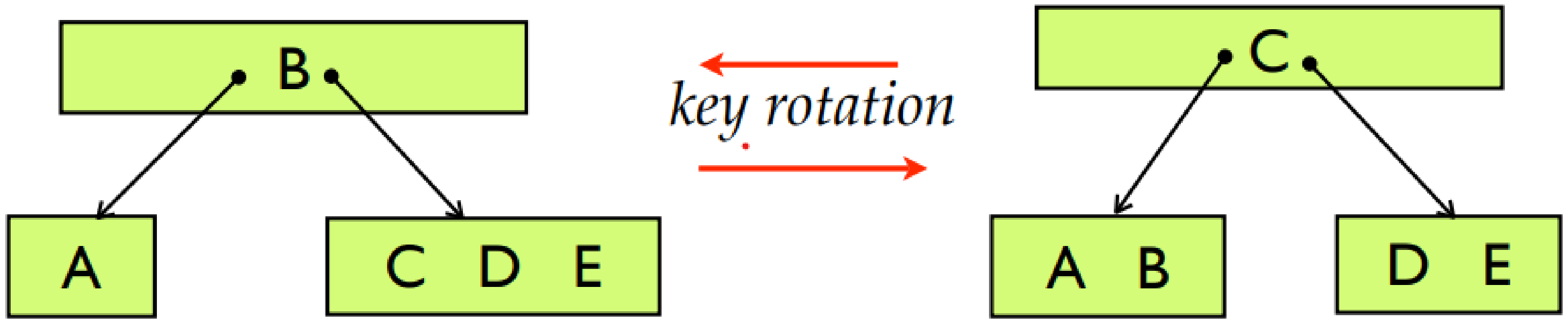
- Insert the following values in a 2-4 tree.
  - 21,14,39,64,11,78,51,37,20,7,19

B-Tree – Insert/Delete Operations



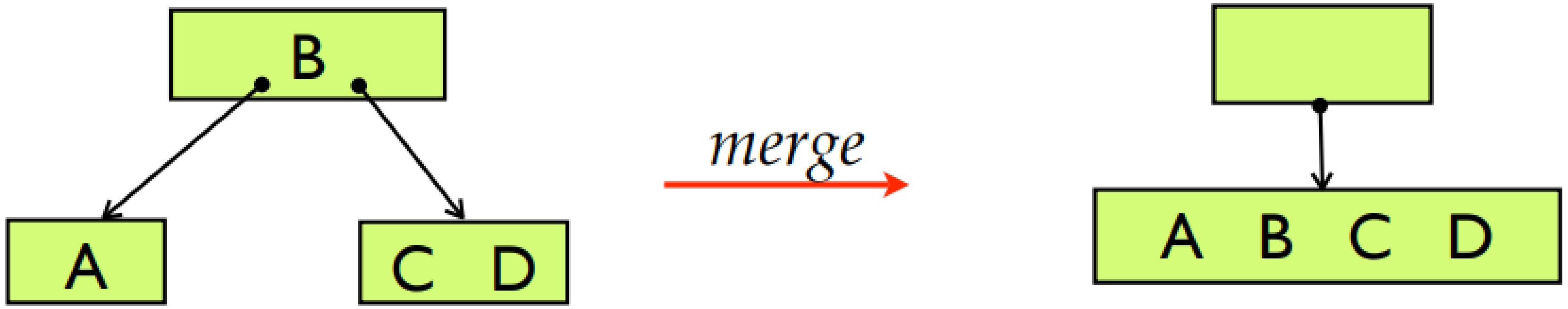
https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/btrees.pdf

B-Tree – Insert/Delete Operations



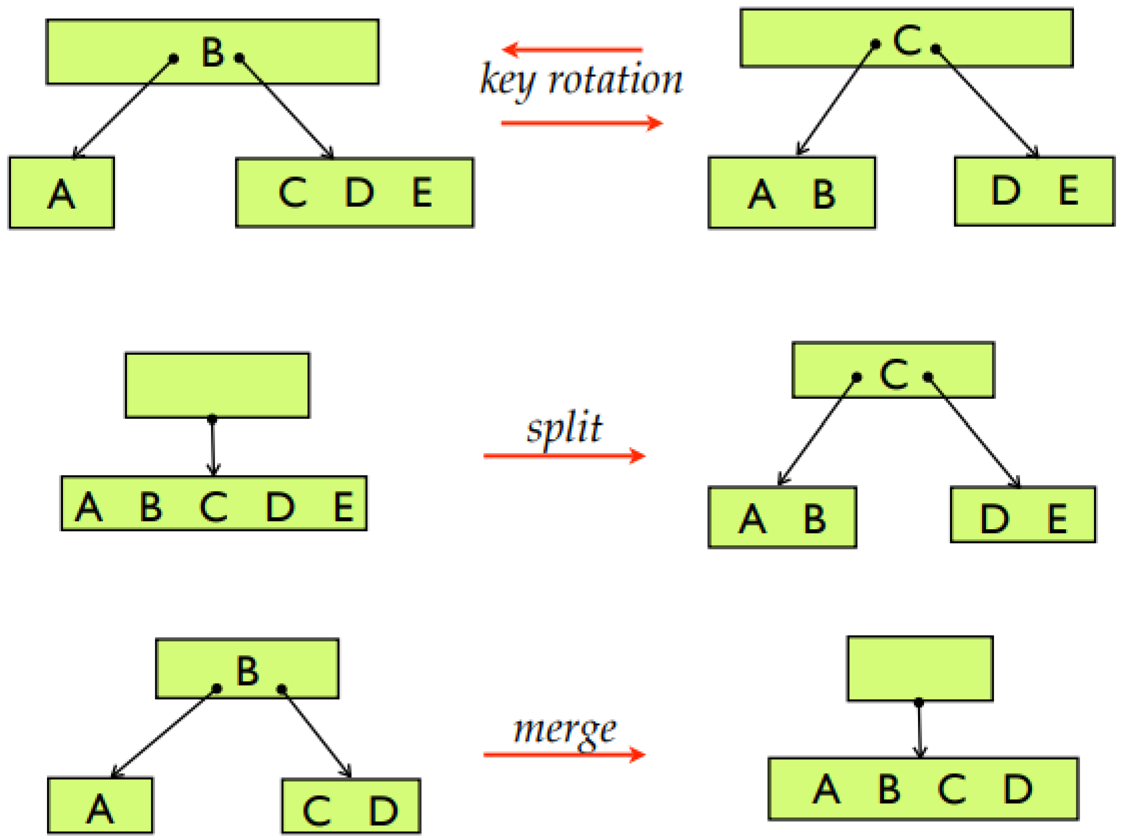
https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/btrees.pdf

B-Tree – Insert/Delete Operations



https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/btrees.pdf

Operators



https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/btrees.pdf

Deletion

[2-3-4 Trees | Algorithm Tutor]

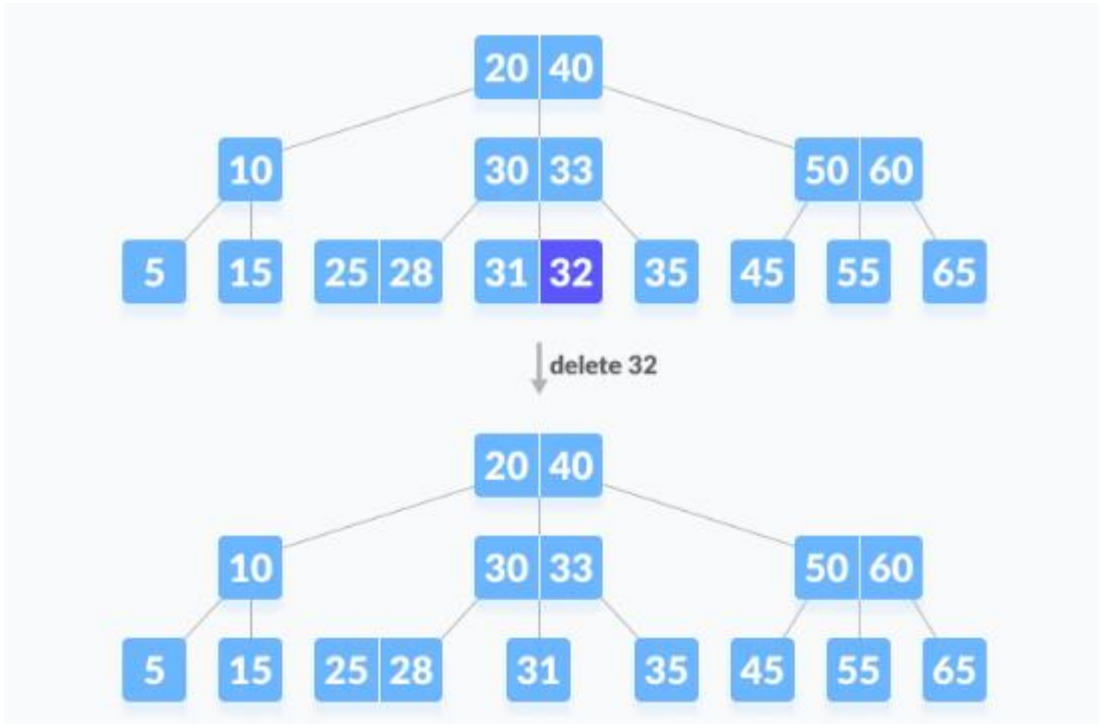
DSAP Sec. 15.3.2, 11.5

Deleting a node from B-Tree

- leaf node
  - with more than min keys:
    - Delete the key, rest is good.
  - with min key:
    - Try to borrow a key from left or right sibling if they have more than mid child. This would happen via rotation.
  - Else
    - Merge with sibling using parent
    - If parent is also min code, apply merging at parent's level too.
- Internal node
  - if the node has more than min child
    - Delete the key, rest if good.
  - Replace by inorder predecessor/inorder successor if left/right child has more than min child
  - Else merge left and right child
  - If the node to be deleted has less then min nodes (after deletion), then merge this node with its siblings.

Deleting a leaf node – Case I

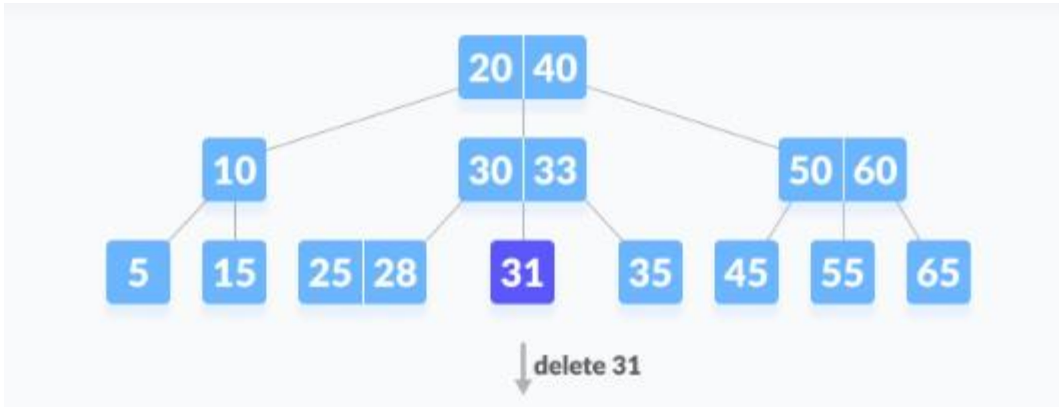
- The deletion of the key does not violate the property of the minimum number of keys a node should hold.

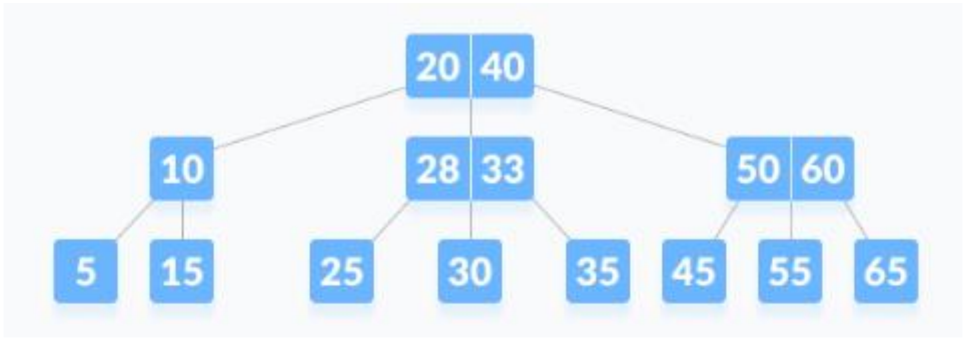


https://www.programiz.com/dsa/deletion-from-a-b-tree

Deleting a leaf node – Case II

- The deletion of the key violates the property of the minimum number of keys a node should hold. In this case, we borrow a key from its immediate neighboring sibling node in the order of left to right.

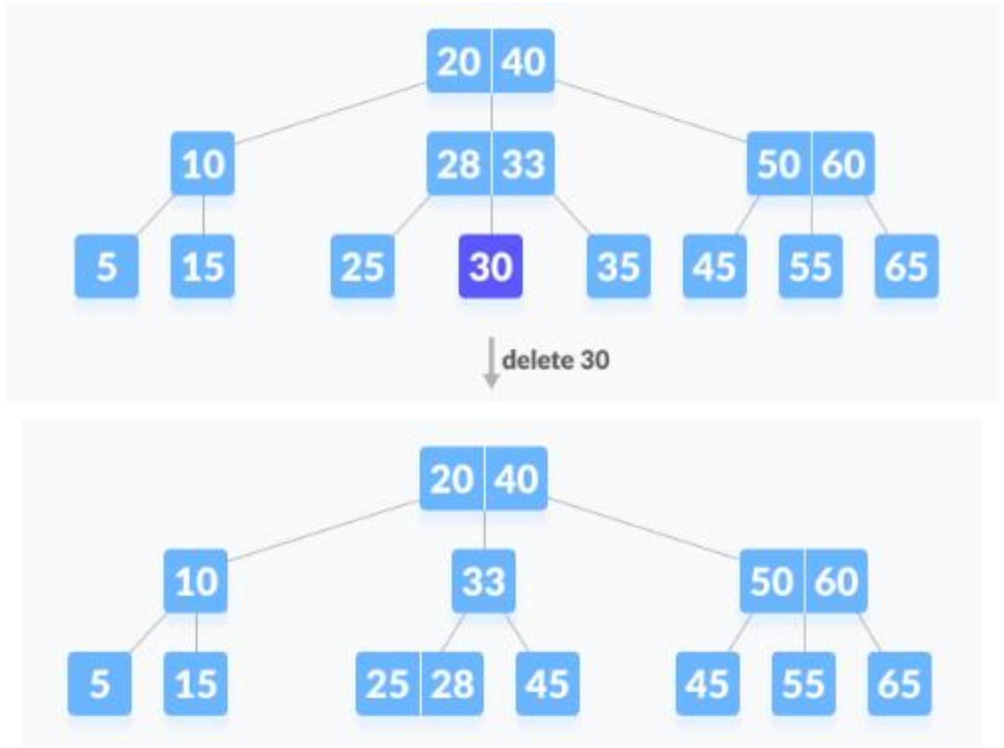




https://www.programiz.com/dsa/deletion-from-a-b-tree

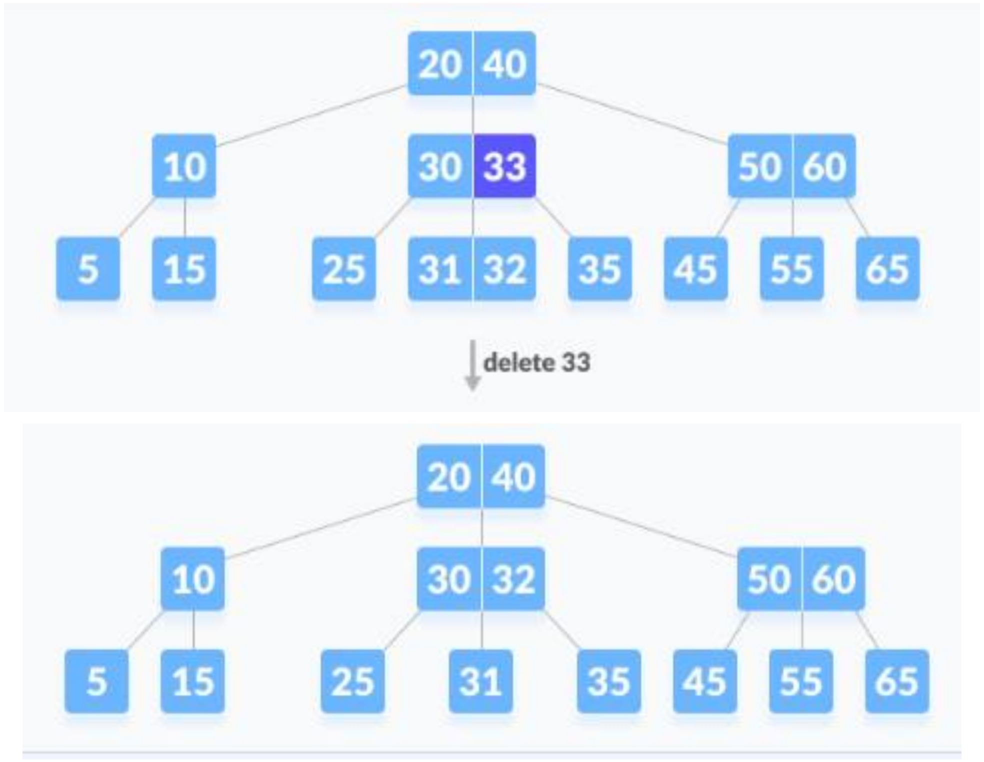
Deleting a leaf node – Case III

- If both the immediate sibling nodes already have a minimum number of keys, then merge the node with either the left sibling node or the right sibling node. **This merging is done through the parent node.**



Deleting an internal node – Case I

- The internal node, which is deleted, is replaced by an inorder predecessor if the left child has more than the minimum number of keys.



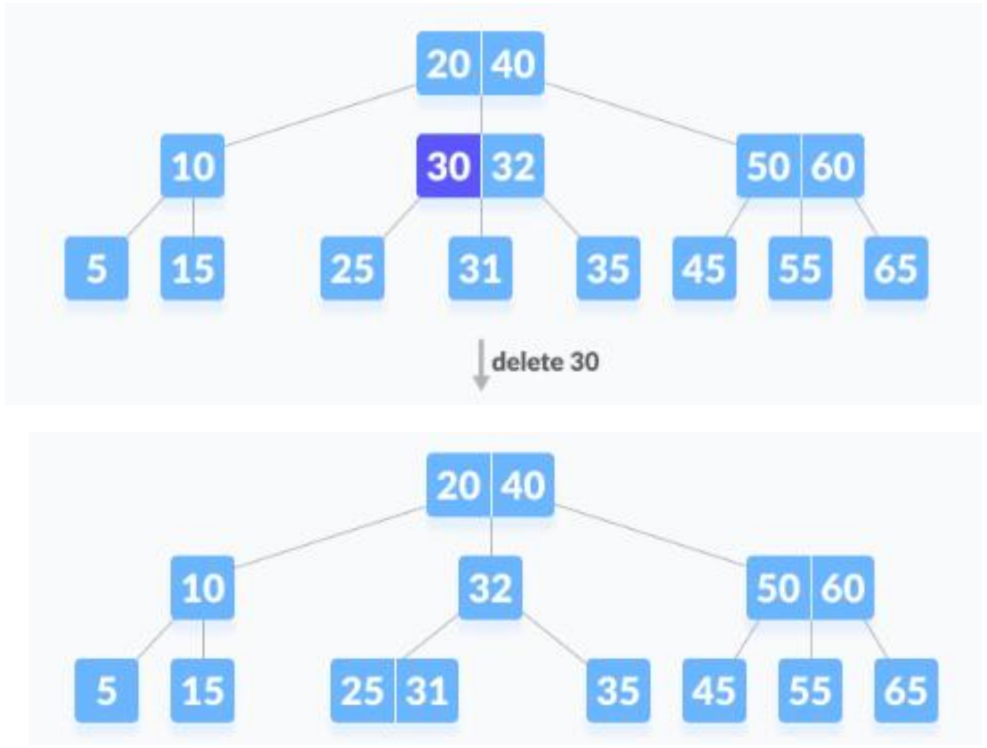
https://www.programiz.com/dsa/deletion-from-a-b-tree

Deleting an internal node – Case II

- The internal node, which is deleted, is replaced by an inorder successor if the right child has more than the minimum number of keys.

Deleting an internal code – Case III

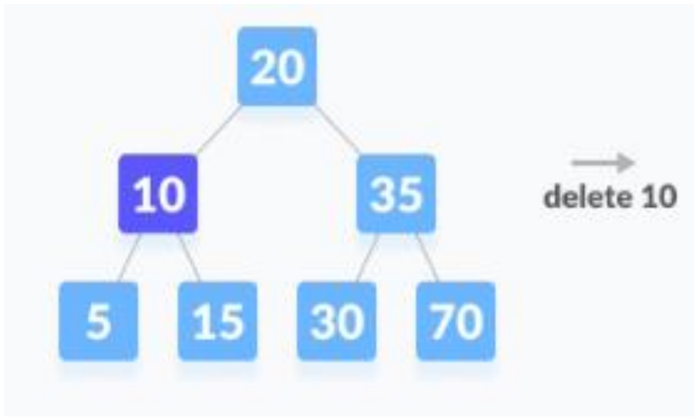
- If either child has exactly a minimum number of keys then, merge the left and the right children.

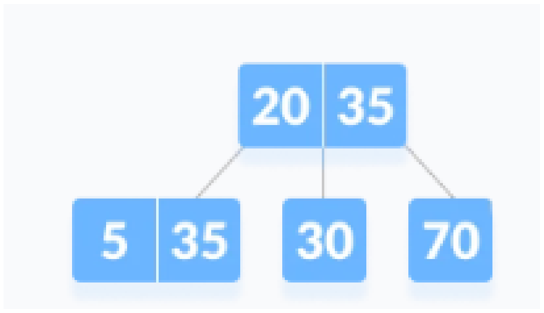


https://www.programiz.com/dsa/deletion-from-a-b-tree

Deleting an internal node – Case IV

- In this case, the height of the tree shrinks. If the sibling also has only a minimum number of keys then, merge the node with the sibling along with the parent. Arrange the children accordingly (increasing order).

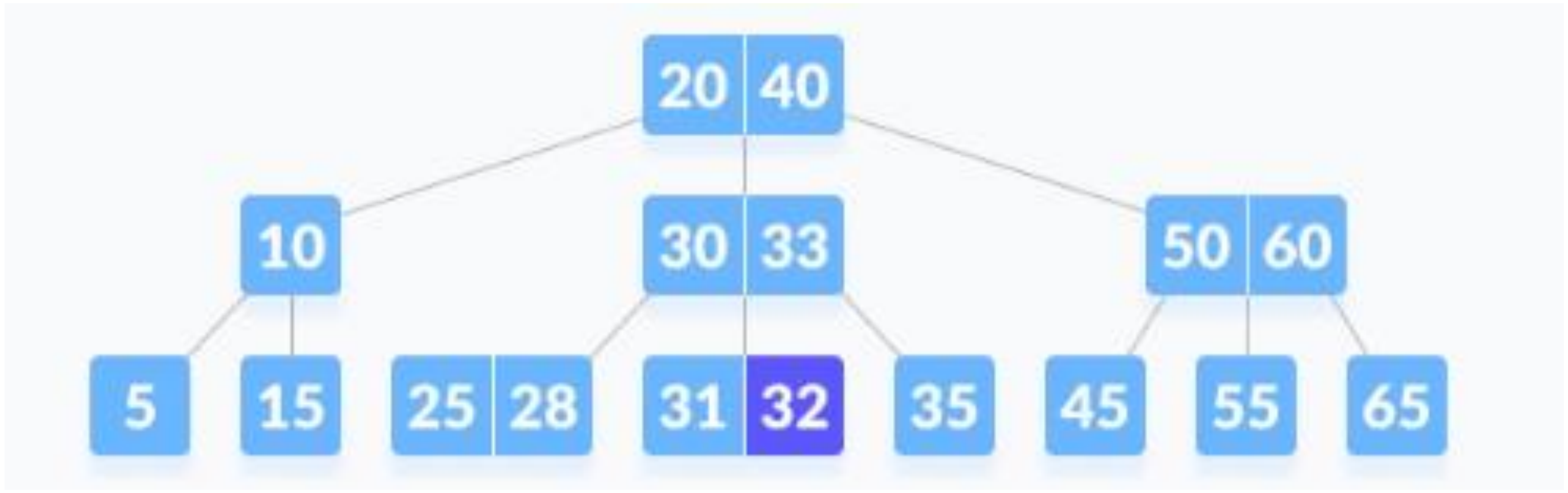




https://www.programiz.com/dsa/deletion-from-a-b-tree

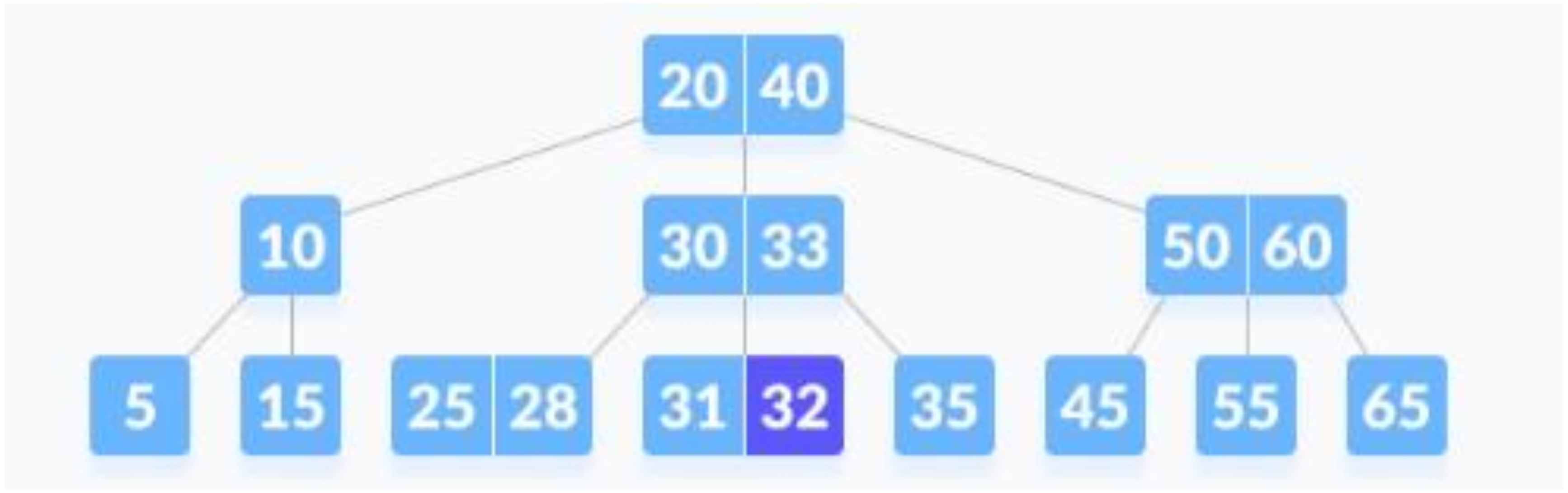
Exercise

- Delete 35.



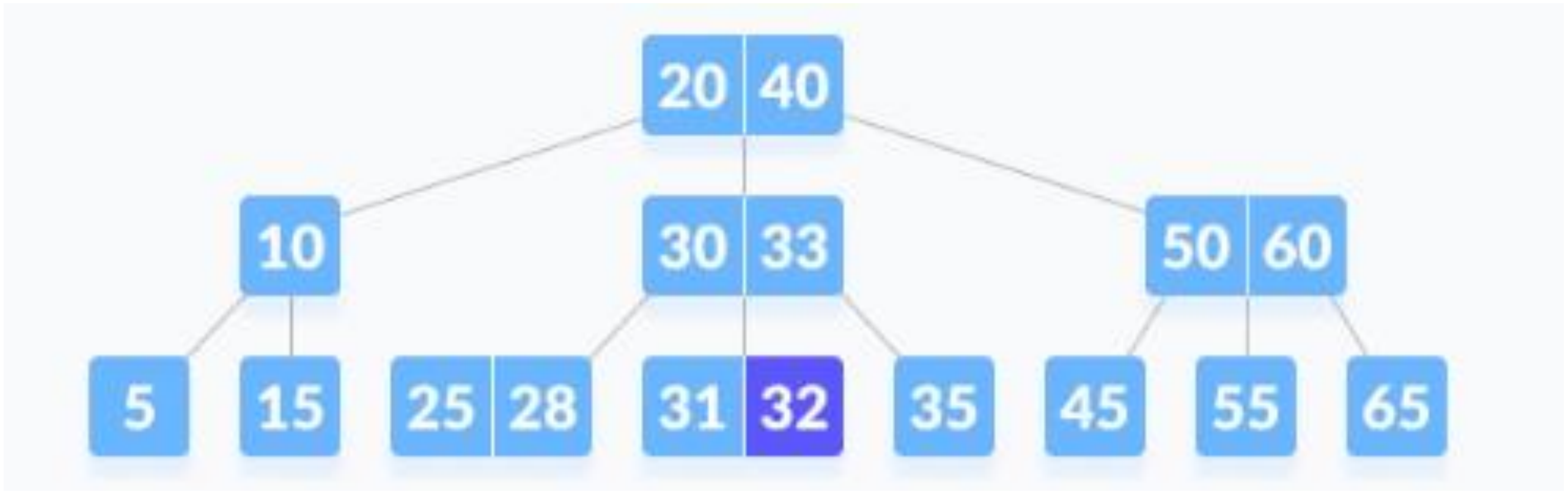
Exercise

- Delete 45.



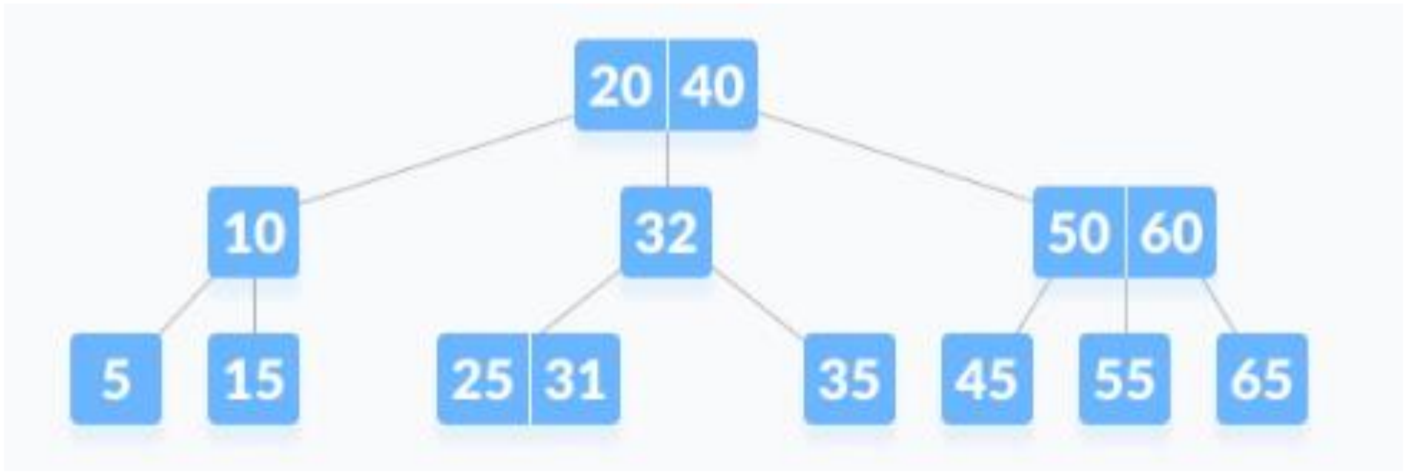
Exercise

- Delete 30.



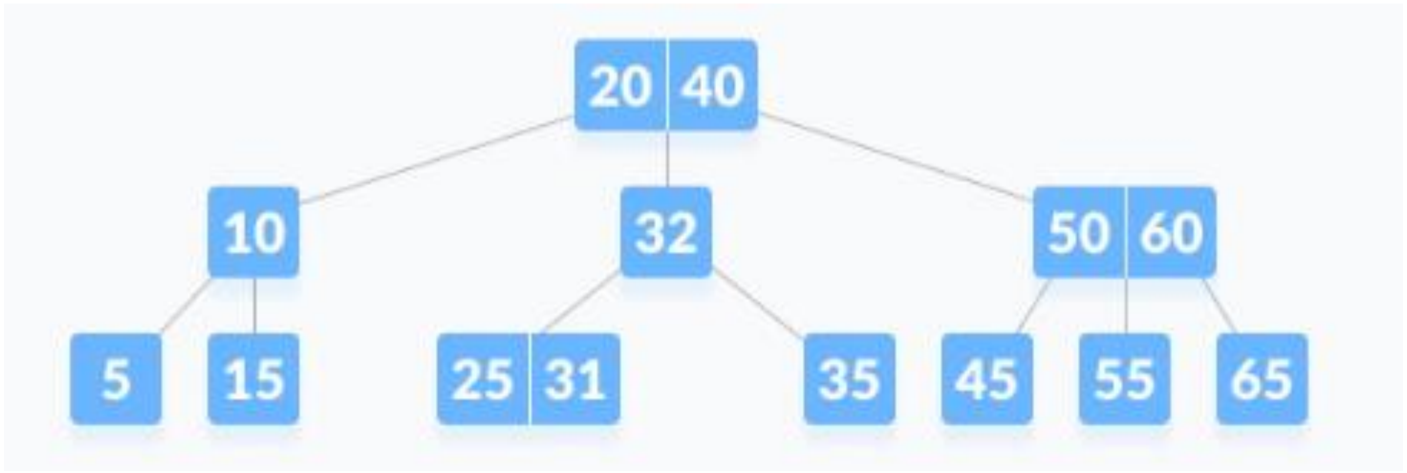
Exercise

- Delete 60.



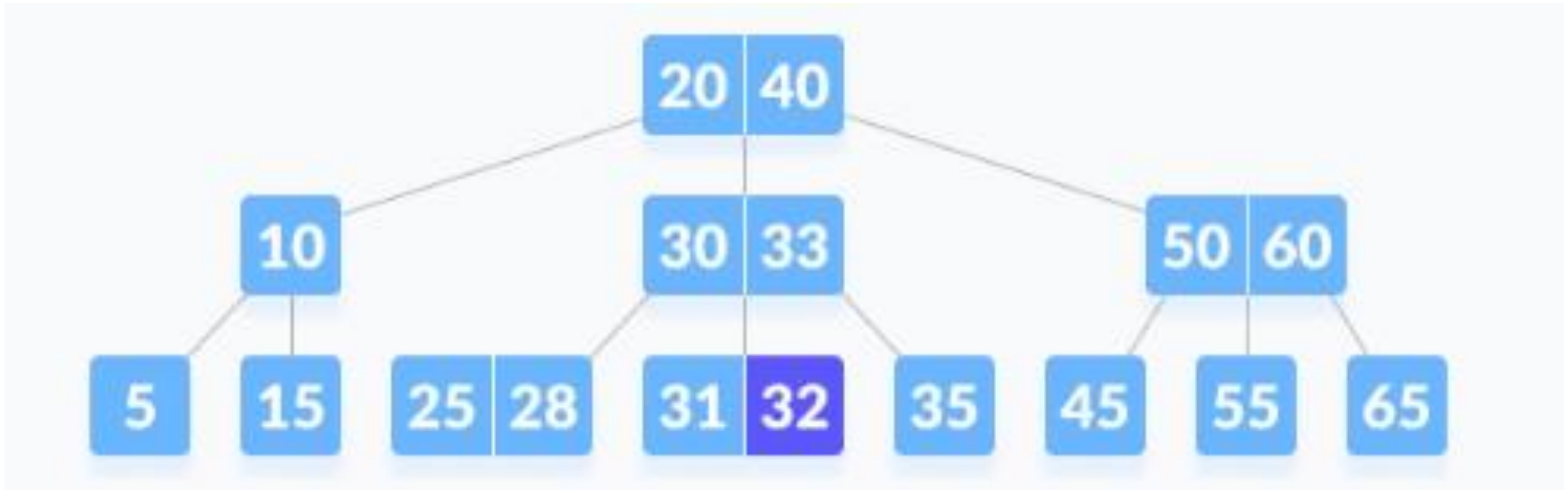
Exercise

- Delete 10.



Exercise

- Delete 5.



B-Tree

- A B-tree of order *‘d’* would be an *(a,b)* tree with *a = ceil(d/2) and b = d.*

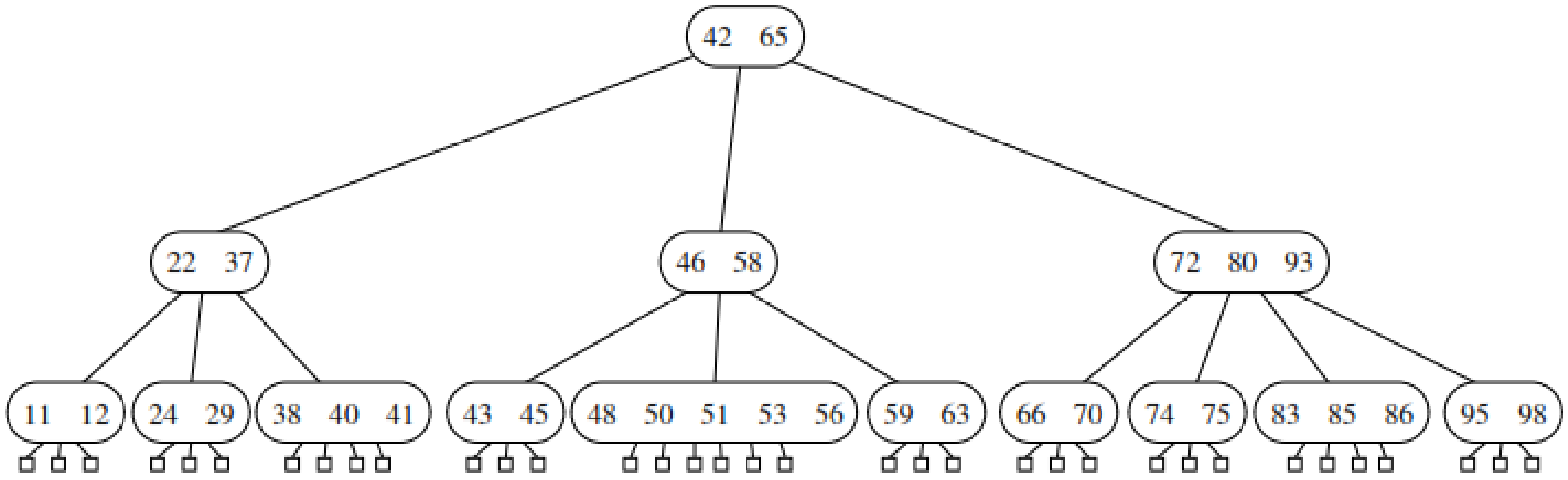


Figure 15.4: A B-tree of order 6.

Exercise

- Which of the following is/are not B-Tree of order 5?

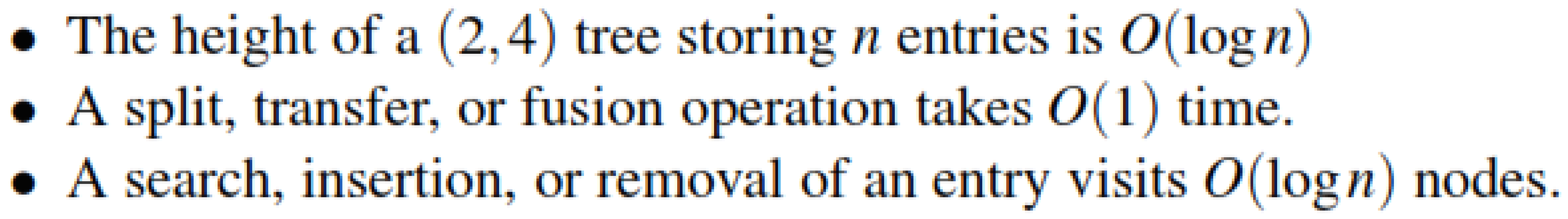
```
      E J O             E J O             E   J   O          E   J   O
    /   / \   \       / /   \   \       /   / \     \      /   / \     \
   ABC   F  KLM  PQRS  ABF   GH   KL  PQRS  AB   FI   KL   PQRS  AB   HI   KL   PQRS
 //\\ /\ //\\//|\\ //\\ /|\ /|\ //|\\  /|\ /|\ /|\ //|\\  /|\ /|\ /|\ //|\\
                                                            FG
                                                            /|\
```

Max number of keys in B-Tree

- What is the MAXIMUM number of KEYS in a B-Tree of order m of height h?

Complexity of find, insertion and deletion

Performance of (2,4) tree



- The height of a $(2,4)$ tree storing n entries is $O(\log n)$
- A split, transfer, or fusion operation takes $O(1)$ time.
- A search, insertion, or removal of an entry visits $O(\log n)$ nodes.

Space utilization is always 50% or above.

B-Tree vs BST/Binary Search

- $O(\log_d n) \Rightarrow O(\log n / \log d)$

Resources

- Data Structures and Algorithms in Python, by Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. 2013. (1st. ed.). Wiley Publishing
- Open Data Structures (pseudocode edition), by Pat Morin. Available online at http://opendatastructures.org
- https://www.programiz.com/dsa/deletion-from-a-b-tree
- https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/btrees.pdf
- https://www.educba.com/b-tree-in-data-structure/
- 2-3-4 Trees | Algorithm Tutor

Thanks