B+ Tree

Anup Halder Joy - 2005005
Afzal Hossan - 2005021
Asif Karim - 2005024

March 13, 2024

Agenda

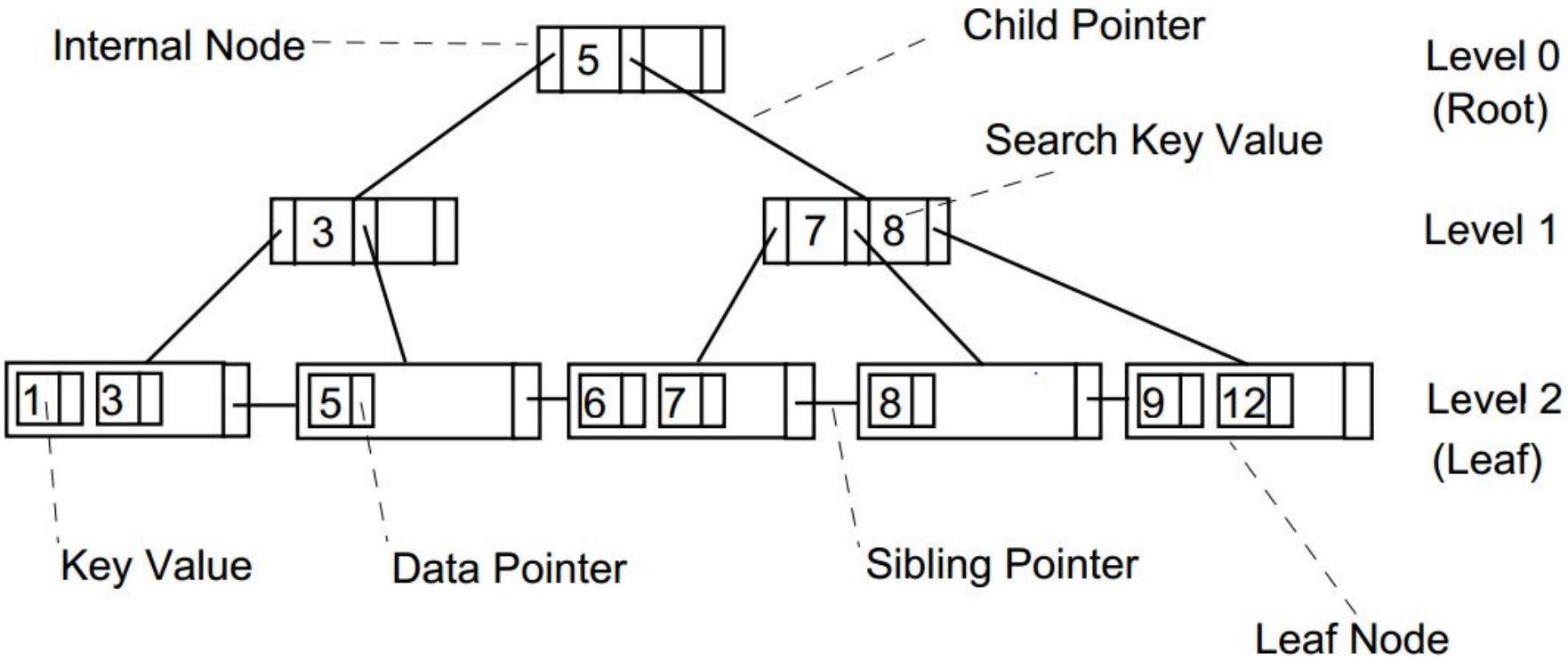


Figure: B+ Tree

WHAT is a B+ Tree?

What is a B+ Tree?

Definition of B+ Tree

A B+ tree is an m-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. [3].

A B+ Tree is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions

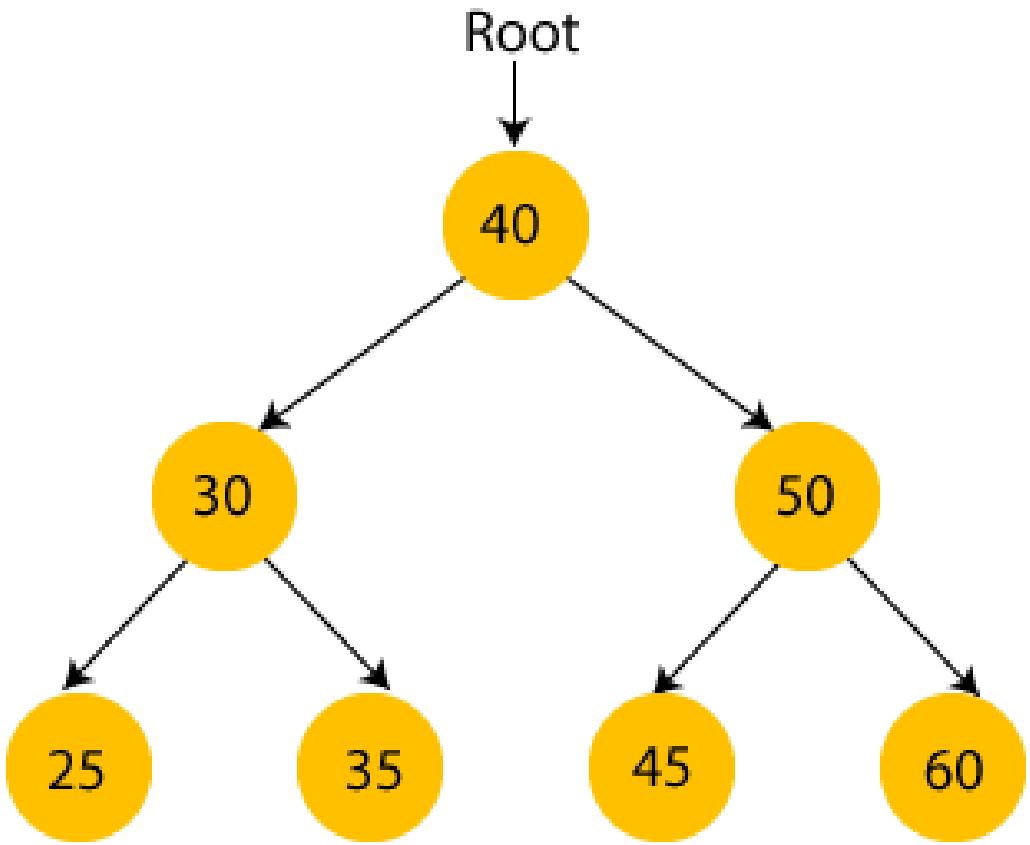
Inventors

Rudolf Bayer

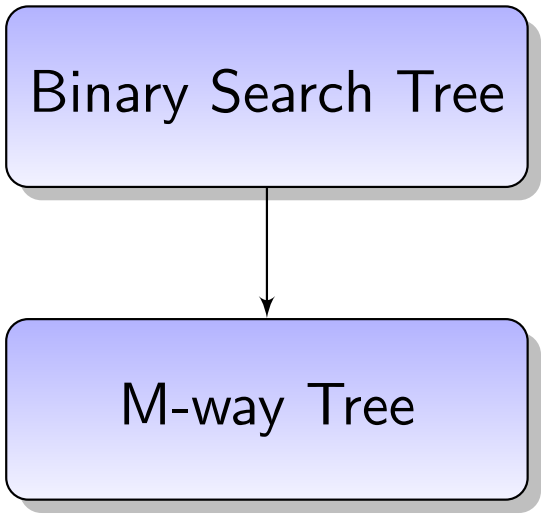
Edward M. McCreight

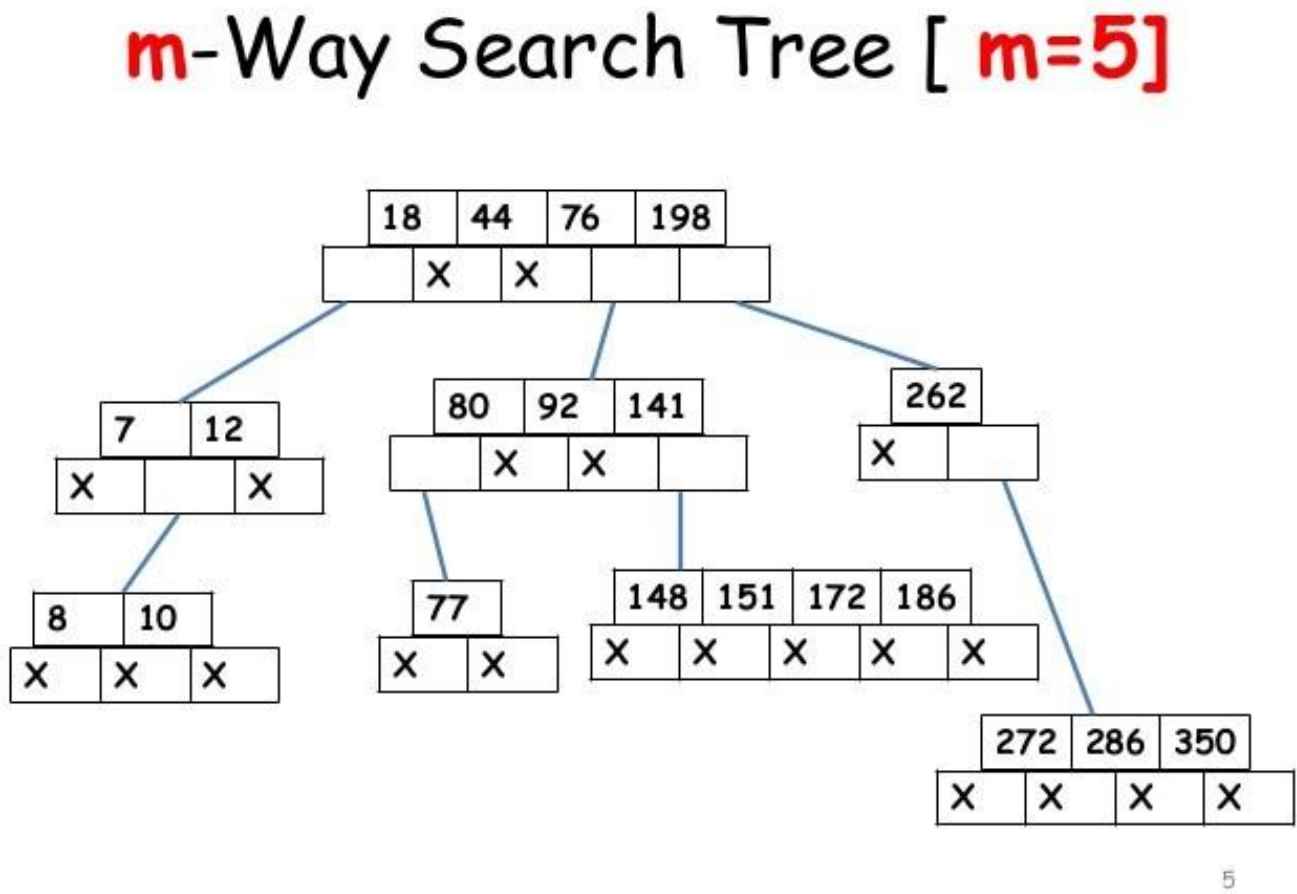
Evolution of B+ Tree

Binary Search Tree

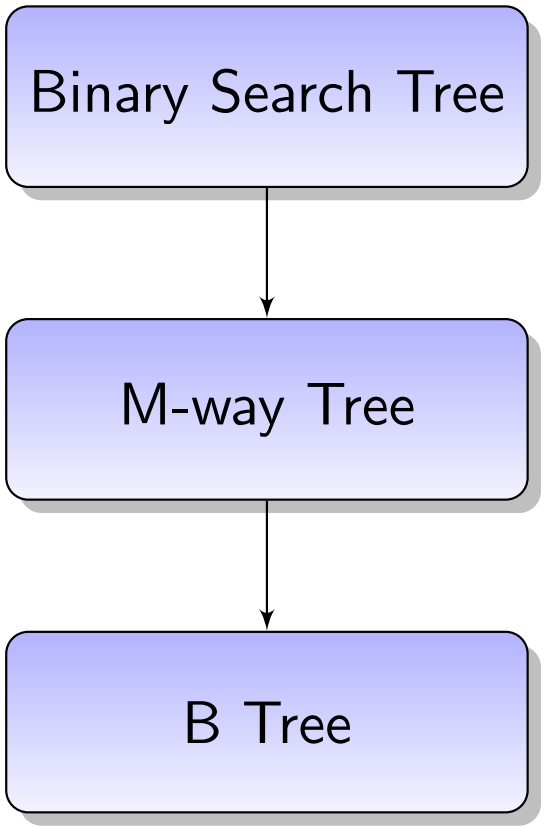


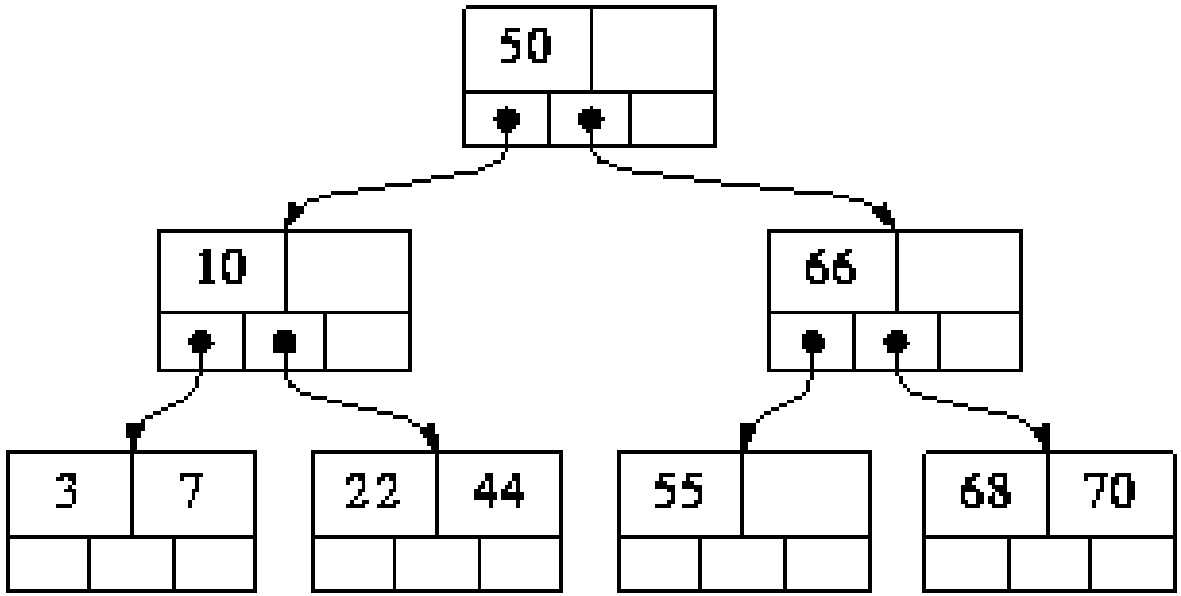
Evolution of B+ Tree



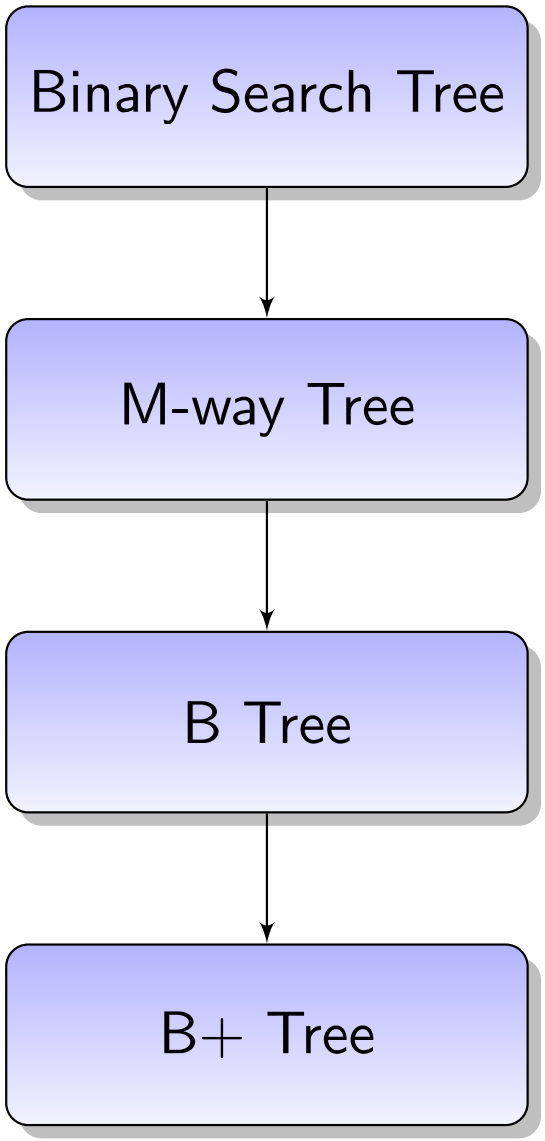


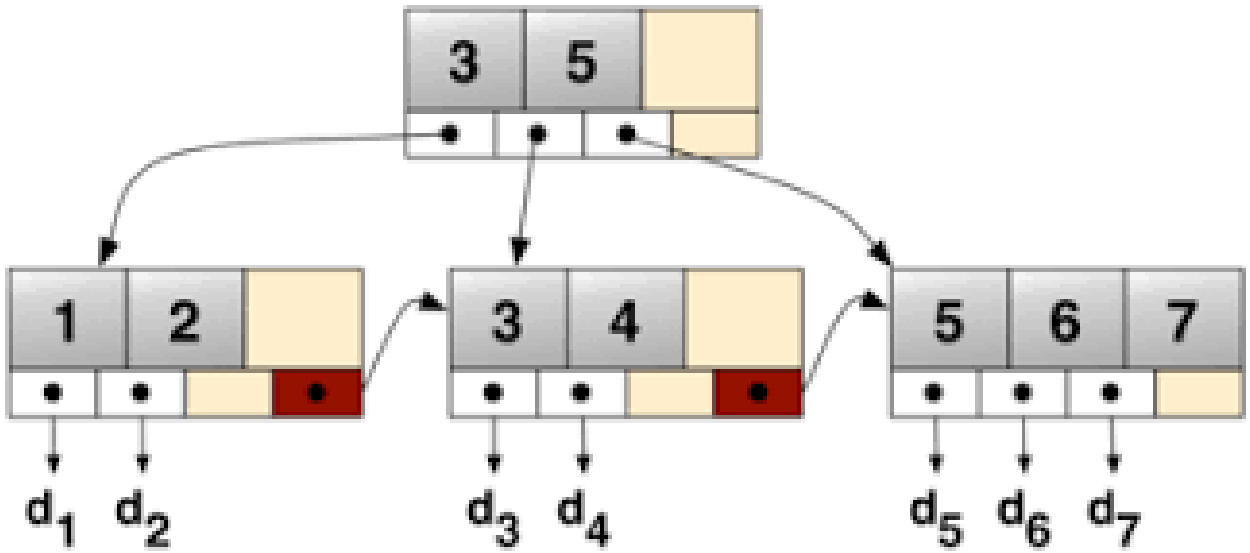
Evolution of B+ Tree





Evolution of B+ Tree





Structure of Internal Node

‘

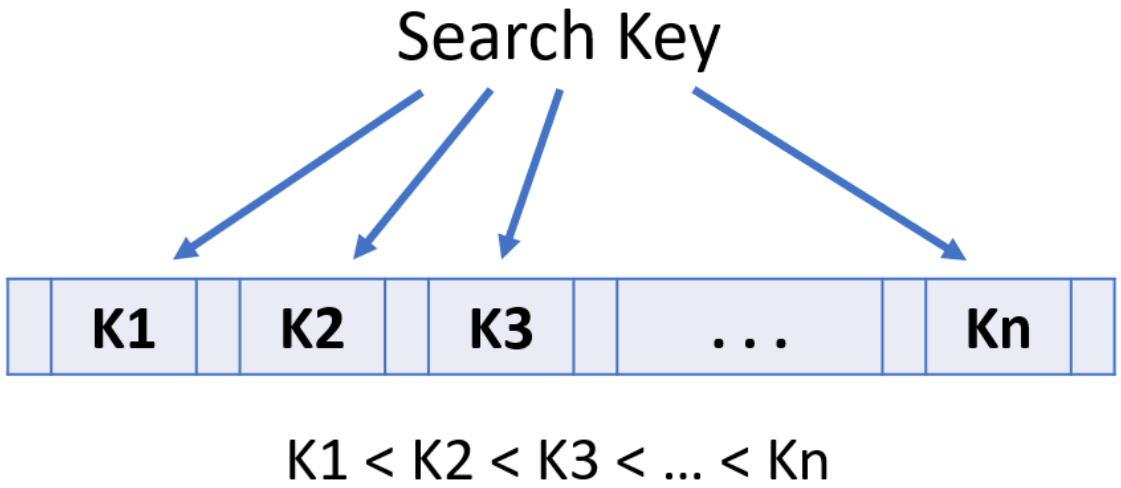


Figure: Internal Node Structure

Structure of Leaf Node

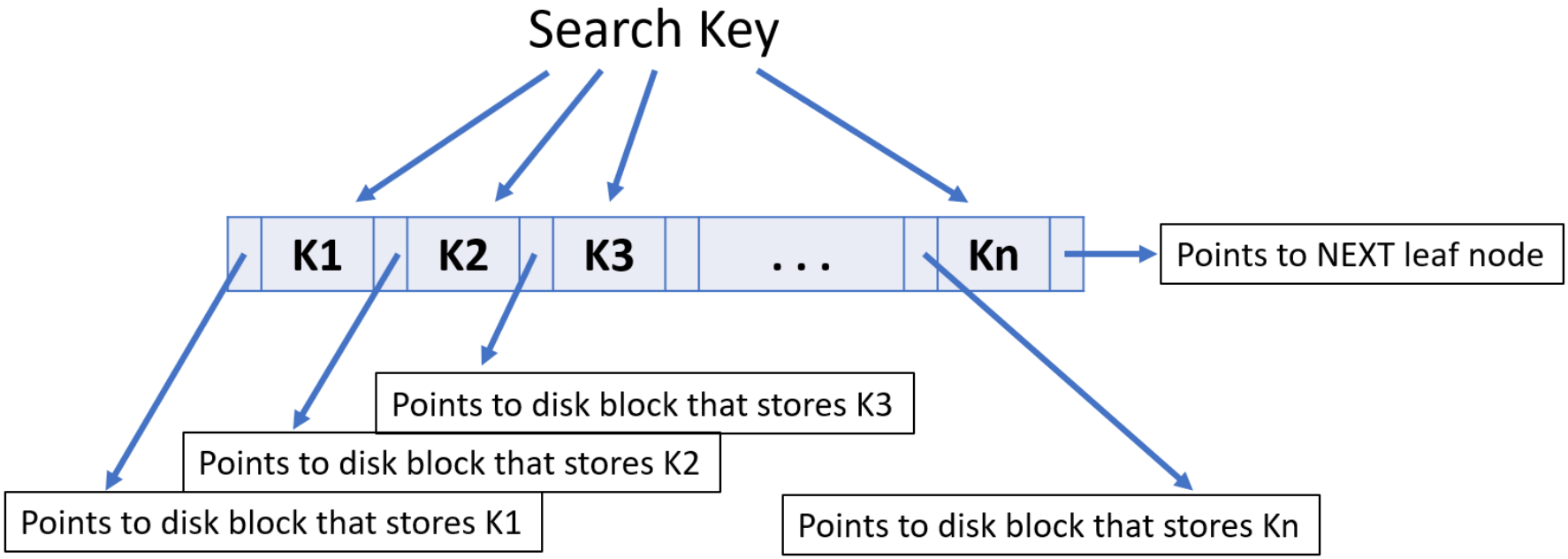


Figure: Leaf Node Structure

Node Bounds

Node Type	**Min #Keys**	**Max #Keys**	**Min #Child**	**Max #Child**
Root Node	1	$M-1$	2[3]	M
Internal Node	$\lceil \frac{M}{2} \rceil - 1$	$M-1$	$\lceil \frac{M}{2} \rceil$	M
Leaf Node	$\lceil \frac{M}{2} \rceil - 1$	$M-1$	0	0

M = Order of B+ Tree

Operations on B+ Tree

HOW do We Operate on B+ Tree?

Insertion

B+ Tree Insertion

How to insert

Since every element is inserted into the leaf node, go to the appropriate leaf node. Insert the key into the leaf node.

Case I:

If the leaf is not full, insert the key into the leaf node in increasing order.

B+ Tree Insertion

How to insert

Since every element is inserted into the leaf node, go to the appropriate leaf node. Insert the key into the leaf node.

Case I:

If the leaf is not full, insert the key into the leaf node in increasing order.

Case II:

1. If the leaf is full, insert the key into the leaf node in increasing order and balance the tree in the following way.
2. Break the node at $\frac{m}{2}$ th position.
3. Add $\frac{m}{2}$ th key to the parent node as well.
4. If the parent node is already full, follow steps 2 to 3.

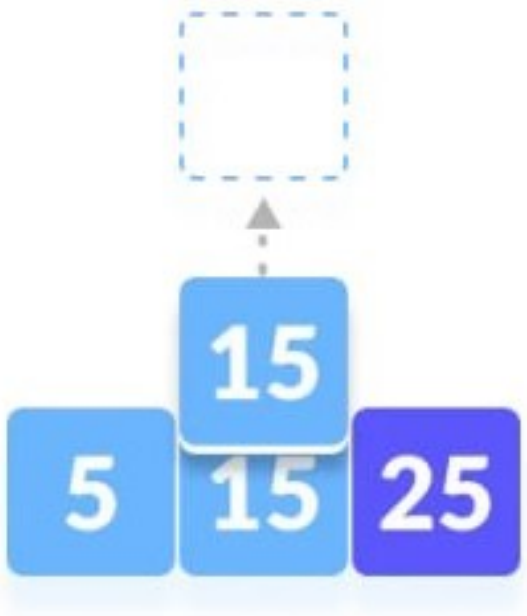
B+ Tree Insertion Animation : Insert 5

Order = 3

B+ Tree Insertion Animation : Insert 15

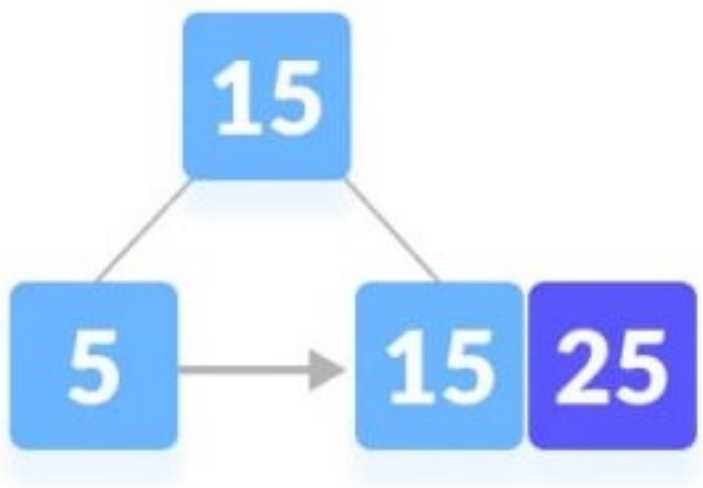
B+ Tree Insertion Animation : Insert 25

B+ Tree Insertion Animation : Insert 25

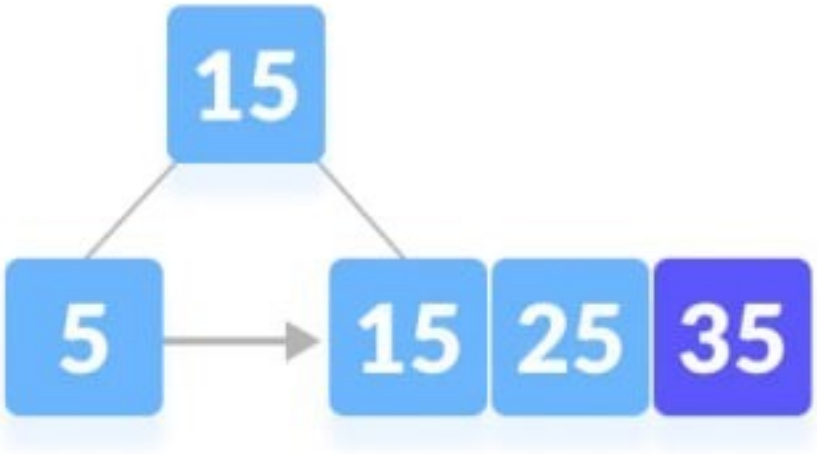


B+ Tree Insertion Animation : Insert 25

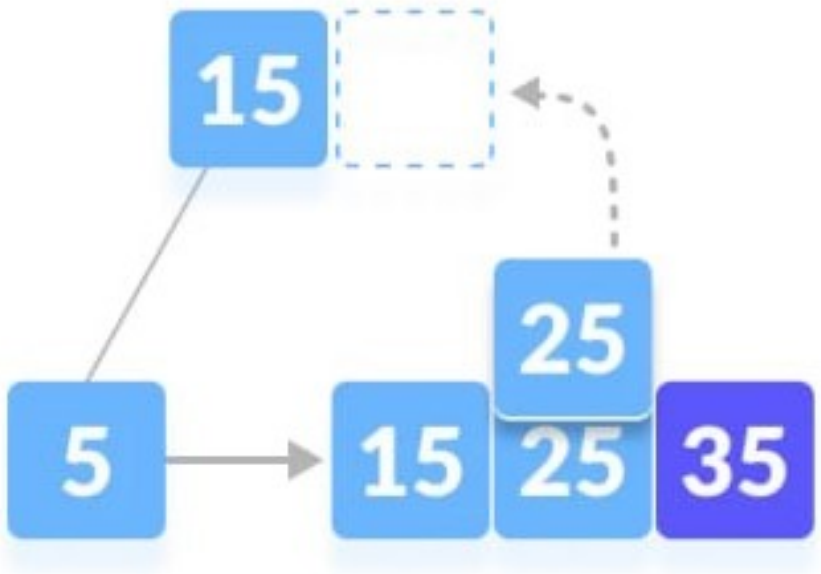
B+ Tree Insertion Animation : Insert 25



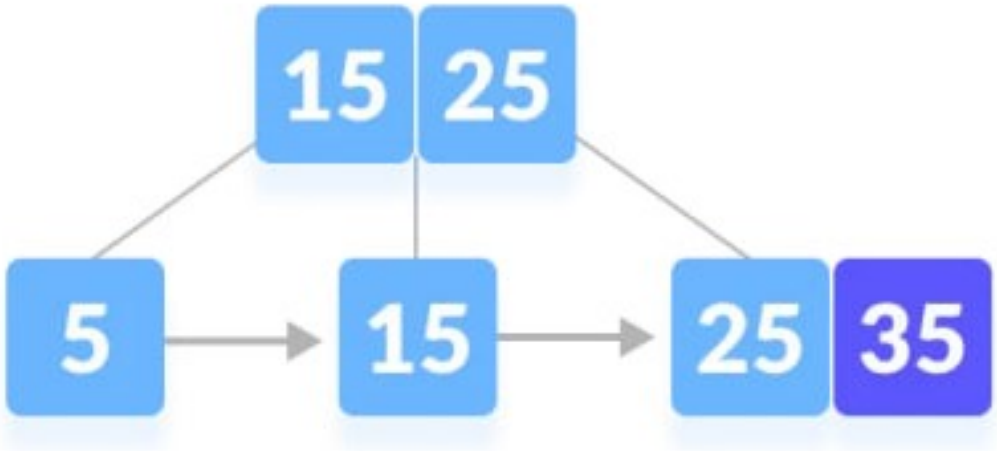
B+ Tree Insertion Animation : Insert 35



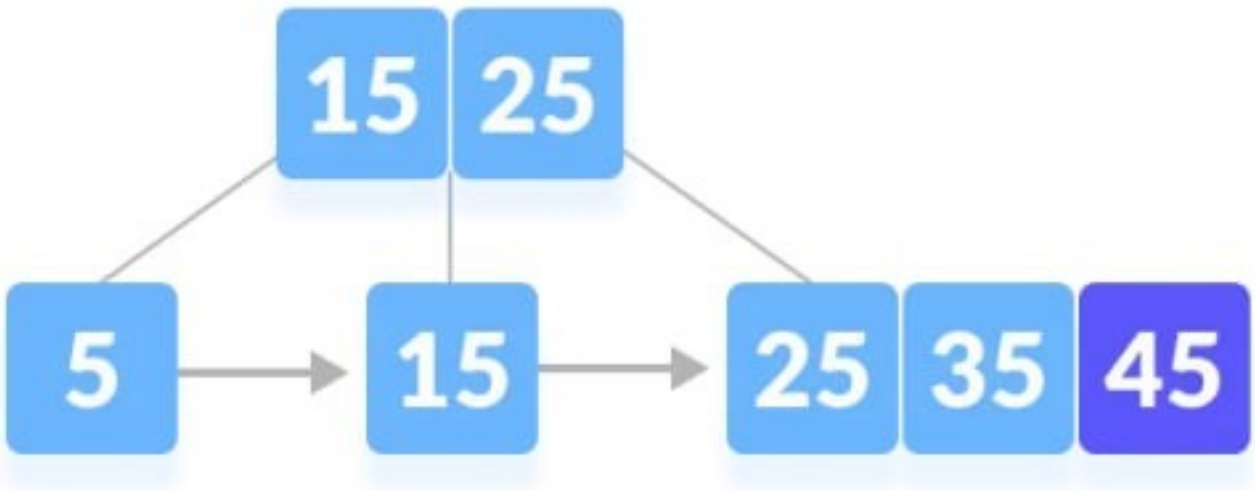
B+ Tree Insertion Animation : Insert 35



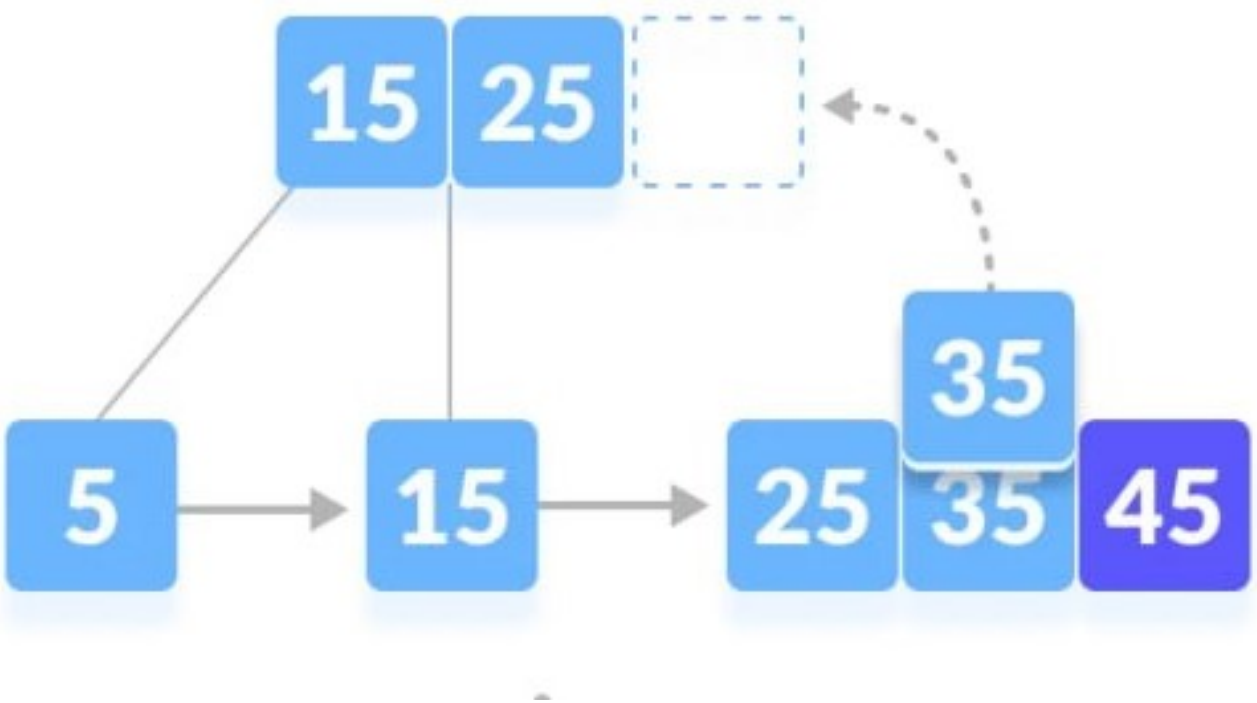
B+ Tree Insertion Animation : Insert 35



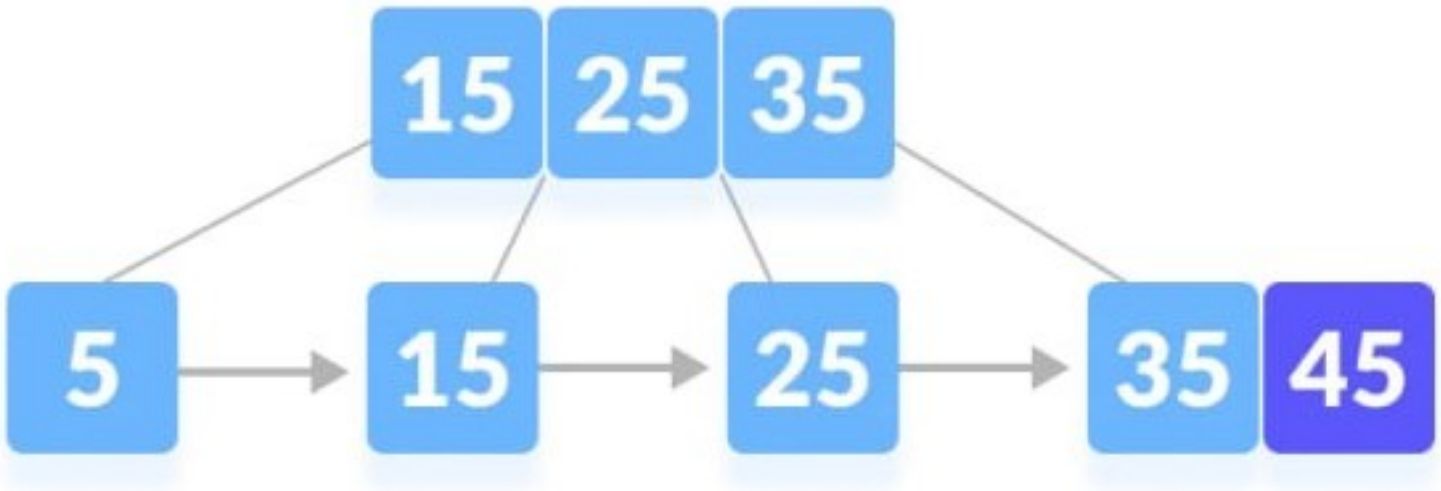
B+ Tree Insertion Animation : Insert 45



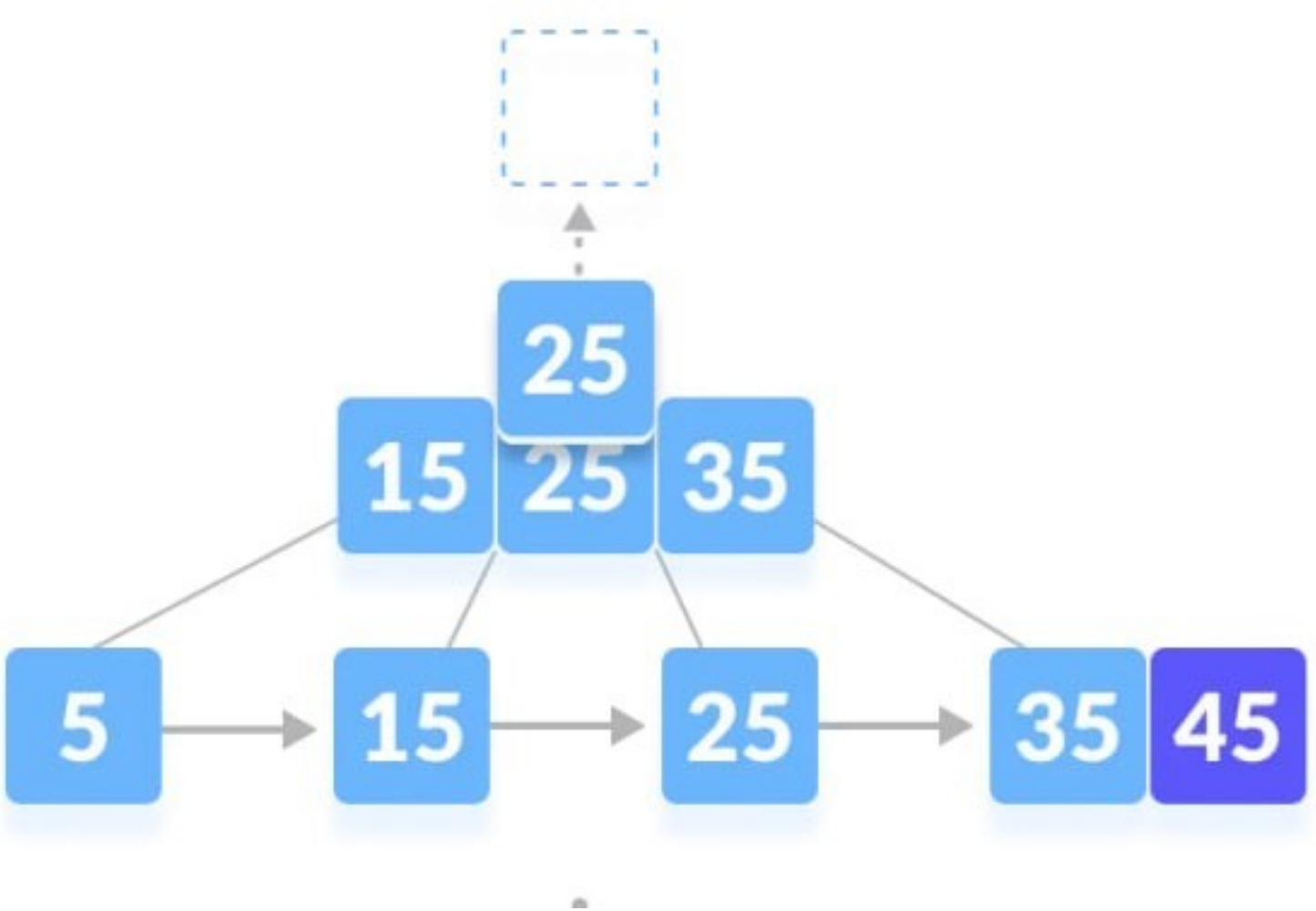
B+ Tree Insertion Animation : Insert 45



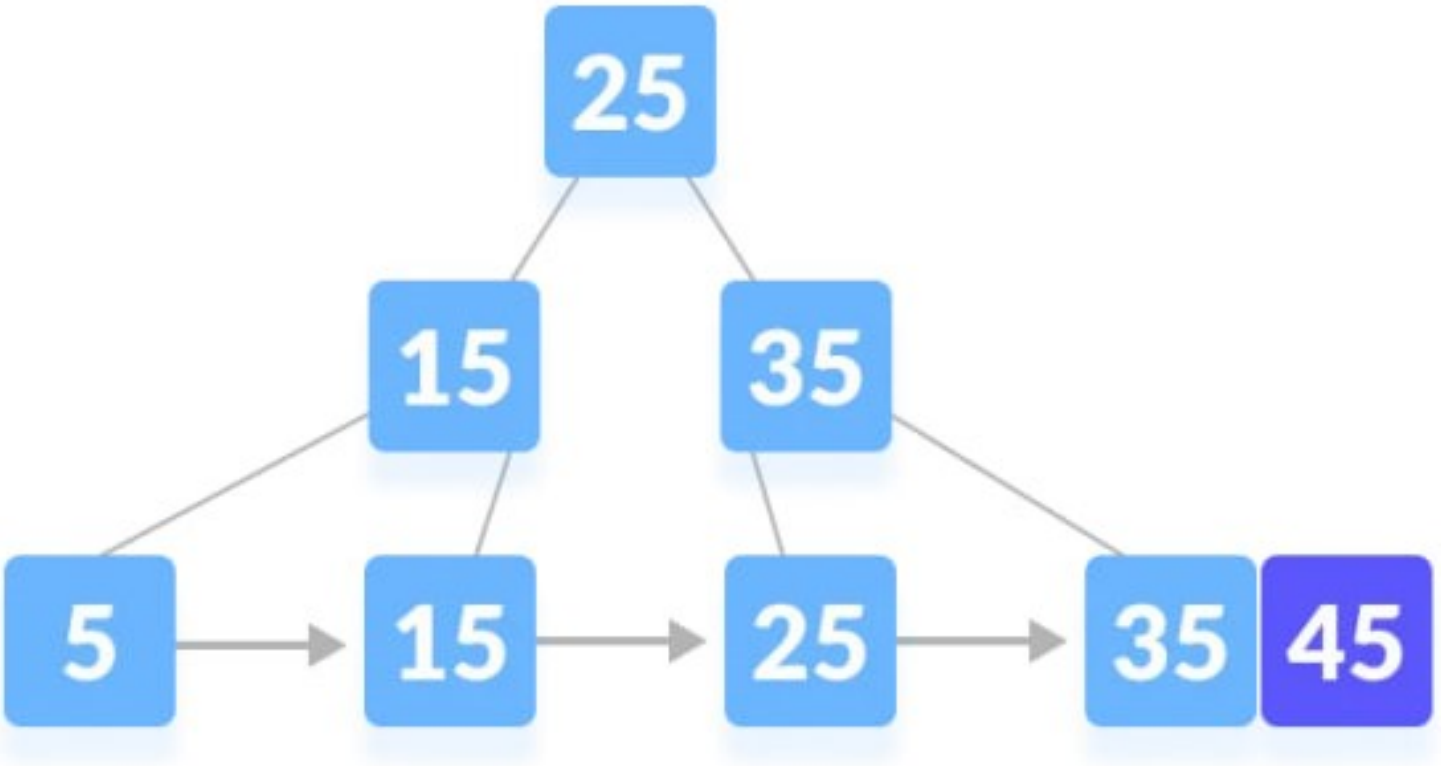
B+ Tree Insertion Animation : Insert 45



B+ Tree Insertion Animation : Insert 45



B+ Tree Insertion Animation : Insert 45



Deletion

B+ Tree Deletion

Before going through the steps below, one must know these facts about a B+ tree of degree m. A node can have -

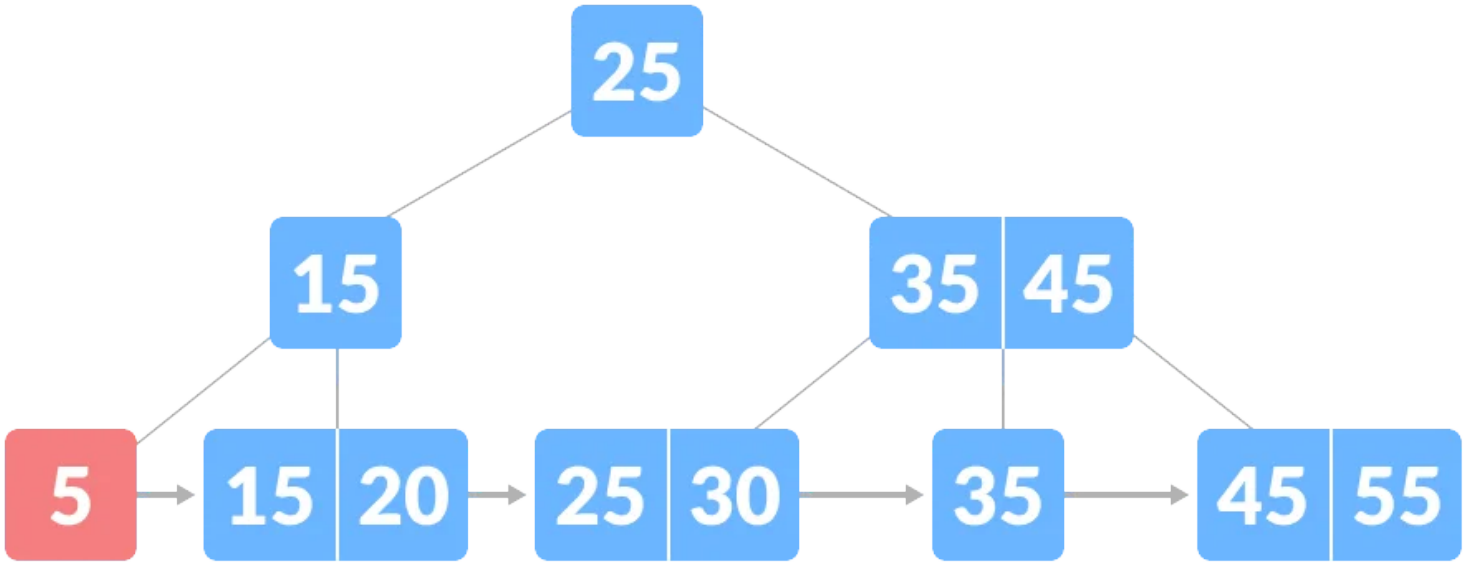
1. a maximum of m children. (i.e. 3)
2. a maximum of m - 1 keys. (i.e. 2)
3. a minimum of $\lceil m/2 \rceil$ children. (i.e. 2)
4. a minimum of $\lceil m/2 \rceil - 1$ keys (except root node). (ie 1)

* In our example m $= 3$

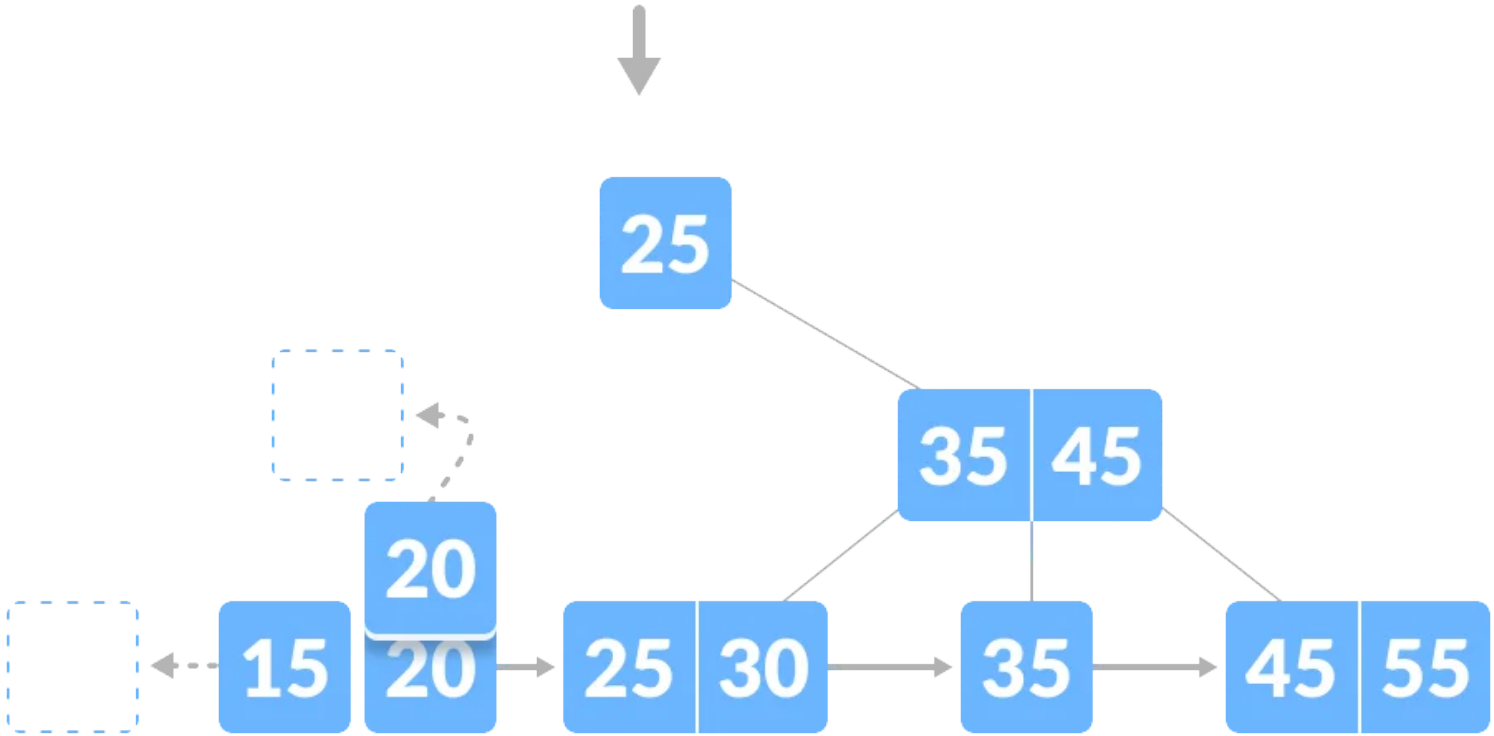
B+ Tree Deletion Animation

Case 1

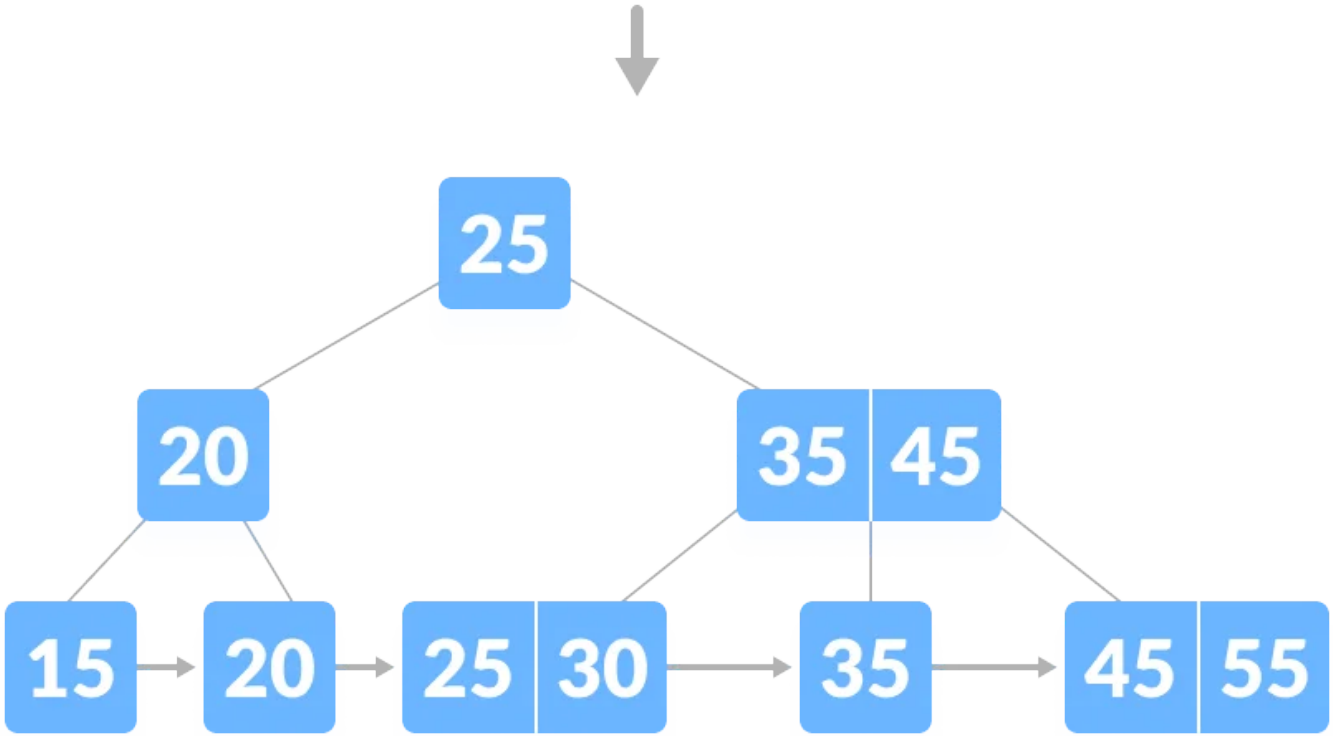
The key to be deleted is present only in the leaf node and not in the internal nodes. For instance 5.



B+ Tree Deletion Animation



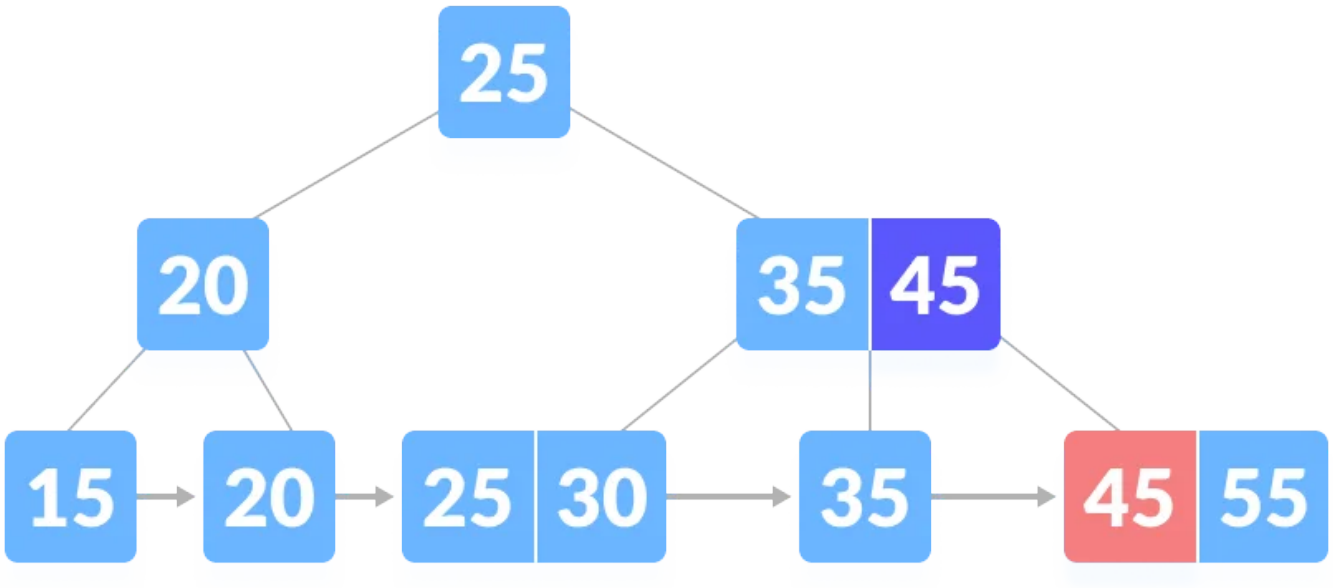
B+ Tree Deletion Animation



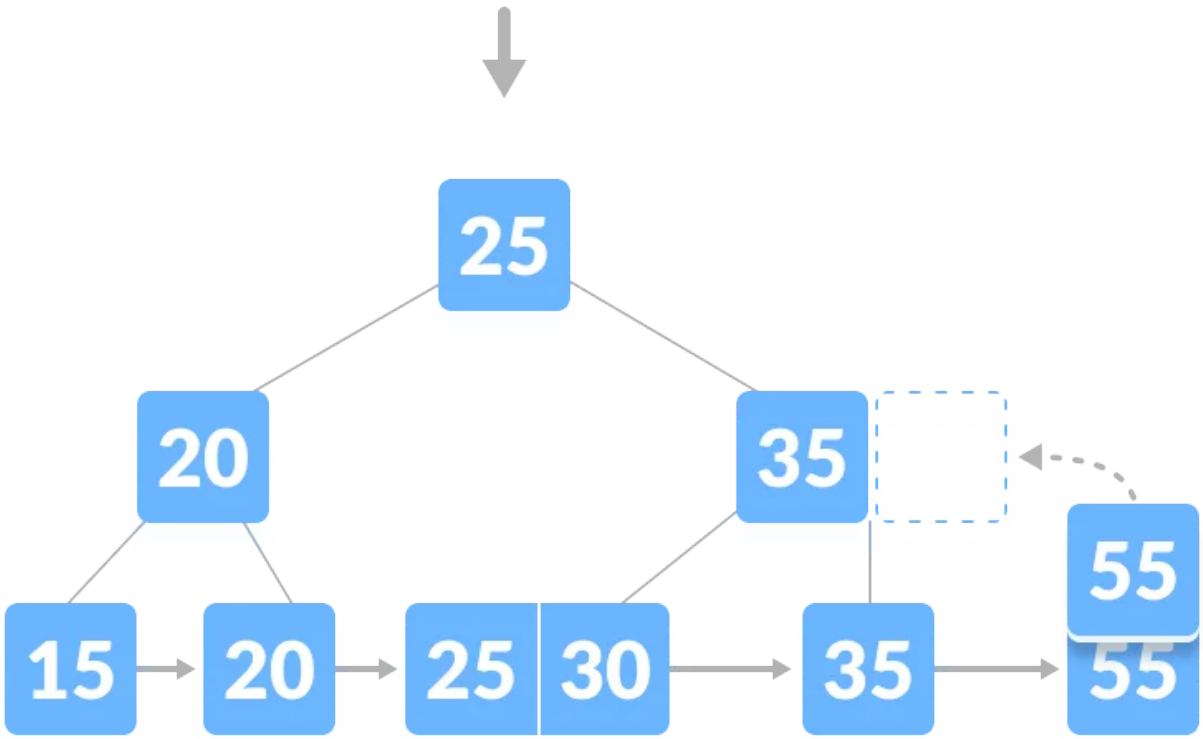
B+ Tree Deletion Animation

Case 2

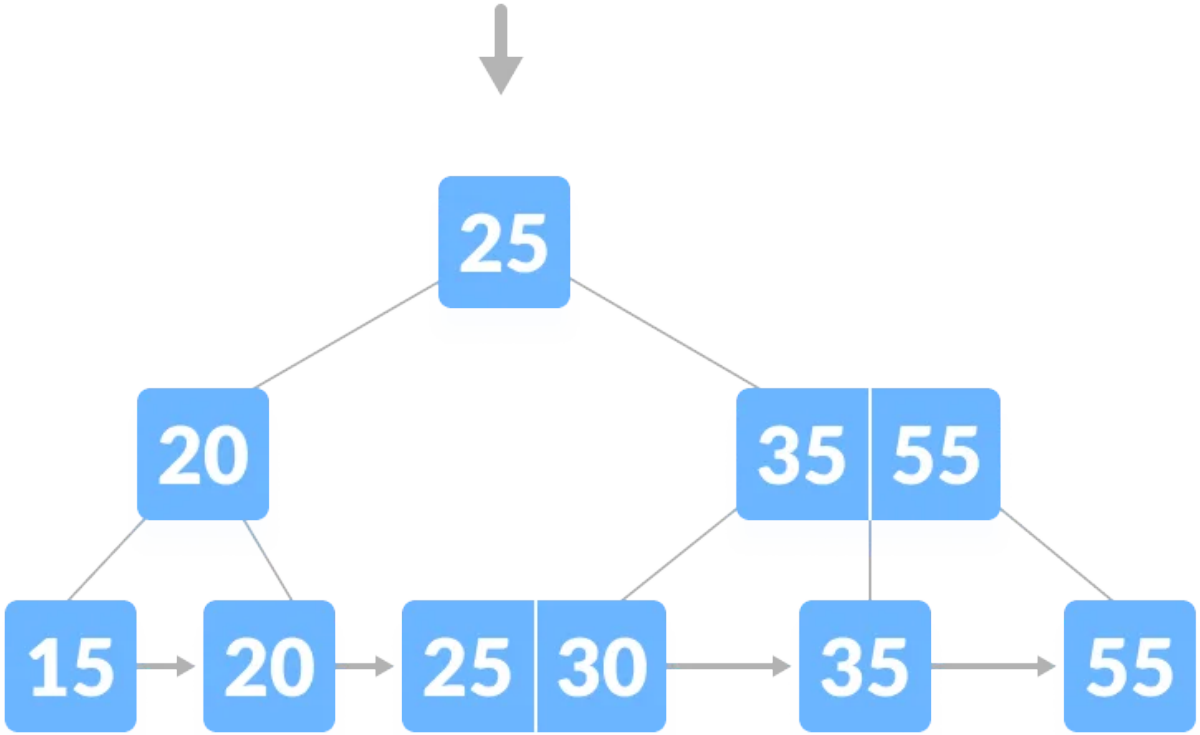
The key to be deleted is present in the internal nodes as well. For example, 45.



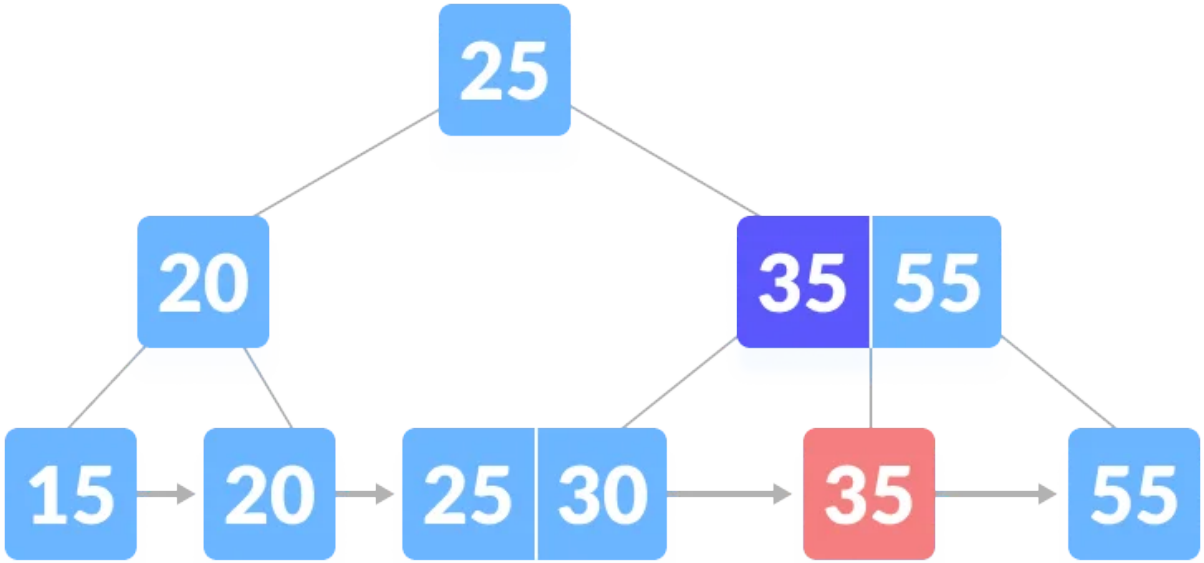
B+ Tree Deletion Animation



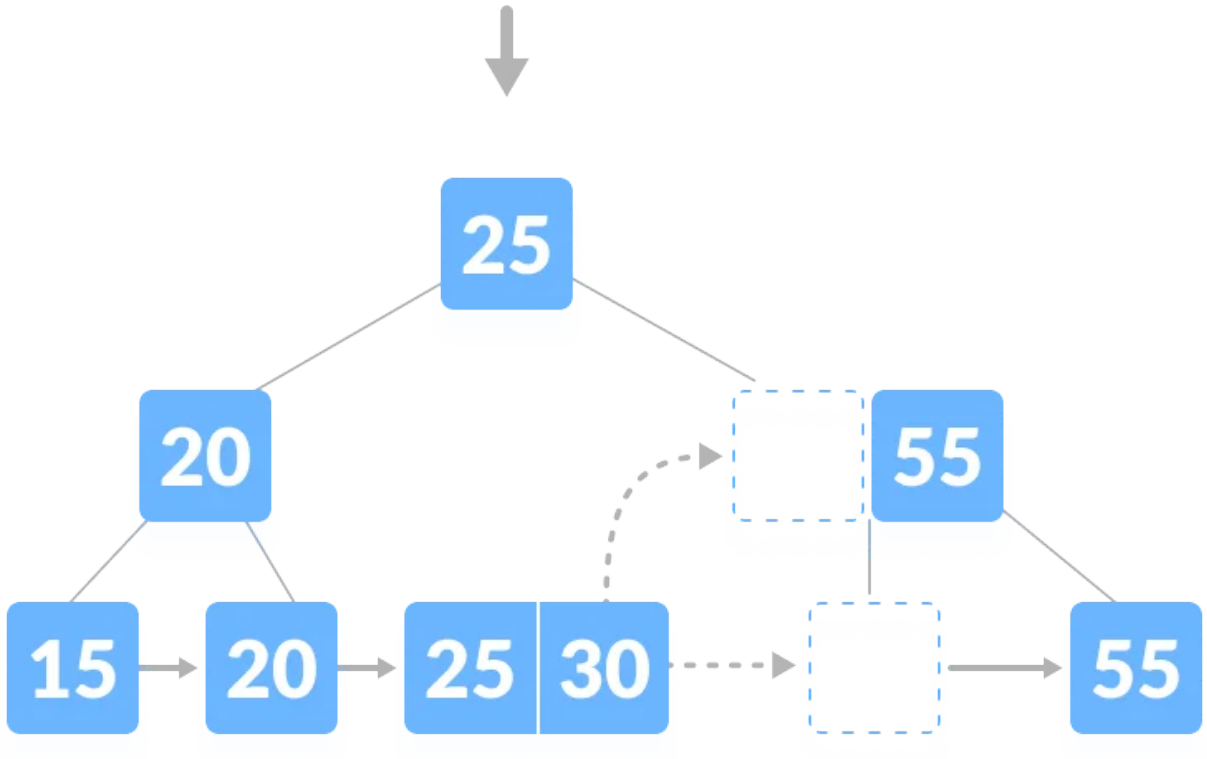
B+ Tree Deletion Animation



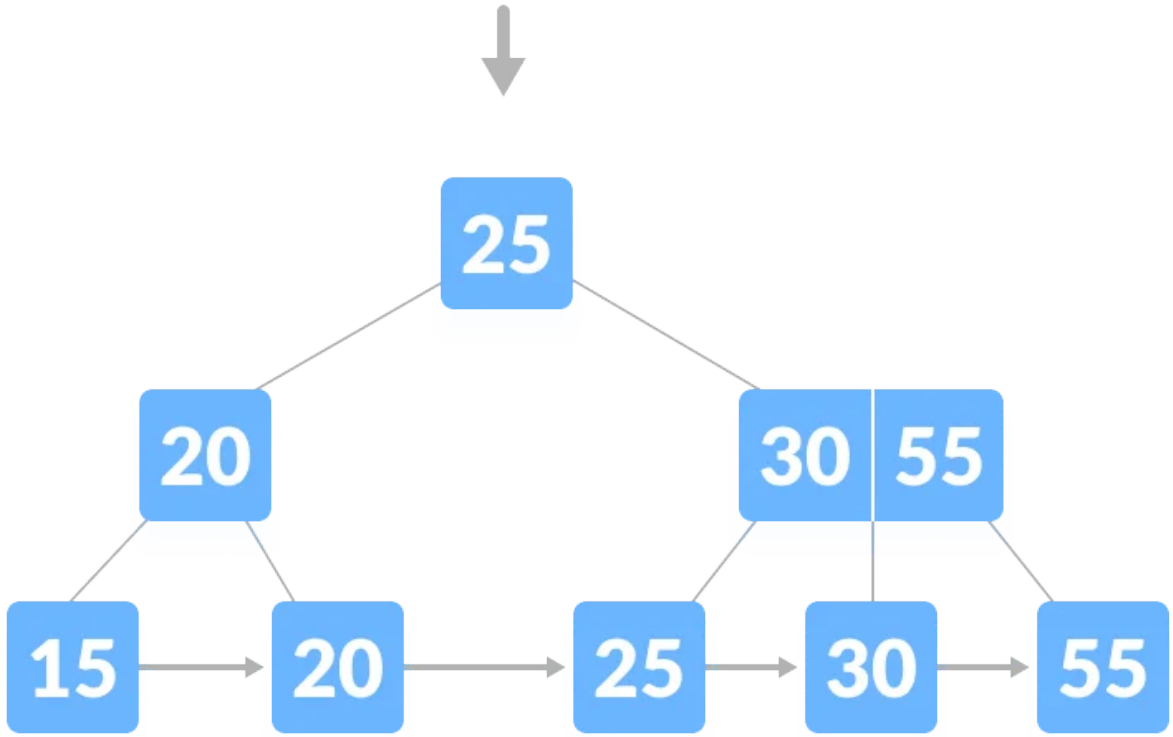
B+ Tree Deletion Animation



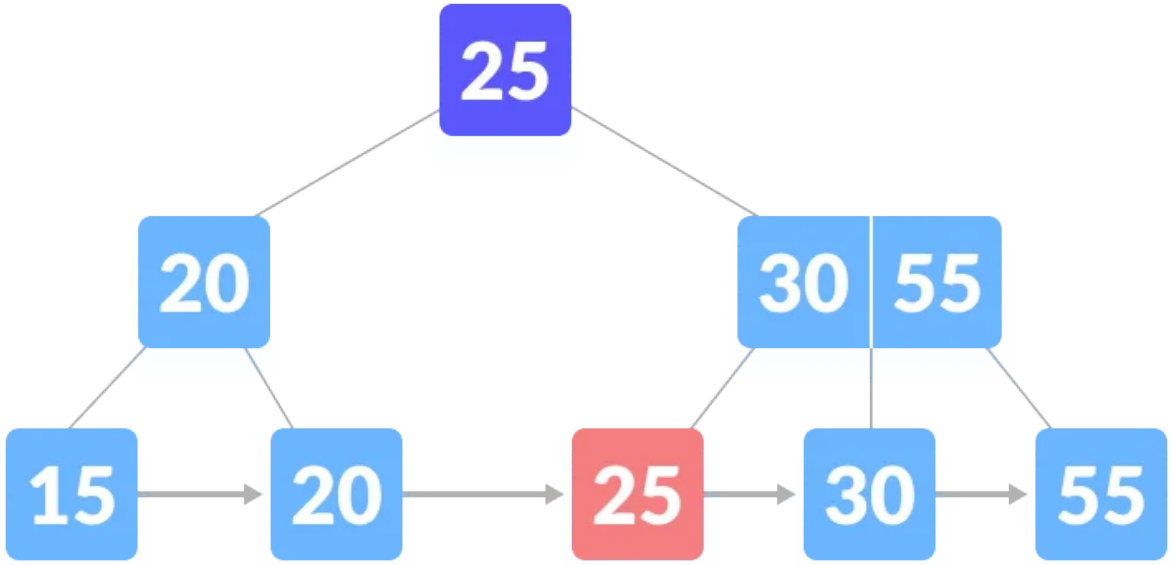
B+ Tree Deletion Animation



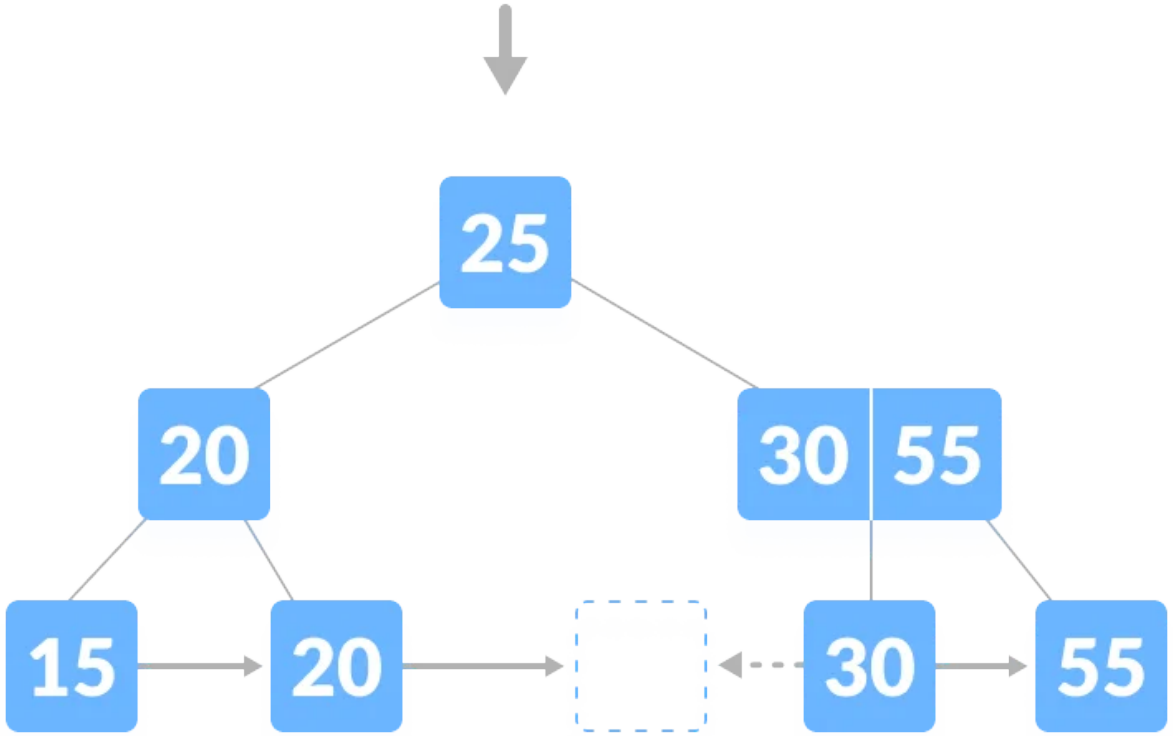
B+ Tree Deletion Animation



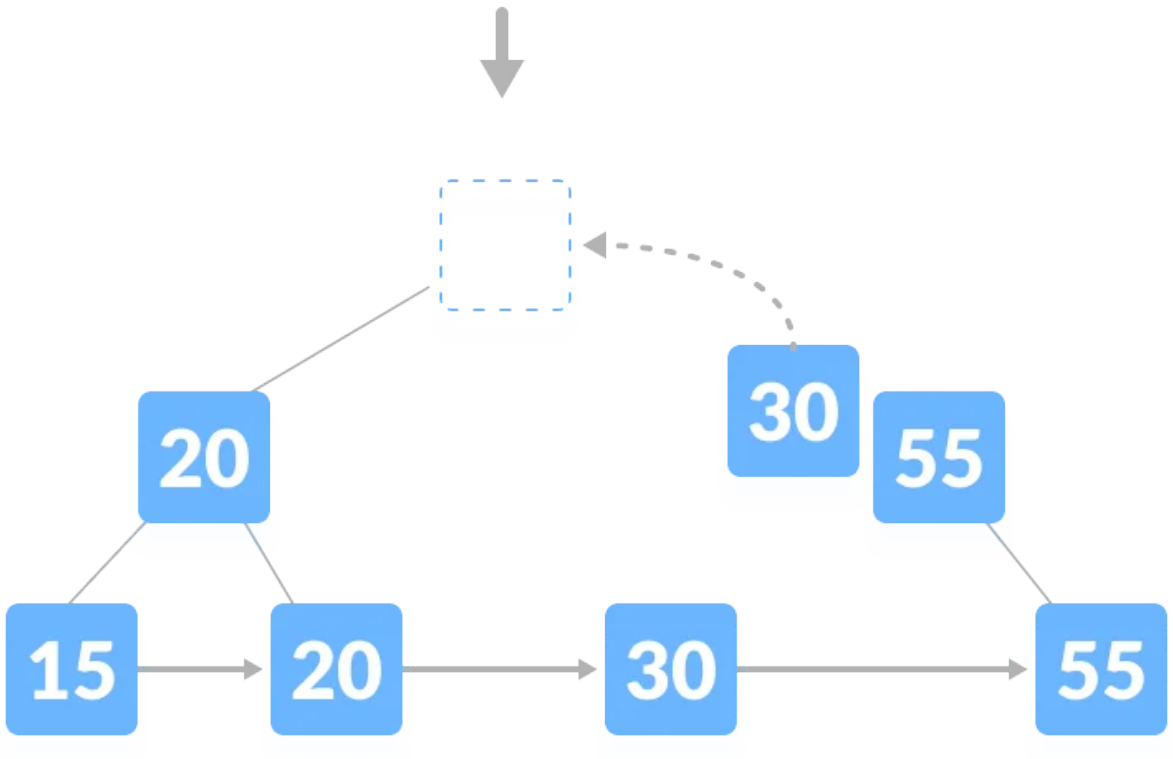
B+ Tree Deletion Animation



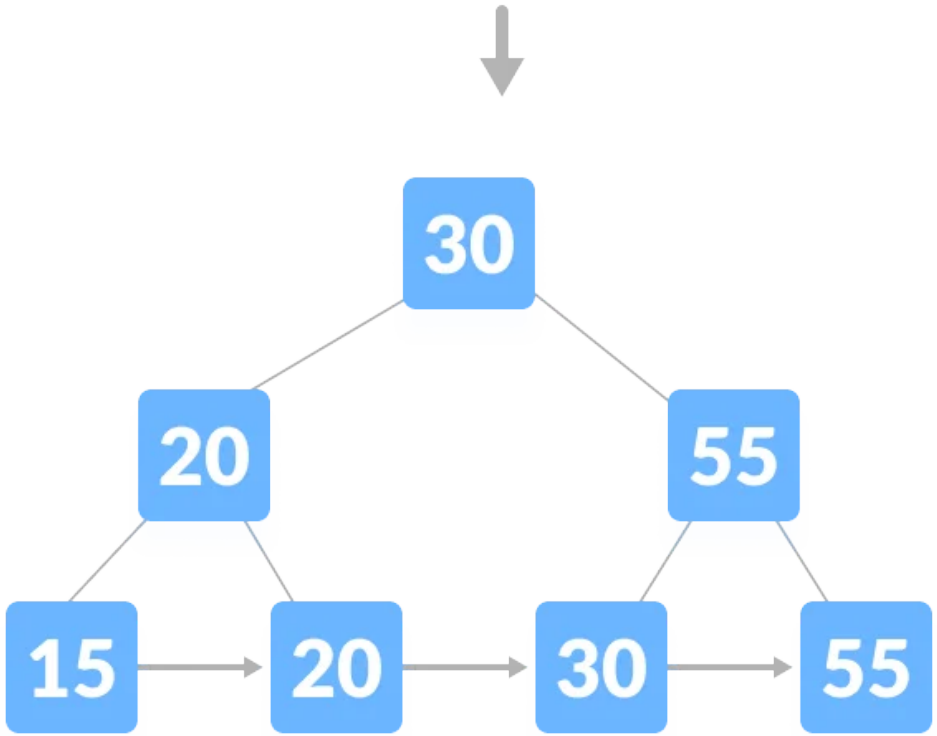
B+ Tree Deletion Animation



B+ Tree Deletion Animation



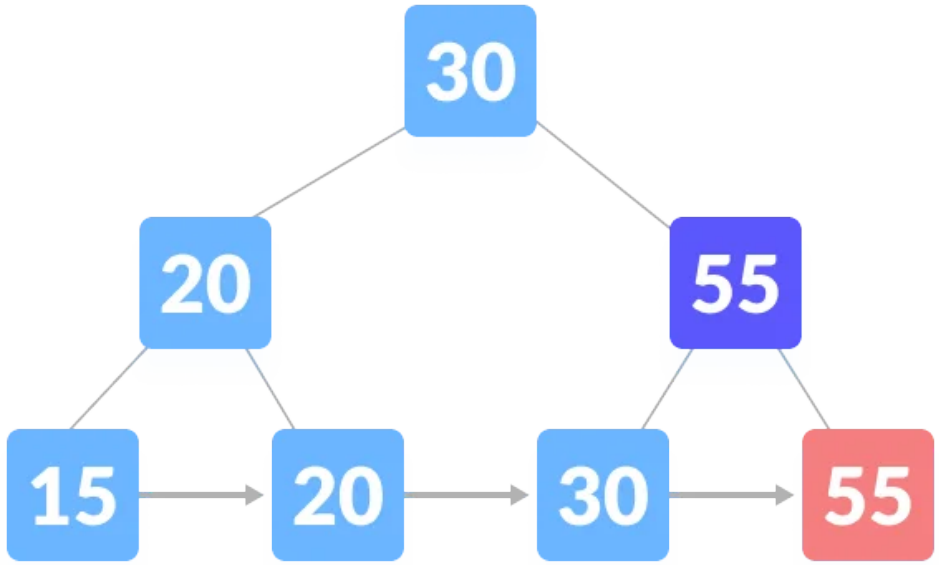
B+ Tree Deletion Animation



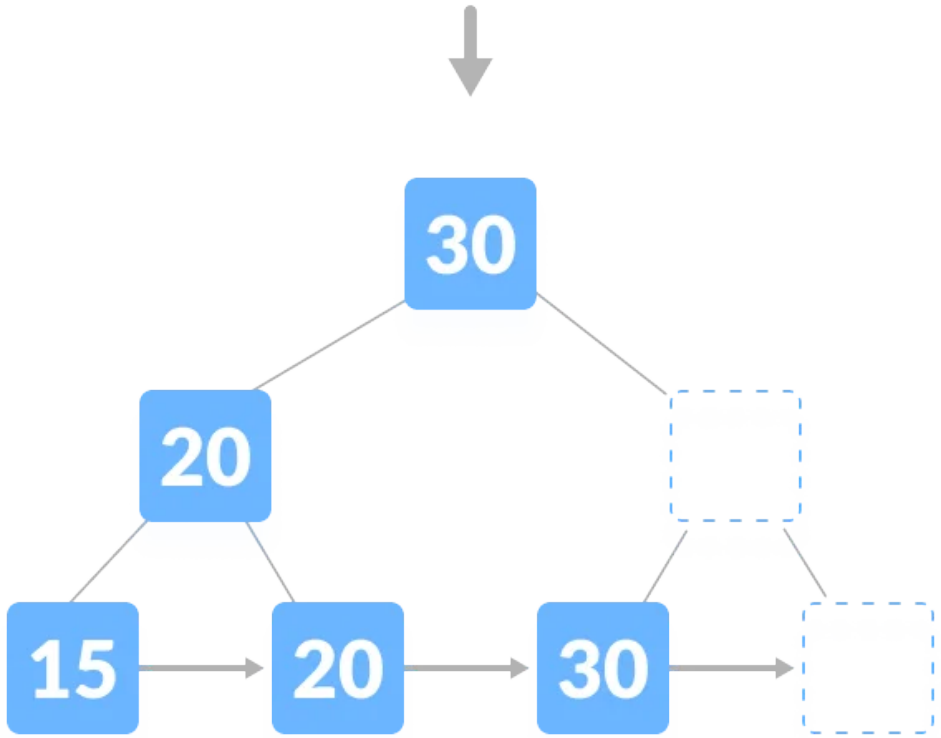
B+ Tree Deletion Animation

Case 3

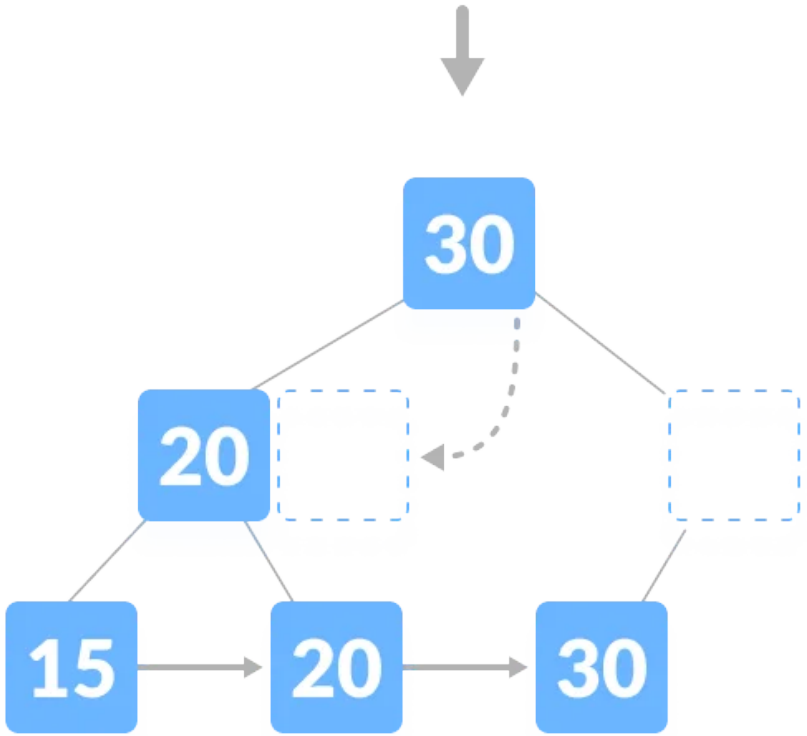
The height of the tree gets shrinked. For example deletion of 55.



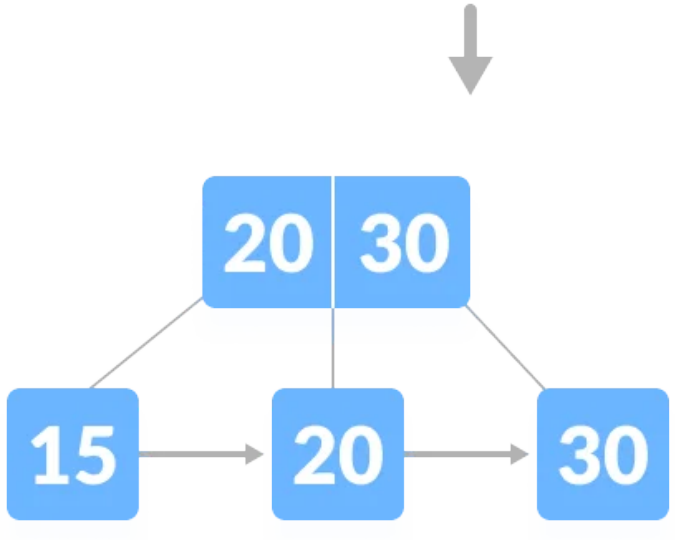
B+ Tree Deletion Animation



B+ Tree Deletion Animation



B+ Tree Deletion Animation



Importance of B+ Tree

WHY do We Actually Need B+ Tree?

Time Analysis

B+ Tree Operation	Time Complexity
Insertion	$O(\log_m n)$

m = Order of B+ Tree

n = Number of elements in B+ Tree

Time Analysis

B+ Tree Operation	Time Complexity
Insertion	$O(\log_m n)$
Deletion	$O(\log_m n)$

m = Order of B+ Tree
n = Number of elements in B+ Tree

Time Analysis

B+ Tree Operation	Time Complexity
Insertion	$O(\log_m n)$
Deletion	$O(\log_m n)$
Search	$O(\log_m n)$

m = Order of B+ Tree
n = Number of elements in B+ Tree

Advantages of B+ Trees

Advantages of B+ Trees

Ordered Structure for Sequential Access

B+ trees maintain data in sorted order, facilitating efficient sequential access to keys. [2].

Advantages of B+ Trees (Cont'd)

Concurrency Control and Performance

B+ trees supports locking and MVCC to ensure consistency and isolation among concurrent transactions [1].

Advantages of B+ Trees (Cont'd)

Optimal Disk Access and Cache Efficiency

B+ trees utilize node-based structures and node sizes that align well with the block size of storage devices, minimizing disk access and maximizing cache hits [4].

Real Life Application of B+ Tree

WHERE do We Use B+ Trees in Real Life?

Real Life Application of B+ Tree

Real Life Application of B+ Tree

Database Indexing

B+ trees enable effective retrieval of data based on indexed characteristics, resulting in faster query execution times.

Real Life Application of B+ Tree

Web Browsers

B+ trees are employed in web browsers for managing bookmarks and history data.

Real Life Application of B+ Tree

Geo-spatial Databases

B+ trees are well-suited for indexing geo-spatial data such as coordinates, shapes, and spatial relationships.

References I


Rakesh Agrawal, Michael J Carey, and Miron Livny.
Concurrency control performance modeling: Alternatives and implications.
ACM Transactions on Database Systems (TODS), 12(4):609–654, 1987.


Hector Garcia-Molina, Jeffrey D Ullman, and Jennifer Widom.
Database system implementation, volume 672.
Prentice Hall Upper Saddle River, 2000.


Shamkant B. Navathe, Ramez Elmasri.
Fundamentals of Database Systems.
Pearson Education, Upper Saddle River, N.J., 6th edition, 2010.


Patrick E. O'Neil, Elizabeth J. O'Neil, and Gerhard Weikum.
Modern b-tree techniques.
ACM Computing Surveys (CSUR), 24(2):123–160, 1992.

End of Presentation

Thank You!