

[Home](#)[Data Structure](#)[C](#)[C++](#)[C#](#)[Java](#)[SQL](#)[HTML](#)[CSS](#)[JavaScript](#)[Ajax](#)[↑ SCROLL TO TOP](#)

# B Tree

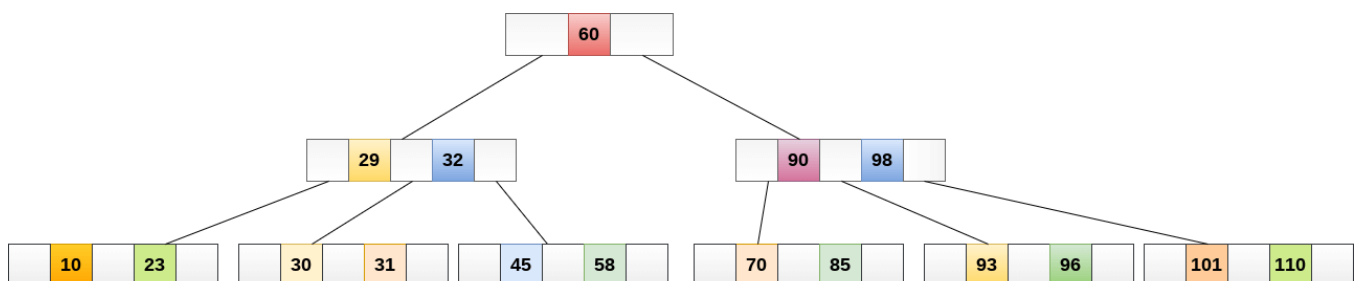
B Tree is a specialized m-way tree that can be widely used for disk access. A B-Tree of order m can have at most m-1 keys and m children. One of the main reason of using B tree is its capability to store large number of keys in a single node and large key values by keeping the height of the tree relatively small.

A B tree of order m contains all the properties of an M way tree. In addition, it contains the following properties.

1. Every node in a B-Tree contains at most m children.
2. Every node in a B-Tree except the root node and the leaf node contain at least  $m/2$  children.
3. The root nodes must have at least 2 nodes.
4. All leaf nodes must be at the same level.

It is not necessary that, all the nodes contain the same number of children but, each node must have  $m/2$  number of nodes.

A B tree of order 4 is shown in the following image.



While performing some operations on B Tree, any property of B Tree may violate such as number of minimum children a node can have. To maintain the properties of B Tree, the tree may split or join.

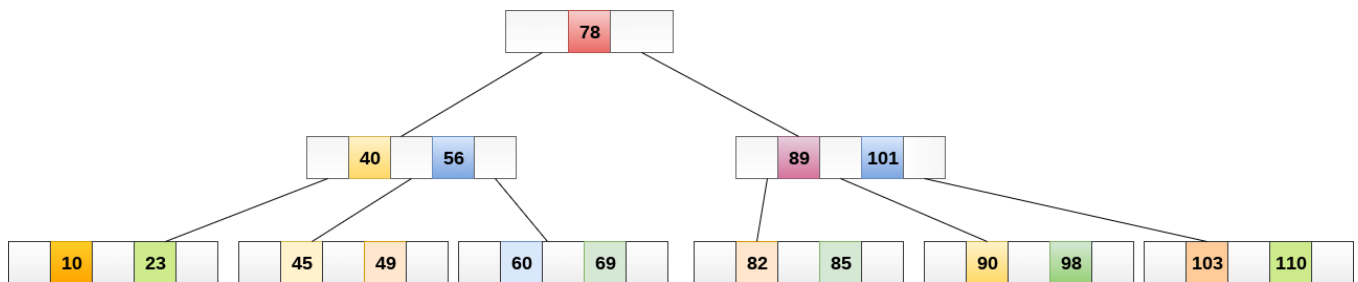
## Operations

↑ SCROLL TO TOP

Searching in B Trees is similar to that in Binary search tree. For example, if we search for an item 49 in the following B Tree. The process will something like following :

1. Compare item 49 with root node 78. since  $49 < 78$  hence, move to its left sub-tree.
2. Since,  $40 < 49 < 56$ , traverse right sub-tree of 40.
3.  $49 > 45$ , move to right. Compare 49.
4. match found, return.

Searching in a B tree depends upon the height of the tree. The search algorithm takes  $O(\log n)$  time to search any element in a B tree.



## Inserting

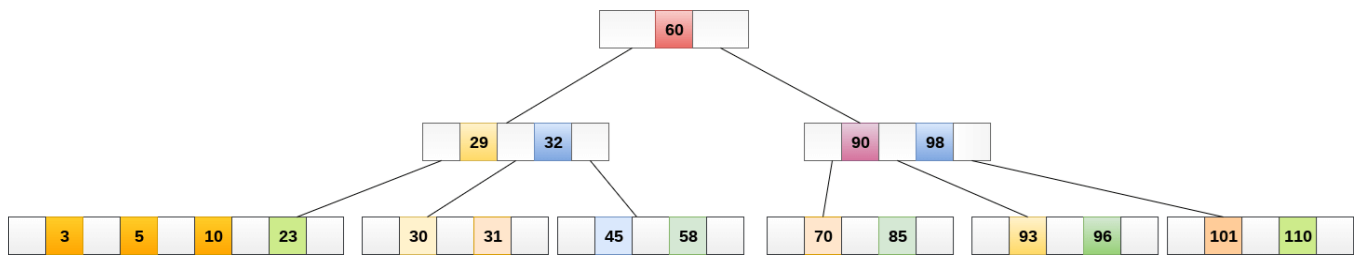
Insertions are done at the leaf node level. The following algorithm needs to be followed in order to insert an item into B Tree.

1. Traverse the B Tree in order to find the appropriate leaf node at which the node can be inserted.
2. If the leaf node contain less than  $m-1$  keys then insert the element in the increasing order.
3. Else, if the leaf node contains  $m-1$  keys, then follow the following steps.
  - Insert the new element in the increasing order of elements.
  - Split the node into the two nodes at the median.
  - Push the median element upto its parent node.
  - If the parent node also contain  $m-1$  number of keys, then split it too by following

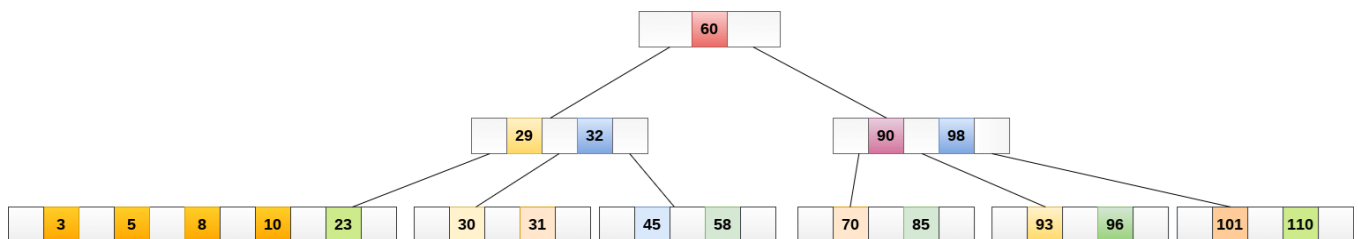
↑ SCROLL TO TOP

**Example:**

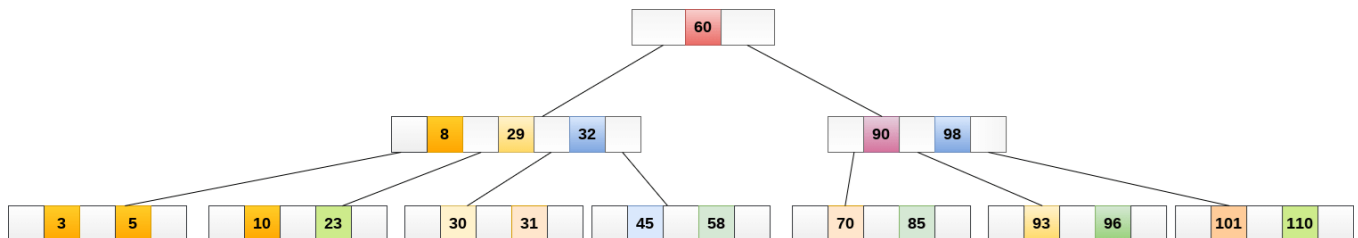
Insert the node 8 into the B Tree of order 5 shown in the following image.



8 will be inserted to the right of 5, therefore insert 8.



The node, now contain 5 keys which is greater than  $(5 - 1 = 4)$  keys. Therefore split the node from the median i.e. 8 and push it up to its parent node shown as follows.

**Deletion**

Deletion is also performed at the leaf nodes. The node which is to be deleted can either be a leaf node or an internal node. Following algorithm needs to be followed in order to delete a node from a B tree.

1. Locate the leaf node.
2. If there are more than  $m/2$  keys in the leaf node then delete the desired key from the node.
3. If the leaf node doesn't contain  $m/2$  keys then complete the keys by taking the element from right or left sibling.

↑ SCROLL TO TOP

If the sibling contains more than  $m/2$  elements then push its largest element to the parent and move the intervening element down to the node where the

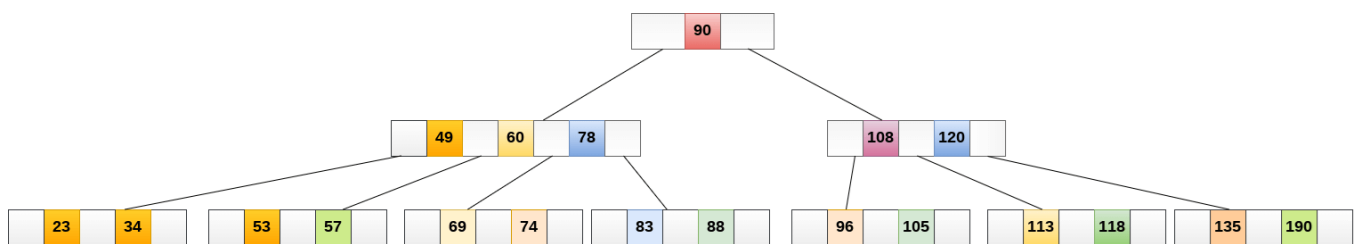
key is deleted.

- If the right sibling contains more than  $m/2$  elements then push its smallest element up to the parent and move intervening element down to the node where the key is deleted.
4. If neither of the sibling contain more than  $m/2$  elements then create a new leaf node by joining two leaf nodes and the intervening element of the parent node.
  5. If parent is left with less than  $m/2$  nodes then, apply the above process on the parent too.

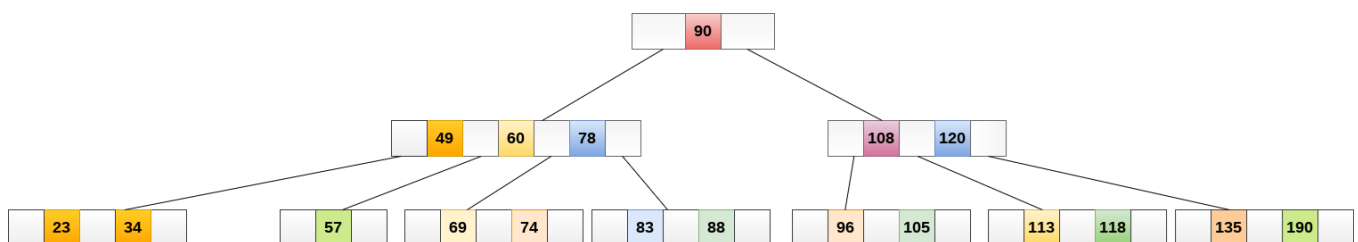
If the the node which is to be deleted is an internal node, then replace the node with its in-order successor or predecessor. Since, successor or predecessor will always be on the leaf node hence, the process will be similar as the node is being deleted from the leaf node.

### Example 1

Delete the node 53 from the B Tree of order 5 shown in the following figure.

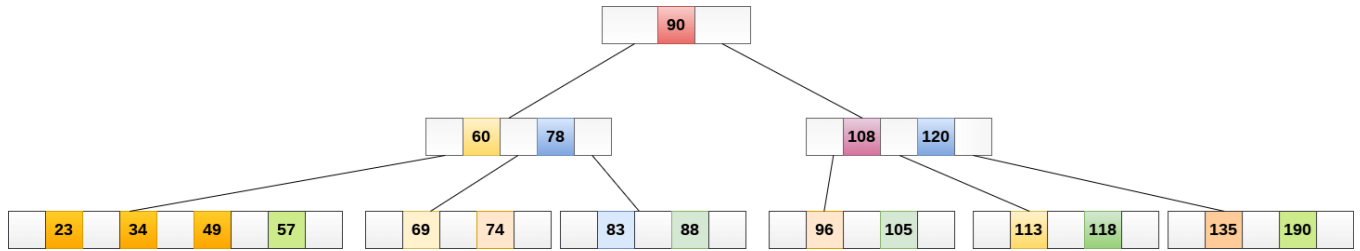


53 is present in the right child of element 49. Delete it.



Now, 57 is the only element which is left in the node, the minimum number of elements that must be present in a B tree of order 5, is 2. it is less than that, the elements in its left and right sub-tree are also not sufficient therefore, merge it with the left sibling and intervening element of parent i.e. 49.

The final B tree is shown as follows.



## Application of B tree

B tree is used to index the data and provides fast access to the actual data stored on the disks since, the access to value stored in a large database that is stored on a disk is a very time consuming process.

Searching an un-indexed and unsorted database containing  $n$  key values needs  $O(n)$  running time in worst case. However, if we use B Tree to index this database, it will be searched in  $O(\log n)$  time in worst case.

[< Prev](#)
[Next >](#)

[For Videos Join Our Youtube Channel: Join Now](#)

## Feedback

- Send your Feedback to [feedback@javatpoint.com](mailto:feedback@javatpoint.com)

## Help Others, Please Share



## Learn Latest Tutorials


[↑ SCROLL TO TOP](#)


[SPSS tutorial](#)  
SPSS

[Swagger tutorial](#)


[T-SQL tutorial](#)  
Transact-SQL

Swagger


Tumblr tutorial  
Tumblr


React tutorial  
ReactJS


Regex tutorial  
Regex

Reinforcement learning tutorial  
Reinforcement Learning

R Programming tutorial  
R Programming


RxJS tutorial  
RxJS

React Native tutorial  
React Native


Python Design Patterns  
Python Design Patterns


Python Pillow tutorial  
Python Pillow


Python Turtle tutorial  
Python Turtle


Keras tutorial  
Keras


Preparation

Aptitude  
Aptitude


Logical Reasoning  
Reasoning


Verbal Ability  
Verbal Ability


Interview Questions  
Interview Questions


Company Interview Questions  
Company Questions

Trending Technologies

Artificial Intelligence  
Tutorial

AWS Tutorial  
AWS

Selenium tutorial  
Selenium

Cloud Computing tutorial  
Cloud Computing

↑ SCROLL TO TOP

Artificial Intelligence Hadoop tutorial Hadoop	ReactJS Tutorial ReactJS	Data Science Tutorial Data Science	Angular 7 Tutorial Angular 7
Blockchain Tutorial Blockchain	Git Tutorial Git	Machine Learning Tutorial Machine Learning	DevOps Tutorial DevOps

## B.Tech / MCA

DBMS tutorial DBMS	Data Structures tutorial Data Structures	DAA tutorial DAA	Operating System tutorial Operating System
Computer Network tutorial Computer Network	Compiler Design tutorial Compiler Design	Computer Organization and Architecture Computer Organization	Discrete Mathematics Tutorial Discrete Mathematics
Ethical Hacking Tutorial Ethical Hacking	Computer Graphics Tutorial Computer Graphics	Software Engineering Tutorial Software Engineering	html tutorial Web Technology
Cyber Security tutorial Cyber Security	Automata Tutorial Automata	C Language tutorial C Programming	C++ tutorial C++
Java tutorial Java	.Net Framework tutorial .Net	Python tutorial Python	List of Programs Programs

[↑ SCROLL TO TOP](#)





Control  
Systems tutorial

Control System



Data Mining  
Tutorial

Data Mining



Data  
Warehouse  
Tutorial

Data Warehouse