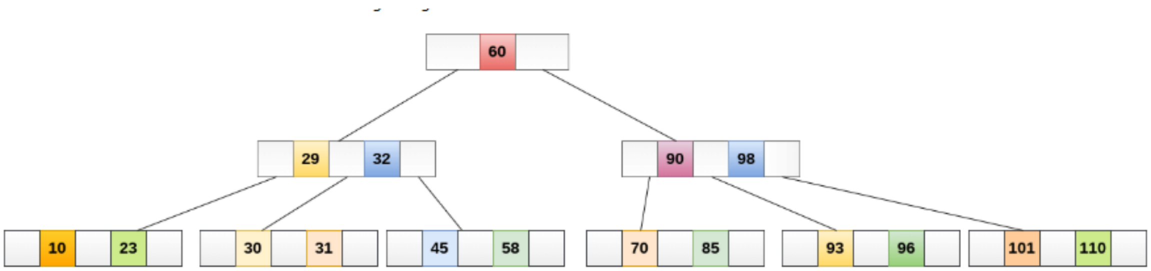
Report

B trees and search engine application

Names:

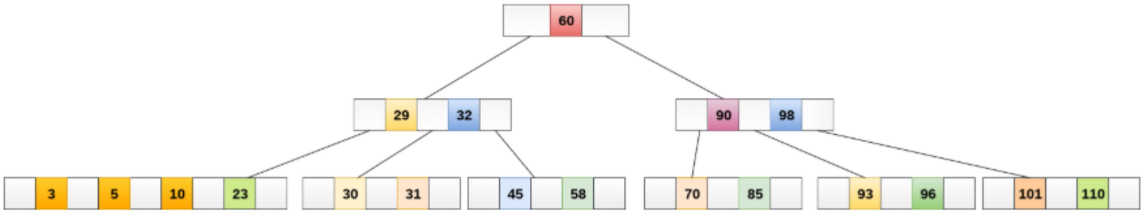
Omar Youssef	ID:45
Nada Fathy	ID:68

• B_Trees:

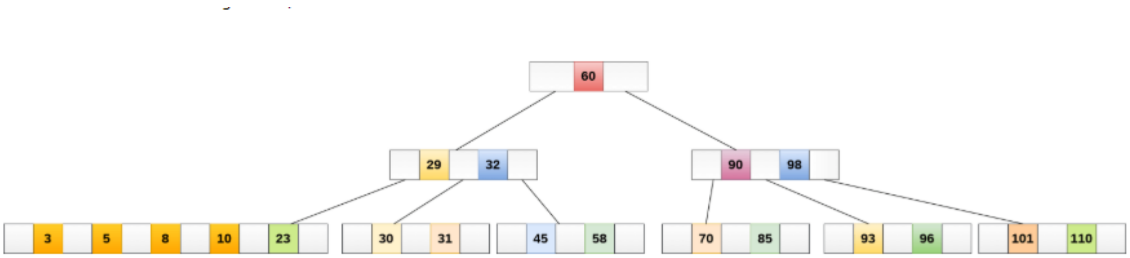


Inserting.

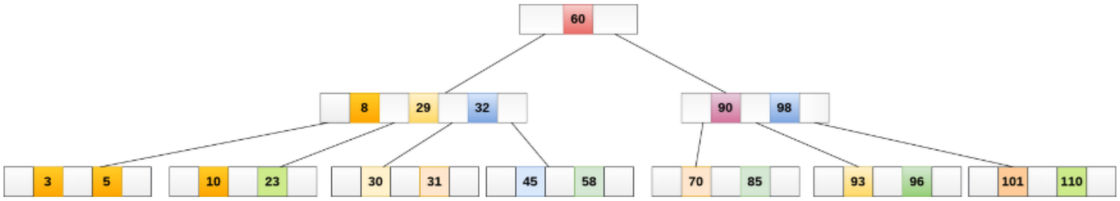
Insert the node 8 into the B Tree of order 5 shown in the following image.



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

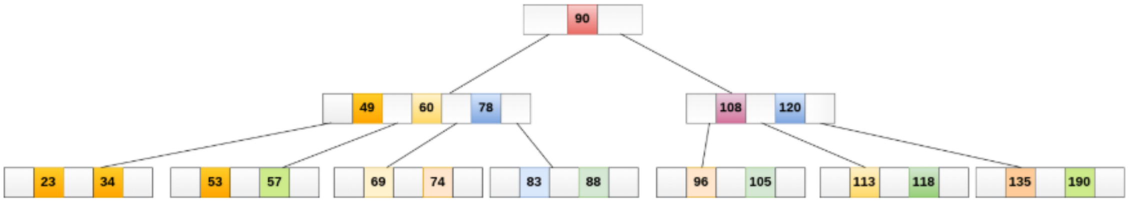


The node, now contain 5 keys which is greater than (5 -1 = 4) keys. Therefore split the node from the median i.e. 8 and push it up to its parent node shown as follows.



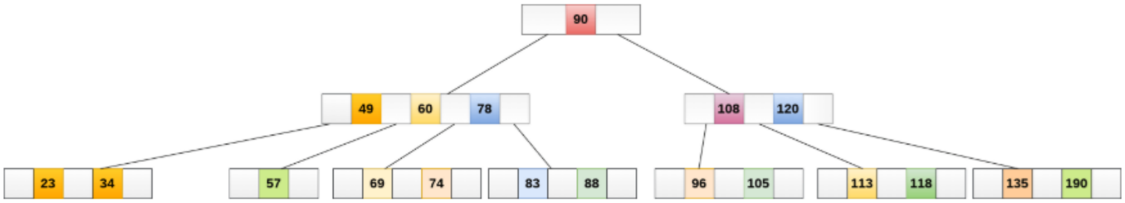
Deleting.

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



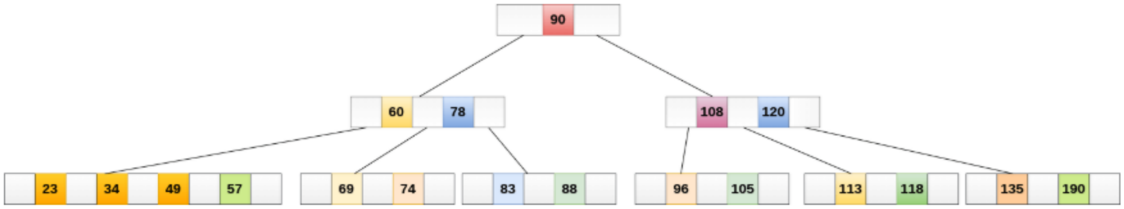
53 is present in the right child of element 49. Delete it.

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.



- ### *Definition of B-tree:*
   - **B-tree** is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in logarithmic time.

- ### *Basic functions of B Trees:*

1. Insert: adding new key in the b tree
   Inserting in b tree is always done in leaf nodes. When the node where we insert key is full, we split this node.
2. Split: splitting a certain node is done by dividing it into two nodes and shifting the median element to the parent node.
3. Search: searching for a key mapping in the tree.
4. Delete: deleting a specific key from the tree.

- ### *Time analysis for each function:*

Time of all previous functions is: $\log_d n$.

Where: d=m/2 (m is maximum node size).

- ## Simple search engine:

It is an application for b trees which is used to search in many xml documents using specific word(s).

- ## *Code design:*

- In search engine we used a single b_tree for indexing all words of the whole XML file (including all subdocuments).
- The words of the documents are used as keys for the tree.
- For each word(key), there is a list of Isearch result interfaces.
- This list represents the value for each key in the b_tree.
- Each element in this list contains the id of the document where the word(key) exists, and the frequency of this word in the document (number of occurrences).

- *Functions of search engine:*

- Index webpage: given XML file, we map **each word** in each document in this file using b tree.

- Index directory: given a directory for some XML files. For each file in this directory we use (Index webpage) function to map words of this file.
- Delete webpage: deleting all id’s of this page (XML file) from the b tree.
- SearchByWordWithRanking: searching in the b tree for a given **word** and return list of search result interfaces containing the id’s and ranking for this word (the value according to the key in the b tree).
- SearchByMultipleWordWithRanking: searching in the b tree for a given **sentence** and return list of search result interfaces containing the id’s and ranking for this sentence (the value according to the key in the b tree).

Time analysis:

- Index Webpage: time approximately equals time of search in the tree for each word in XML file.
- Index directory: it takes same time as ***index webpage*** for each XML file in the directory.
- Delete webpage: for each word in XML file, it takes same time as ***delete*** function in the b tree.
- SearchByWordWithRanking: takes logarithmic time same as ***search*** function in b tree.