COP 5536 Spring 2021 Programming Project Report

B+ Tree

**UFID:** 4360-0170

**Name:** Riyaz Basha Shaik

**Email:** [riyaz.shaik@ufl.edu](mailto:riyaz.shaik@ufl.edu)

**Introduction:**

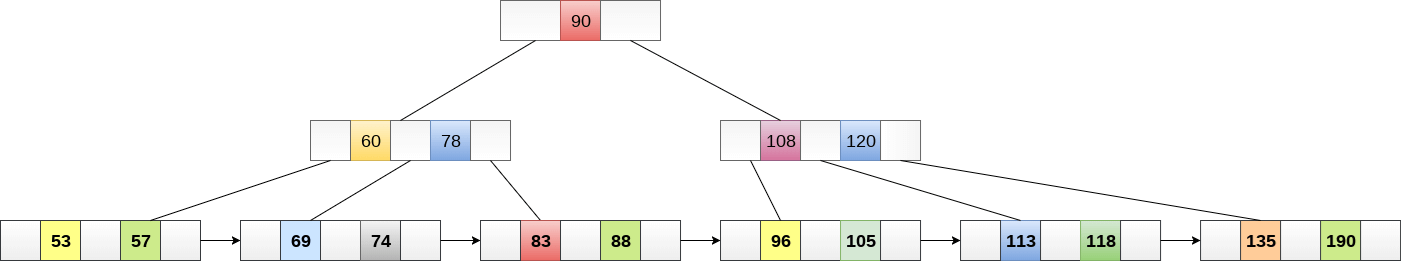
A B+ tree is an N-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children. A B+ tree can be viewed as a B-tree in which each node contains only keys (not key–value pairs), and to which an additional level is added at the bottom with linked leaves.

The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context — in particular, filesystems. This is primarily because unlike binary search trees, B+ trees have very high fanout (number of pointers to child nodes in a node, typically on the order of 100 or more), which reduces the number of I/O operations required to find an element in the tree.

Due to the fact that, size of main memory is always limited, the internal nodes (keys to access records) of the B+ tree are stored in the main memory whereas, leaf nodes are stored in the secondary memory.

The internal nodes of B+ tree are often called index nodes.

A B+ tree of order 3 is shown in the following figure:



**File Structure:**

**bplustree.java** is the source file which contains all the classes and methods needed to initialize a B+ Tree, perform insert, delete and search operations. This class contains main method which is the point of entry of the project. It takes the input file name from the command line argument, search for the file, opens it and reads it line by line, performing 5 kind of operations defined by the input file (initialize, insert, search by key, search between keys, delete) and writes the output of the searches to a new file, named “output\_file.txt”.

The following are the classes created for this project:

* **bplustree** – This is class where B+ Tree is implemented. It supports initialize, insert, search by key, search between keys, delete operations provided through the input file.
* **Data** - Nested class to hold B+ tree node key value pair.
* **DataComparator** - Nested class to sort key value pairs in the increasing order of keys.
* **InternalNode** - Nested class to represent Internal Node of a B+ Tree. Contains Internal Node as parent and left, right sibling Internal nodes forming a doubly liked list of Internal nodes, list of keys whose values are present in the leaf nodes of its children, list of child pointers
* **LeafNode** - Nested class to represent Leaf Node of a B+ Tree. Contains Internal Node as parent and left, right sibling leaf nodes forming a doubly liked list of leaf nodes and list of key value pairs stored by the leaf node
* **OutputRecordsFormatter** - Nested class to format the output file records.

Please follow the below steps to run the project:

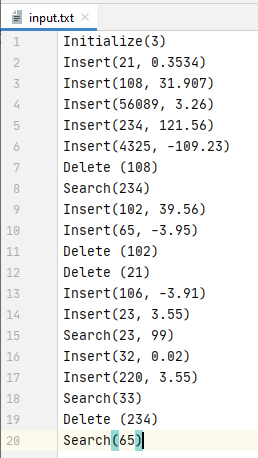
* **javac bplustree.java**
* **java bplustree <input\_file\_name>** 
  + **example: java bplustree input.txt**

To run using makefile. Please follow the below steps:

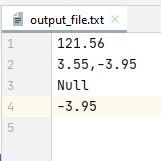
* Run **‘make bplustree.class’ or** just simply Run **‘make’**
* Run **‘make bplustree’ (It assumes default input file name is input.txt)**

or Run ‘**java bplustree <input\_file\_name>’ example : java bplustree input.txt**

The following is the sample input given to the project:



The following is the output generated for the above input:



**Program Structure:**

Below are the classes and methods created and their structure as part of this project:

