Akshat Bajpai UFID:69210478

University of Florida

Advanced Data Structures Programming Project

Report

B+ TREE

A Java-based Approach

B+Tree

- - - - X

B+Tree is an extension of the B-Tree Search which itself is an extension of Binary Search Tree. Rather than having a single element in the node, B+Tree has multiple keys in a single level. Also, unlike other trees B Tree and B+Tree follow the bottom-up approach of progression. There is an order associated with the B+Tree which is the maximum number of children a node can hold. Beyond which the node splits through its middle element. Just like other Data Structures, we can initialize,insert,delete and search the tree. The Assignment follows all these approaches.

The Programming Language used to implement the data structure is Java. There is a module for B+Tree already available for implementing it, but this assignment makes use of primitive data types.

File Structure of the Project

The folder comprises of following types of files:

- “.java” files that carry the codes
    - Generate.java : The main class that carries the main function. Handles file handling using buffers.
    - Tree.java : Structure and properties for B+Tree
    - Leaf.java :Handles the properties for the leaf nodes
    - Edge.java : Looks after balancing,overflow and split
    - Branch.java: Handles the internal nodes
- “.class” files that you get after compiling the .java code
- MakeFile that compiles all the code altogether and produces the corresponding .class file
- Input.txt: Where we type our input

B+ TREE

- Output.txt: Is generated automatically storing the corresponding outputs for the input file

Various Detailed Functionalities of the B+Tree:

- **INSERT:**
    - There are two types of values in the assignment:
        - Values that carry references to children nodes: This links the branches joining key nodes that are internal to the lower leaf nodes and the root nodes. This is an alternative to pointers in C++. These are basically node type and therefore can be used for referencing.
        - Values that are corresponding to the key in a node: These are the values that user inputs corresponding to their key values. These are mostly floats/integers/strings. Sorting does not apply on these.
    - There are two type of key inserts:
        - Keys at leaf: that is handled by the Leaf and edge classes.
        - Keys at internal that has their duplicates in the leaf nodes as well. This is managed by Branch class.
    - The insert is done in a sorted manner. Therefore the B+tree is already a bst while it is added. But we do check for NULL and overflow situations.
    - Returns nothing.

B+ TREE

- **REMOVE:**
  - This functionally makes use of the following routes:
    - Search(the element that you want to delete)
    - Traverse(upto the element)
    - Remove the element(element[previous]->element[next])
    - Check for Underflow, Null or even Overflow situation
  - Returns nothing.

- **SEARCH(single key):**
  - This function searches mostly in the edge doubly linked list.
  - Returns the value for key.

- **SEARCH(range1,range2):**
  - This function traverses through the range1 to range2 of the B+Tree.
  - As it traverses, it keeps storing the intermediate <key,values> in a String type object and returns them.

Node structure:

```
class Leaf {     //Leaf node contains key: value
   double key;
   String val;

  public Leaf(double key, String val){     //constructor to initialize
the Leaf node
      this.key = key;
      this.val = val;
  }
}
```

B+ TREE

```
class Node {         //Node properties to determine rotation:-
   int order;           //Maximum number of child that it can have
   boolean Edge = false;    //is the node leaf?
   boolean overFlow = false;    //Is the node full and thus, does it
require Splitting?

}
```

Compiling and Running the Project

- - - - X

To run it in "thunder.cise.ufl.edu" server.

- Login to the above mentioned server through either of the telnet clients. (I used Putty)
- Run the Makefile.
- Then just run:
    - $ java generate input
- Then in the current working directory, an output file will automatically be created. Open that to see the output.

For users outside the server.

- Open terminal/cmd in the assignment folder path
- To Compile:
    - $javac *.java
- To Run:
    - $java generate input

B+ TREE

- Then in the current working directory, an output file will automatically be created. Open that to see the output.