COP 5536 Advance Data Structures

Fall 2017

Programming Project Report

B+ Tree implementation

Jeshalraj Thakaria

UFID:

Jthakaria@ufl.edu

Table of Contents

1. Project Statement

- Develop and test a class B+Tree to store pairs of the form (key, value). For this project you are to
  implement a memory resident B+tree (i.e., the entire tree resides in main memory).
- Implementation must be able to store multiple pairs that have the same key (i.e., duplicates). The supported operations are:
  1. Initialize(m): create a new order m B+Tree
  2. Insert (key, value)
  3. Search (key): returns all values associated with the key
  4. Search (key1, key2): returns (all key value pairs) such that key1 <= key <= key2.

2. Languages and operating system used

Used Language : Java

Operating System: Windows (Eclipse IDE),

Linux(thunder.cise.ufl.edu)

3. Definition of functions and classes

3.1 Class KeyValuePair:

Data Members:

Name	Type	Description
value	String	Stores string value at any given Node
key	Float	Used to store the float key at any given Node

Class Methods:

1	**KeyValuePair (Float key, String value)**
	Constructor to set the key and value pairs in each object of the class

3.2 Class BtreeNode

Data Members:

Name	Type	Description
maxdegree	Int	The order of the B_tree to be used
Size	Int	Stores the size of the B_tree
parent	Node*	Parent of the Node
Previous	Node*	Previous pointer for the doubly linked list in Node
Next	Node*	Next pointer for the doubly linked list in Node
Al	Arraylist< KeyValue Pair>	Arraylist to store the key and value pairs at a Node
Child	Arraylist< BtreeNod e>	Arraylist to store the pointer of the children of a Node
KeyVal	Arraylist< Float>	Arraylist to store the index key at index nodes

Class methods:

1	**BtreeNode(int m,BtreeNode Node)**
	Constructor to Initialize the B+ tree
2	**Void add (float key, String value, BtreeNode Node)**
	Insert a key value pair at the leaf Node passed in the function. Check if the leaf is empty, add the key and value pair to it. Call the divide function if the leaf gets full i.e. number of index or key value pair in the node is equal to the order, else add the pair and return from the function.
	Void divide (BtreeNode Node)

3	Divide the the splitting of nodes according to the conditions: 1) If root is full and it is a leaf node 2) Root is divided and root is not a leaf 3) Internal index node is full 4) Leaf node is full and it is not the root Based on this the splitNode function which implements the splitting of the node.
4	**Void splitNode (BtreeNode Node,int flag)**
	The int flag is the one passed from the divide function according to the conditions mentioned in the divide function. Splitting of the key_value pairs and child from the node which got full into two. Each splitting function is implemented according to the conditions of splitting involved for the four cases in divide.
5	**BtreeNode findNode (BtreeNode Node,float Key)**
	This function traverses the tree and finds the leaf node appropriate for the key passed into the function.
6	**Void findRange(BtreeNode Node, float key1, float key2)**
	This function uses findNode to find the leaf node associated with the key. This function also is used both for single key search and range based search. For single value search, we pass the same key to both arguments. This function finds the leaf nodes for both the keys and then traverses from one leaf node to other printing all the key value pairs in between the range.

3.3 Class Btree

Data Members:

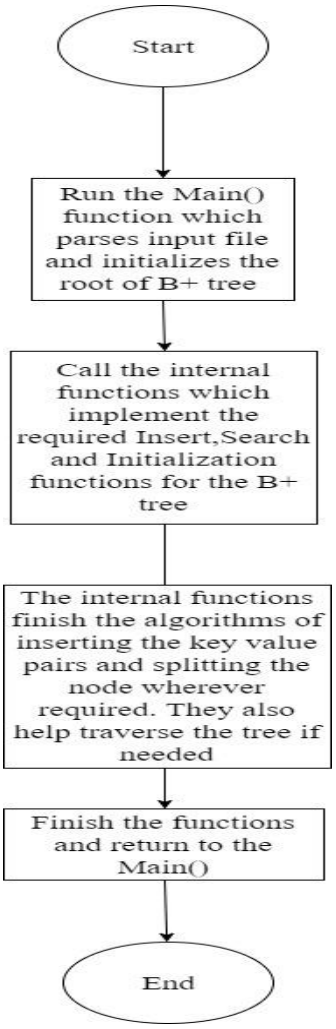
Name	Type	Description
br	BufferedReader 6	Used to read input file
m	int	Stores order of the B+tree
line	String	Stores the lines from the input file
root	Node*	Initial root of the B+tree initialized

Class methods:

1	**void main()**
	Parses the input file and calls the function required for processing the request. Transfer all the output to the output.txt file.
2	**BtreeNode Initialize(int m)**
	Initialize a new B+ tree using the BtreeNode class constructor.
3	**Void Insert(BtreeNode root, String key1, String value)**
	Call the add function on the root and pass key1 and value as arguments after converting key1 to float.
4	**Void Search(BtreeNode root, String key1,String key2)**
	Calls the findRange function in using key1 and key2 as arguments after converting them to float. This implements the basic search functionality in the code.

4. Program Structure:

Btree.java is the main file that contains the main function which initializes and runs the program, parsing the input file and calling the required functions to operate on the B+ tree. Once the appropriate functions are called, they invoke the functions from BtreeNode.java file which has algorithm implementation for insert, split, search and constructor for initial tree.



Instructions to run the code:

1. To compile all the java files, run the “Make” command on the terminal in the folder with the files.
2. “Java Btree filename” to run the program with the input file.

Materials Used:

1. Class Lectures and ppts
2. Java Documentation