COP 5536 Advance Data Structures Spring 2017 Programming Project Report

B+Tree

Rohan Naik

UFID: 02167483

rhnnik@ufl.edu

Table of Contents

1. Goals of the project

- Implement B+ Tree with the following functionalities:

  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. Programming Environment

Language : Java

Operating System: Linux Ubuntu, Windows

Data Structure Used: B+ Tree

3. Function Prototypes

3.1 Class KeyValuePair:

Data Variables		
Name	**Type**	**Description**
Key	Float	Stores key of a record
Value	String	Stores data/ value of a record

3.2 Class BtreeNode

Data Variables		
Name	**Type**	**Description**
Maxdegree	Int	Degree of Btree
parent	Node	Parent of this BtreeNode
al	ArrayList<>	Arraylist of key-value pairs
child	ArrayList<>	Arraylist of child nodes

3.2.1 Public functions:

1	**Add**
	Insert key value pair into the tree entering from leaf node If the node is a leafnode and it is not full, it adds the keyvalue pair straightaway If the node is full, it calls the split method which is explained below.
2	**Split**
	When a node is full, it needs to be split according to the algorithm explained in section 4. This is done by using one of the below cutNode methods depending on the case. It stores the child nodes after every split in temp1 and temp2 The parent of temp node is set to the node formed by the new node formed by midpoint of original node. If the splitted node is not a leafnode, the cutnode2 and cutnode3 also handle the next and previous pointers are set after every split such that the left node's next points to right node.
3	**cutNode**
	If node has no parent and no child (root node is a leafnode), use this split method
4	**cutNode2**
	If node has no parent but has at least one child (is not a leafnode), use this split method
5	**cutNode3**
	If node has a parent and at least one child(is not a leaf), use this split method
6	**cutNode4**
	If node has a parent but no child (is a leafnode)
7	**findNode**
	Traverses through the tree to the LeafNode of interest
8	**findRange**
	Returns the key and values of interest For Search(key) it returns just the value and for Search(key1, key2) it returns list of key and value pairs for all keys such that k1<= key <=key2

3.3 Class treesearch (contains main)

Data Variables		
Name	**Type**	**Description**
m	integer	Degree of tree
root	BtreeNode	Object for root node of tree
printflag	Flag	To check if the query is a Search or SearchRange normal search

3.3.1 Public functions:

1	**Initialize(degree)**
	Initializes the B+tree with degree m
2	**Insert (key, value)**
	Inserts key value pairs in the tree after checking various conditions
3	**Search(key)**
	Search for a key and returns value if key is found or returns null
4	**Search(key1, key2)**
	Takes two keys as arguments and returns all key value pairs between key1 and key2

4. Program Structure

The core of the B+tree data structure is the insertion algorithm. Once the Btree of degree m is initialized, key value pairs are inserted one by one until the root node is full.

At each insertion, the findNode method traverses to the actual position of insertion in the LeafNode. After checking for the conditions if the Node is a leafNode and is full or not, the keyvaluepair is inserted at the right position with the help of split function.

The following conditions were checked exploited for correct implementation of insert:

Leaf Node	Index Node	Split Logic
Not Full	Not Full	- Search the intended leaf node's position by search helper - Insert the record and sort the elements
Full	Not Full	- Call the split function - Split the leaf node - Insert the middle key in the parent node and sort the elements - Check parent node size (not full) - Left node will contain elements with key <= middle key - Right node will contain elements with key > middle key
Full	Full	- Call the Split function -Check whether the scenario qualifies for cutnode1, cutnode2 or cutnode3 - Split the leaf node accordingly - Insert the middle key in the parent node and sort the elements - Left node will contain elements with key <= middle key - Right node will contain elements with key > middle key - Check parent node size (full) - Split the index page - Left node will contain elements with key <= middle key - Right node will contain elements with key > middle key - Middle key goes to its own parent and 1 level higher - Recursively check until the parent node is not full

Readme:

In order to run the program, enter the following instructions:

1. **make (to compile the java files)**
2. **java treesearch filename**

References:

i. **https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html**
ii. **http://www.geeksforgeeks.org/b-tree-set-1-insert-2/**