Advanced Data Structures Project Report

Srajan Paliwal, UFID : 51739915 , Email: srajanpaliwal@ufl.edu

***Abstract*—In this report, I will discuss the implementation of four basic operations on memory resident B Plus Tree. The four operations are - Initialize(m): creates a new B Plus Tree of order 'm', Insert (key, value): insert given key value pair in the tree, Search (key): returns all values associated with the given key, and Search (startKey, endKey): returns all key value pairs between startKey and endKey.**

I. INTRODUCTION

This implementation of B+tree is in Java. In this report first, I will a describe the class structure for the program and give an overview of each class. Then, The following section will cover detailed information about the implementation of that class and its functions.

II. OVERVIEW OF CLASS STRUCTURE

The program contains to two packages namely - bTree[*com.ads.bTree*] and doublyLinkedList[*com.ads.doublyLinkedList*]. The doublylinkedlist package implements a Doubly Linked List used to connect B+Tree's leaf nodes for efficient range search. The bTree package implements B Plus Tree. Apart from these, There is main java class named *treesearch.java* that takes input file as command line argument and outputs a text file with the result of operations.

Finally,The complete class structure is given in figure[insert figure]. And, The source for the program is in *./src/* folder.

III. B PLUS TREE PACKAGE

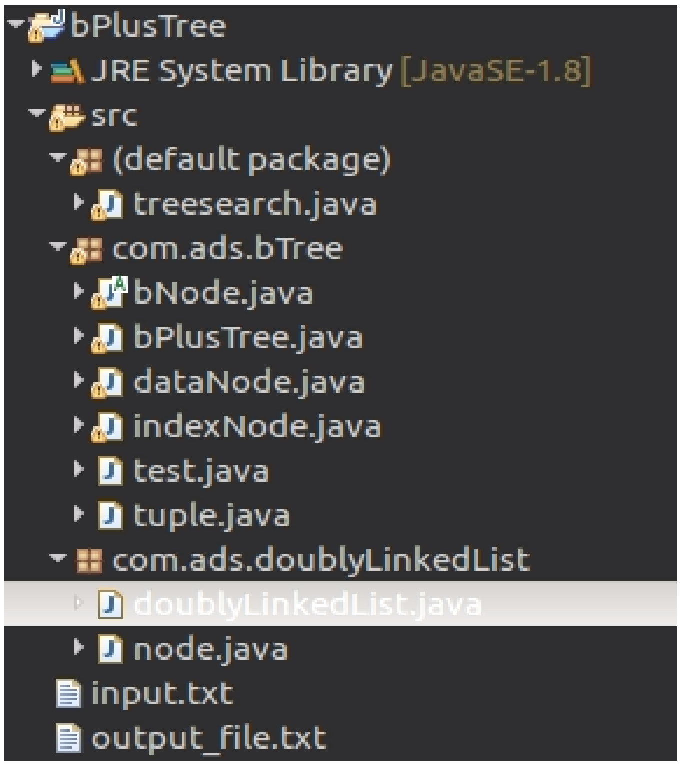
As shown in figure 1, The package has five classes-B Plus Tree [*bPlusTree.java*], Index Node [*indexNode.java*], Leaf node [*dataNode.java*], B Node [*bNode.java*] an abstract class inherited by index and leaf nodes, and tuple [*tuple.java*] a simple key-value pair class to store data at the leaf node.

A. B Plus Tree Class

This class implement above mentioned operations on B Plus Tree. The main functions and properties of this class are described below. The class ouline is shown in figure 2.

1) Properties: This class has three properties namely - **order**: contain the maximim degree of leaf nodes, **root**: points to the root of B Plus Tree, **leafList**: Doubly linked list to connect all leaf nodes.

2) Constructor: The constructor of B Plus Tree Class initializes a tree of given order. It sets order of tree and creates root and leaf List. This is called when treesearch class creates a new B plus Tree.



```
bPlusTree
  JRE System Library [JavaSE-1.8]
  src
    (default package)
      treesearch.java
    com.ads.bTree
      bNode.java
      bPlusTree.java
      dataNode.java
      indexNode.java
      test.java
      tuple.java
    com.ads.doublyLinkedList
      doublyLinkedList.java
      node.java
  input.txt
  output_file.txt
```

Fig. 1. Complete class structure

3) insert: This functions take **item**: the key value pair to be inserted.

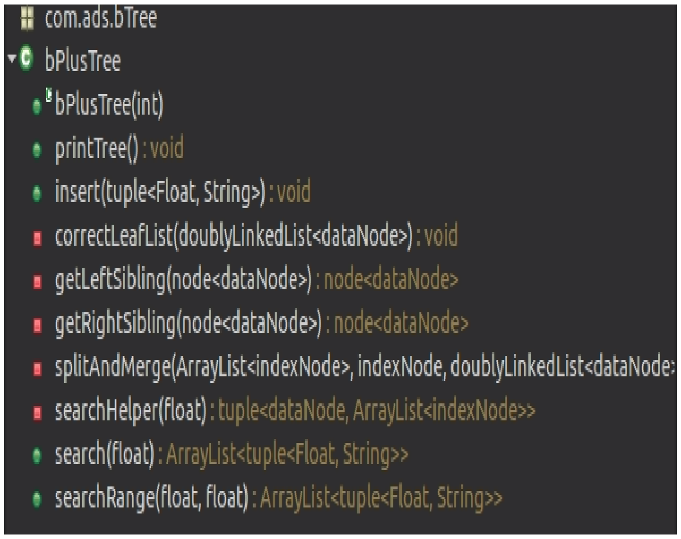
First, this function calls *searchHepler* function to perform a search to determine what leaf node the new record should go into. If the node is not full i.e node size is less than order -1. The tuple is added to that node. Otherwise, its split the leaf node. This function only splitting at the leaf level.

Second, The tree formed by middle key as root and second part of overflow node as the left child is passed to the *splitAndMerge* [add reference] function. The *splitAndMerge* function recursively splits parent nodes until an empty parent is found or it reaches root.

4) splitAndMerge: This function takes **parentStack**: the path to node, **mergeTree**: the tree to be inserted, **treeLeafList**: leaf list of merge tree as value.

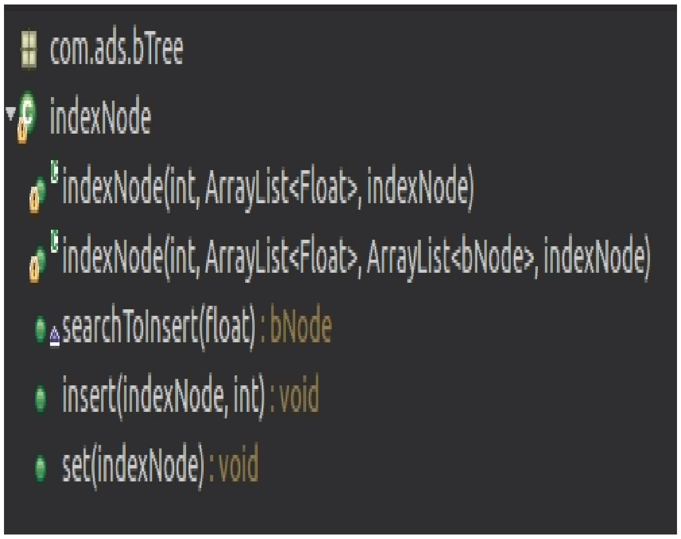
First, If the parent node is empty, it inserts the root of merge tree in the parent and corrects the leaf list for the B Plus Tree by calling *correctLeafList* function.

Second, if the parent is full, it splits it too. and recursively calls *splitAndMerge* function until a parent is found that is empty. If the function reaches the root and it splits a new root is created which has one key and two pointers. This leads to



```
com.ads.bTree
bPlusTree
  bPlusTree(int)
  printTree() : void
  insert(tuple<Float, String>) : void
  correctLeafList(doublyLinkedList<dataNode>) : void
  getLeftSibling(node<dataNode>) : node<dataNode>
  getRightSibling(node<dataNode>) : node<dataNode>
  splitAndMerge(ArrayList<indexNode>, indexNode, doublyLinkedList<dataNode:
  searchHelper(float) : tuple<dataNode, ArrayList<indexNode>>
  search(float) : ArrayList<tuple<Float, String>>
  searchRange(float, float) : ArrayList<tuple<Float, String>>
```

Fig. 2. Outline for B Plus Tree.



```
com.ads.bTree
indexNode
  indexNode(int, ArrayList<Float>, indexNode)
  indexNode(int, ArrayList<Float>, ArrayList<bNode>, indexNode)
  searchToInsert(float) : bNode
  insert(indexNode, int) : void
  set(indexNode) : void
```

Fig. 3. Outline for Index Node Class.

the growth of B Plus Tree. B Plus Trees grows at the root and not at the leaves.

5) searchRange: This function take **startKey** and **endKey** as inputs. Then, it finds the node containing startKey or node containing some values greater startKey using *searchHepler*. Then, it iterates over leaf node using the doubly linked list until keys are less than or equal to end key.

6) search: This fuction take a single **key** as input and calls *searchRange* with startKey and endKey eaqaul to input key.

7) searchHealper: This function searches the tree for given key until it reaches a leaf node. On the way down the B Plus Tree, it stores the path to leaf Node. Finally, it returns path to leaf and leaf node.

B. B Node class

This is an abstract class that defines the generic structure of index and leaf node. Properties of this class **order**: degree of b plus tree, **items**: list of values store, and **links**: list of children nodes.

This Class also implements a *size* function that returns the number of items stored in the node.

C. Index Node Class

This class implements internal node of the B Plus tree. These nodes only contain keys. The properties are inherited from bNode. Some important functions implemented in this class are. The class outline is shown in figure 3.

1) searchToInsert: This function take key as input and returns link to the child node. The child node is selected such that item before that is less than key and item after it is greater than and equal to key.

This function is called by the *searchHepler* to search the path to correct leaf node.

2) insert: This function takes a **mergerTree**, described above, and **index** as input values. This root Key of merge tree is added to the items list after given index. Its children are added to *links*.

It is called by *splitAndMerge* function to insert item to a empty internal node.

D. Leaf node Class

dataNode class implements leaf node of the B Plus tree. These nodes only contain tuples, key value pairs. The properties are inherited from bNode.The class outline is shown in figure 4. Some important functions implemented in this class are.

1) searchToRange: This function takes **startKey** and **endKey** as input and returns a list of all tuples with keys between given inputs. It also returns a Boolean value to indicate if the next node in leafList should be check or not

This function is called by the *searchRange* to traverse leaf nodes.

2) insert: This function takes a **tuple** as input values. This tuple is added to the leaf node such that item before it is less that *tuple.key* and item after it is greater then and equal to *tuple.key*.

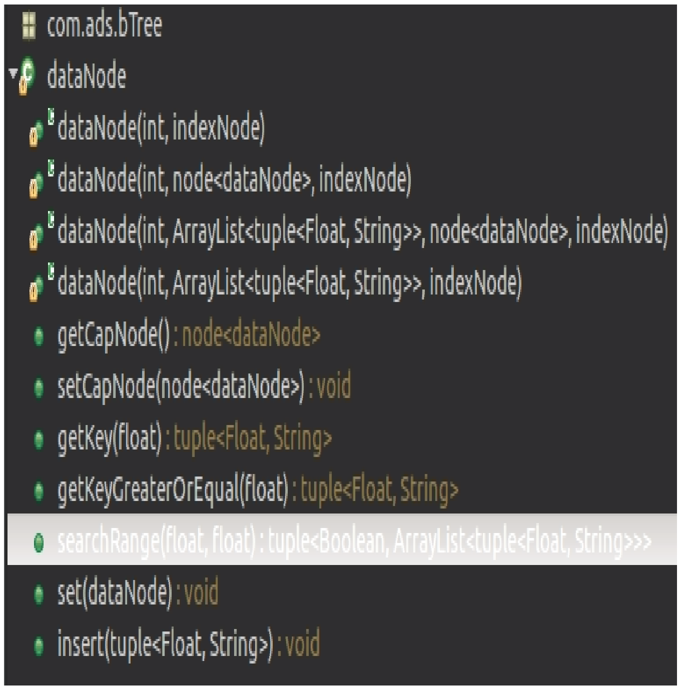
It is called by *insert* function to insert item to a empty leaf node.

E. Tuple Class

This a generic key-value pair class. The *¡valueType¿* in the class definition is a generic type. This class was implemented to hold data of any data to object type.

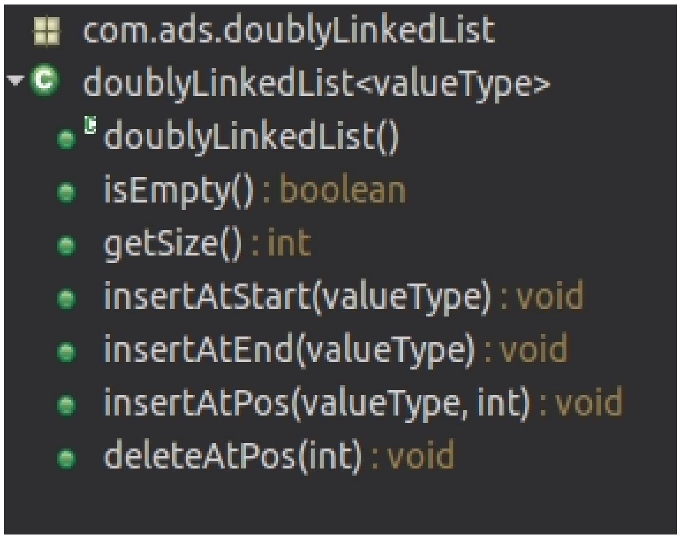
IV. Doubly Likened List

As shown in figure[insert here], The package has two classes- node, doublyLinkedList. These classes are also implemented to hold any datatype or object type. This is achieved



```
com.ads.bTree
dataNode
  dataNode(int, indexNode)
  dataNode(int, node<dataNode>, indexNode)
  dataNode(int, ArrayList<tuple<Float, String>>, node<dataNode>, indexNode)
  dataNode(int, ArrayList<tuple<Float, String>>, indexNode)
  getCapNode() : node<dataNode>
  setCapNode(node<dataNode>) : void
  getKey(float) : tuple<Float, String>
  getKeyGreaterOrEqual(float) : tuple<Float, String>
  searchRange(float, float) : tuple<Boolean, ArrayList<tuple<Float, String>>>
  set(dataNode) : void
  insert(tuple<Float, String>) : void
```

Fig. 4. Outline for Leaf Node Class.



```
com.ads.doublyLinkedList
doublyLinkedList<valueType>
  doublyLinkedList()
  isEmpty() : boolean
  getSize() : int
  insertAtStart(valueType) : void
  insertAtEnd(valueType) : void
  insertAtPos(valueType, int) : void
  deleteAtPos(int) : void
```

Fig. 5. Outline for Doubly Linked List Class.

by implementing java generic types. The class outline is shown in figure 5.

A. Doubly Linked List Class

1) insertAtStart: This function adds the given node at the head.

2) insertAtEnd: This function adds the given node at the tail.

3) insertAtPos: This function adds the given node at the given position.

4) deleteAtPos: This function deletes the given node at the given position.

V. Main Java Class

This is the file which contains the main function. This function takes input file name as command line input. The input file is read and parsed to read the order to initialize B Plus Tree. Then, The pareses operations like insert, search and range search and updates the tree. The output of search calls is stored in output file.

VI. Running the Program

The list of useful commands.

- make : Builds the project and creates *treesearch.class* file in the current directory.
- make clean : Removes the **.class* files and *./com/* folder.
- java treesearch input.txt : Runs the program and creates an output file.