ADVANCED DATA STRUCTURES

COP5536 FALL 2017

Instructor: Dr. Sartaj Sahni

Programming Project

(Implementation of Initialize, Insert and Search operations in B+ Tree to store Dictionary pairs)

Name: Aditya Vashist

UFID: 13571398

UF Email: aditya.vashist@ufl.edu

Table of Contents

Associated Files

- BPlusTree.java: - This is the main file which includes all the functions to implement the B+ tree. The functions are mainly: Initialize, Insert, Search, Range Search. Splitting the node whenever overflow happens.

- ParentNode.java: The ParentNode.java file is a blueprint for the parent node and have the shared characteristics fields and methods for leaf and treenode instances. like overflow conditions in the node and key arraylist, degree of the tree.

- Leaf.java: This file defines the blueprint for the leaf node of the given B+ Tree instance. It have fields specific to leaf nodes like arraylist of values associated with the given key arraylist.

- TreeNode.java: This file handles the insertion of keys in the index node and maintains them in sorted order.

- treesearch.java: This is the main class for the given program. It handles reading the input file initializing the instance of the B+ Tree with the given Degree and implement all the commands given in the input file. It also handles writing the output into the output_file.txt will all the results from the search queries.

- KeyNodePair.java: This class defines the blueprint for a key node pair instance which will store the value of key and node associated with it whenever the split happens.

- makefile: A makefile an easier way to handle and organize the compilation of code. It can handle “make” and “clean make” commands.

Function Prototypes and Program Structure

ParentNode.java

Fields

1. protected boolean isLeaf
   define whether the instance of ParentNode is leaf or TreeNode
2. protected ArrayList<Double> keyList
   Store the list of all the keys associated with that node. Since it is being used by both Leaf and TreeNode thus include in the parent class.
3. protected Integer degree
   Stores the degree of the given instance of B+ tree

Methods

1. public ParentNode(Integer degree)
   Constructor to create the Node instance with the given degree
2. public boolean isOverflow()
   Method to check the overflow of the node

TreeNode.java

Fields

1. protected ArrayList<ParentNode> childNodes
   List contains all pointers to the child nodes of the given TreeNode

Methods

1. public TreeNode(Double key, ParentNode child0, ParentNode child1, Integer degree)
   Constructor to create an instance of TreeNode with a single key value which will have child0 and child1 as the right and left children
2. public TreeNode(List<Double> newKeys, List<ParentNode> newChildren, Integer degree)
   Constructor to create an instance of TreeNode with list of keys and list of children nodes
3. public void insertInSortedOrder(KeyNodePair e, int index)
   Insert the keyNode pair in the given sorted order.

Leaf.java

Fields

1. protected ArrayList<String> values
   list of Values associated with the given leaf node.
2. protected Leaf leafRight
   It is a pointer to the right leaf node of the tree.
3. protected Leaf leafLeft
   It is a pointer to the left leaf node of the tree.

Methods

1. public Leaf(Double firstKey, String firstValue, Integer degree)
   Constructor to create an instance of leaf node when there is only one key and one value to populate.
2. public Leaf(List<Double> newKeys, List<String> newValues, Integer degree)
   Constructor to create an instance of new leaf node with the given list of keys and corresponding list of values.
3. public void insertInSortedOrder(Double key, String value)
   Method to insert new key value pair into the given leaf node in sorted order.

BPlusTree.java

Fields

1. public ParentNode root
   root of the Instance of B+ plus tree
2. public Integer maxSize
   maximum size of the tree or leaf node i.e. degree-1
3. public Integer minSize
   minimum size of the tree or leaf node i.e. ceil(degree/2) -1
4. private Integer degree
   degree of the given instance of the B+ tree

Methods

1. public BPlusTree(int degree)
   Constructor to initialize the BPlustree instance with the given degree. It all assign values to minSize and max Size of the tree and leaf nodes.
2. public String searchTheKey(Double key)
   return the string with the value corresponding to the given key.
3. public ArrayList<String> searchRange(Double key1, Double key2)
   search the given instance of B+ tree for the range of key values ranging from key1 to key2. Returns the list of key value pairs which fall in the given range.
4. private ParentNode fetchTreeNode(ParentNode parentNode, Double key)
   return the Parent Node with the given key as one of the keys in the range of the given nodes. Input parameters are tree node as root node and key to look for.
5. public void insert(Double key, String value)
   Insert the key value pair into the given instance of B+tree.
6. private KeyNodePair getTheChildEntry(ParentNode parentNode, KeyNodePair entry, KeyNodePair keyNodePair)
   This method is called in insert function will return null if no split is required or will return the keyNodePair where the split happens.
7. public KeyNodePair splitTheLeafNode(Leaf leaf)
   Split the given leaf node and return the keyNodePair Instance with key as the parent and corresponding node as a child
8. public KeyNodePair splitTheIndexNode(TreeNode treeNode)
   Split the given treeNode and retrn the KeyNodePair instance with key as the parent and corresponding node as a child

KeyNodePair.java

Fields

1. private ParentNode parentNode
   Node associated with the given key value
2. private Double key
   key value associated with the given node

Methods

1. public KeyNodePair(Double key, ParentNode parentNode)
   constructor to create a key node pair with key and parentNode as it key and value
2. public Double getKey()
   returns the key value of the given keyNodePair instance
3. public ParentNode getValue()
   Returns the Node associated with the given KeyNodePair instance

treesearch.java

Fields

1. private static BPlusTree tree
   B+ Tree instance to be initialized in initialise(int m) function

Methods

1. private static void initialise(int m)
   Initialize the BPlusTree instance with degree 'm' passed as an input.
2. private static void insert(String input)
   read the input string and then separate the key and value pairs before inserting it into the B+ Tree
3. private static void search(String input, StringBuffer output)
   Search the given string and store the output into a StringBuffer to be stored in the output file once all the input commands are processed.
4. private static void searchRange(String input, StringBuffer output)
   read the input string and read the two key values separated by ','. The call the searchRange function of the B+ tree class to fall the values, falling in between these keys. Then store it into the StringBuffer to be stored at the end.
5. public static void main(String args[])
   This is the entry point for the given application. args[0] is the input file to process all the commands in the given file and store the output back into output_file.txt.

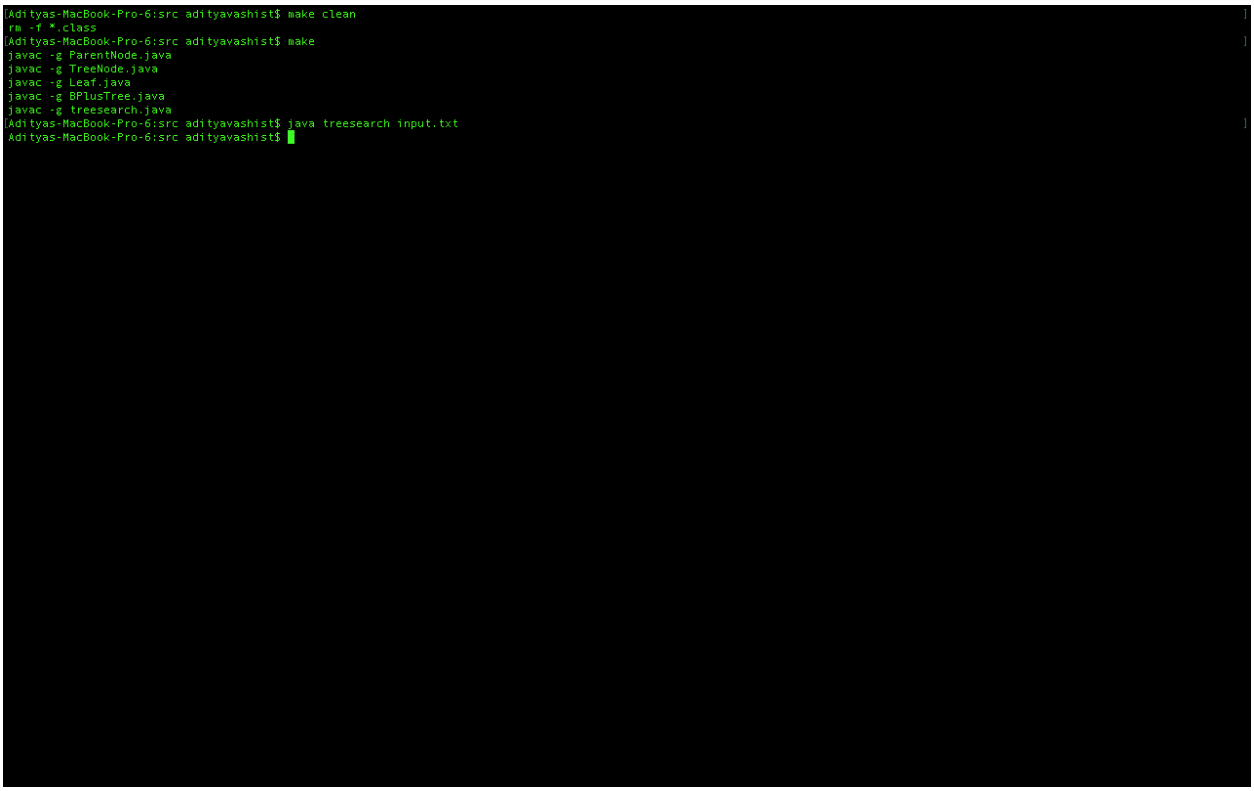
Running the Program

The project has been compiled and tested on the Mac Platform using the javac compiler jre version 1.8.0_77.

It has also been compiled and tested on Thunder.

Commands to execute the project:

1. make
2. java treesearch input_filename.txt



@thunder

```
thunder:17% make clean
rm -f *.class
thunder:18% make
javac -g ParentNode.java
javac -g TreeNode.java
javac -g Leaf.java
javac -g BPlusTree.java
javac -g treesearch.java
thunder:19% java treesearch input.txt
thunder:20% cat output_file
Value41
(-0.31,Value84), (0.89,Value42), (1.04,Value50), (15.52,Value73), (22.75,Value48), (26.72,Value49), (27.37,Value9)
Value113,Value149,Value184,Value212
(-28.83,Value99), (-28.74,Value100), (-20.28,Value125), (-13.78,Value86), (-12.82,Value199), (-8.95,Value222), (-4.66,Value47), (-3.84,Value207), (-0.31,Value84), (0.89,Value42), (1
.04,Value50), (15.52,Value73), (17.99,Value170), (22.75,Value48), (25.29,Value139), (26.72,Value49), (27.37,Value9), (34.58,Value186), (36.57,Value226), (37.58,Value168), (37.78,Val
ue71), (39.46,Value164), (42.02,Value23), (44.15,Value133), (46.7,Value103), (48.68,Value135), (54.56,Value60), (54.74,Value213), (56.49,Value132)
Value7219
(10.46,Value792), (10.51,Value1018), (10.63,Value3458), (10.64,Value5376), (10.94,Value5868), (11.14.Value877), (11.14,Value3533), (11.2,Value3202), (11.31,Value6699), (11.32,Value7
023), (11.77,Value3391), (11.81,Value6215), (11.84,Value2988), (11.92,Value6099), (12.02,Value2612), (12.28,Value3971), (12.29,Value4365), (12.42,Value6303), (12.43,Value5350), (12.
51,Value4808), (12.66,Value5501), (12.85,Value5127), (12.99,Value1133), (13.0,Value5073), (13.25,Value684), (13.41,Value579), (13.44,Value1019), (13.7,Value6921), (13.74,Value5563),
 (13.81,Value6078), (14.12,Value5875), (14.14,Value5404), (14.15,Value1027), (14.25,Value5216), (14.49,Value3884), (14.55,Value4719), (14.75,Value6437), (14.75,Value7043), (14.97,Va
lue1572), (14.98,Value7159), (15.06,Value715)
null
Value9957,Value9983
Value9952
null
null
null
thunder:21%
```