ADVANCED DATA STRUCTURES

COP5536 FALL 2017

Instructor: Dr. Sartaj Sahni

Programming Project

(Implementation of Initialize, Insert and Search operations in B+ Tree)

Name: Akash Prakash Barve

UFID: 5546-6983

UF Email: akash.barve@ufl.edu

INDEX:

Associated Files

- b_plus_tree.java: This is the file which includes all the important functions to implement the B+ tree. These functions Initialize, Insert, Search, Range Search.

- node.java: The node.java defines the node of the tree. It contains method to handle overflow condition in the parent node.

- leaf_node.java: This file handles the logic for the leaf nodes. We use an array list of array list of the string so that we can handle duplicate keys. It contains methods to perform insert in sorted order as well as to update the list.

- index_node.java: The index_node file defines the index node of the tree. This file handles the insertion of keys in the index node and maintains them in sorted order.

- key_node_map.java: This class defines the mapping of the key and node. It associates the key to a particular value.

- treesearch.java: This file contains the main method. The main method handles the reading of input from a .txt file and printing output on a .txt file. It uses switch case to perform the operation required.

- makefile: A makefile is an easier way to handle and organize the compilation of code. We run the command ‘make’ and all the project classes are compiled.

Function Prototypes and Program Structure

The file b_plus_tree.java includes the following methods needed for the implementation of the given project assignment:

1. tree_insert(parameter1, parameter2):
    i. Parameters: Double key, String value
    ii. Return value: void
    iii. Description: The insert function takes the key and value as arguments and inserts it into the respective position.

2. get_child(parameter1, parameter2, parameter3):
    i. Parameters: node node, key_node_map entry, key_node_map new_child
    ii. Return value: key_node_map
    iii. Description: If the key to be inserted was not simply inserted at the leaf nodes, this function will handle the insertion.

3. split_leaf(parameter1):
    i. Parameter: leaf_node leaf
    ii. Return value: key_node_map
    iii. Description: This function handles the splitting of a leaf node and returns the new right node and the splitting key.

4. split_index(parameter1):
    i. Parameter: index_node index
    ii. Return value: key_node_map
    iii. Description: This function handles the splitting of an index node and returns the new right node and the splitting key.

5. value_search(parameter1):
    i. Parameter: Double key
    ii. Return value: ArrayList<String>
    iii. Description: This function takes key to be searched as the argument and returns the value of the leaf node if the key is at the leaf, else returns a null value.

6. tree_search(parameter1, parameter2):
    i. Parameters: node node, Double key
    ii. Return value: node
    iii. Description: This function searches if the key value is at an index node and returns value using linear searching.

7. range_search(parameter1, parameter2):
    i. Parameters: Double key1, Double key2
    ii. Return value: StringBuilder
    iii. Description: This function takes 2 keys as the arguments for the range search and by using the string builder returns the key values returned by this search in the form of a string.

The file node.java includes the following methods needed for the implementation of the given project assignment:

1. overflow():
    i. Return value: Boolean
    ii. Description: Handles the node overflow condition.

The file leaf_node.java includes the following methods needed for the implementation of the given project assignment:

1. leaf_node(parameter1, parameter2):
    i. Parameters: Double firstKey, String firstValue
    ii. Description: This function handles leaf node insertion for the first key and first value.

2. leaf_node(parameter1, parameter2):
    i. Parameters: ArrayList<Double> new_keys, ArrayList<ArrayList<String>> newValue
    ii. Description: This function handles leaf node addition for new key and value caused due to a split operation.

3. ordered_insert(parameter1, parameter2):
    i. Parameters: Double key, ArrayList<String> valueList
    ii. Return value: void
    iii. Description: This function inserts the keys into an array list in the sorted order.

4. update_list(parameter1, parameter2):
    i. Parameters: int index, String value
    ii. Return value: void
    iii. Description: Updates the list after each insert.

The file index_node.java includes the following methods needed for the implementation of the given project assignment:

1. index_node(parameter1, parameter2, parameter3):
    i. Parameters: Double key, node child0, node child1
    ii. Description: Constructor to handle a single insertion into the insert node during initial operation.

2. index_node(parameter1, parameter2):
    i. Parameters: List<Double> new_keys, List<node> newChildren
    ii. Description: For the insert caused by the split operation.

3. ordered_insert(parameter1, parameter2):
    i. Parameters: key_node_map e, int index
    ii. Return value: void
    iii. Description:This function inserts an entry into a node at the specified index in the sorted order.

The file key_node_map.java includes the following methods needed for the implementation of the given project assignment:

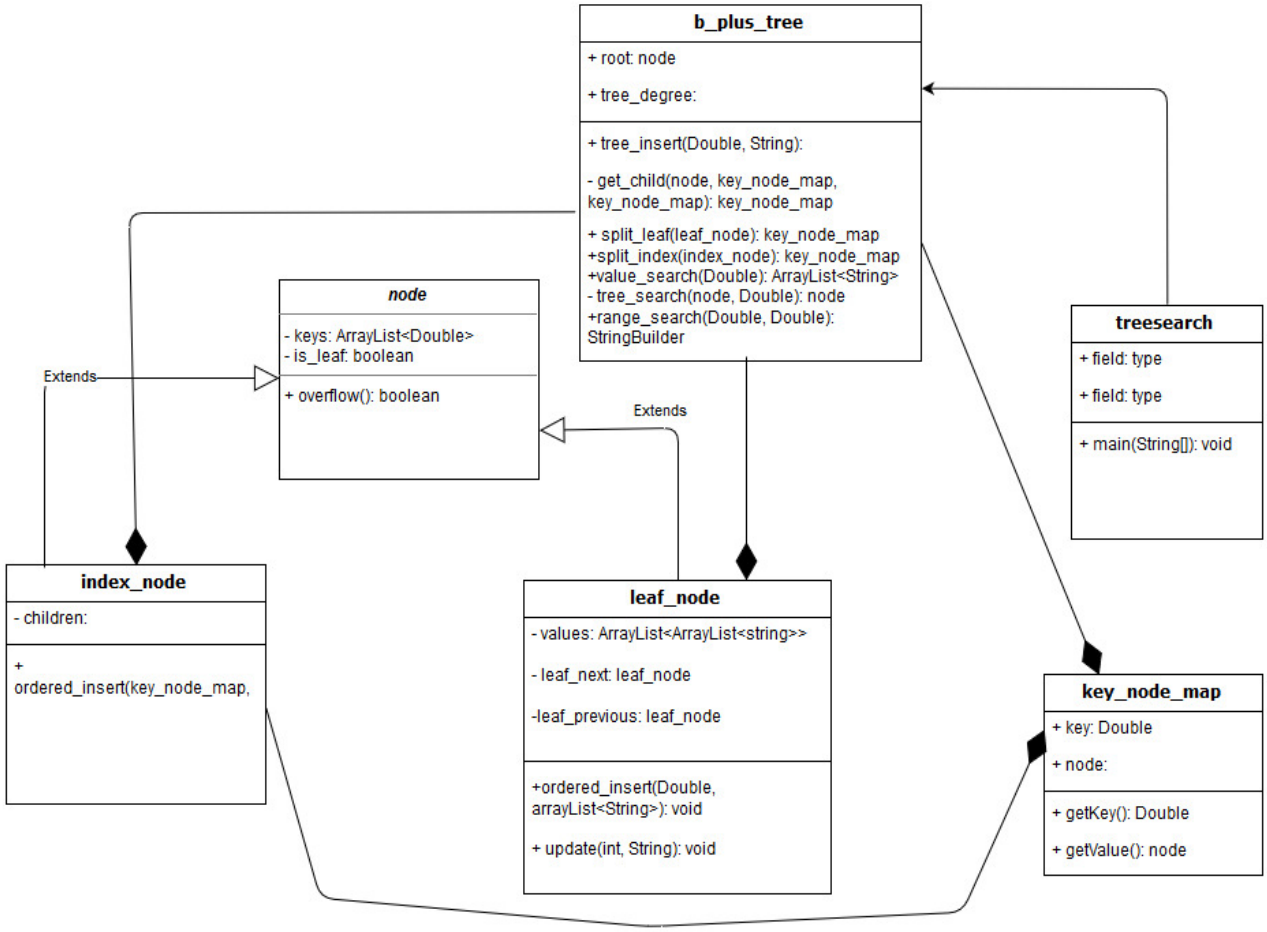
1. getKey():
    i. Return value: Double
    ii. Description: returns the key.

2. getValue():
    i. Return value: node
    ii. Description: returns the node.

The file treesearch.java includes the following methods needed for the implementation of the given project assignment:

1. main():
    i. Description: This is the main method which handles the input from input file taking its name from the command line argument. Prints output to the 'output_file.txt'. It uses switch case to perform the appropriate operation depending on the input. A nested switch case is used to handle the search and range search operation.

The structure of the project is as follows:



Running the Program

The project has been compiled and tested on the Windows 10 platform using the javac compiler.

It has been run on the thunder.cise.ufl.edu remote server to test its readiness before final submission.

Steps taken to execute the project after gaining access to the remote server are as follows:

1. Move all the files into a directory.
2. Use ‘cd /path’ to change directory to the file location.
3. Execute the ‘make’ command.
4. Run ‘java treesearch input_filename.txt’.
5. The output is generated as the ‘output_file.txt’ file.

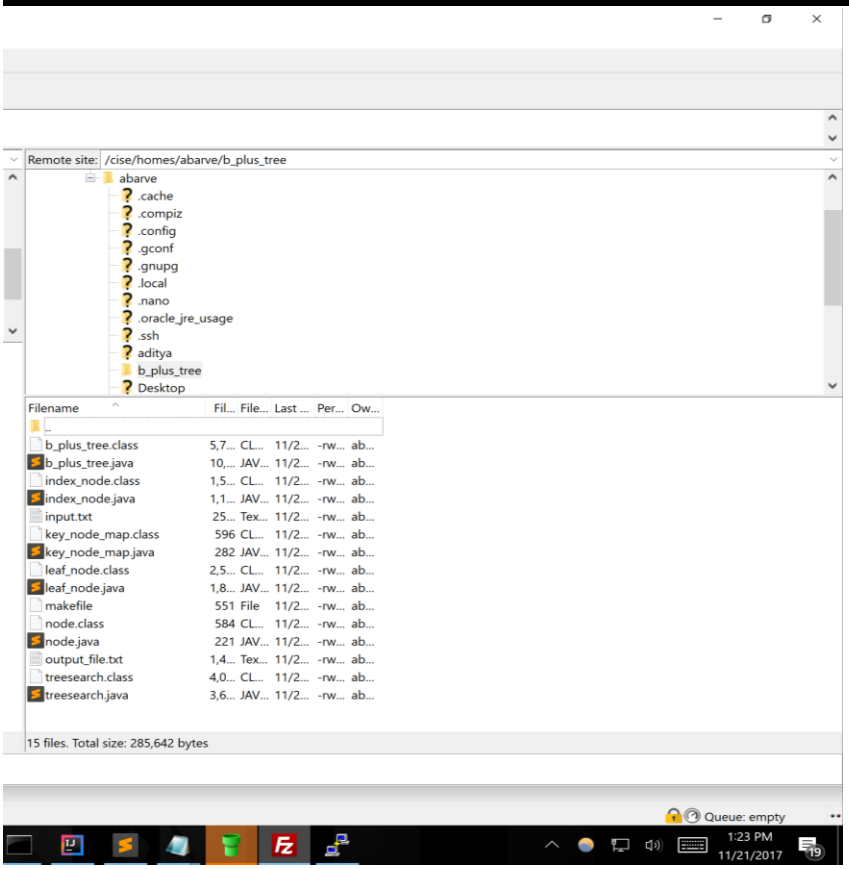
Summary

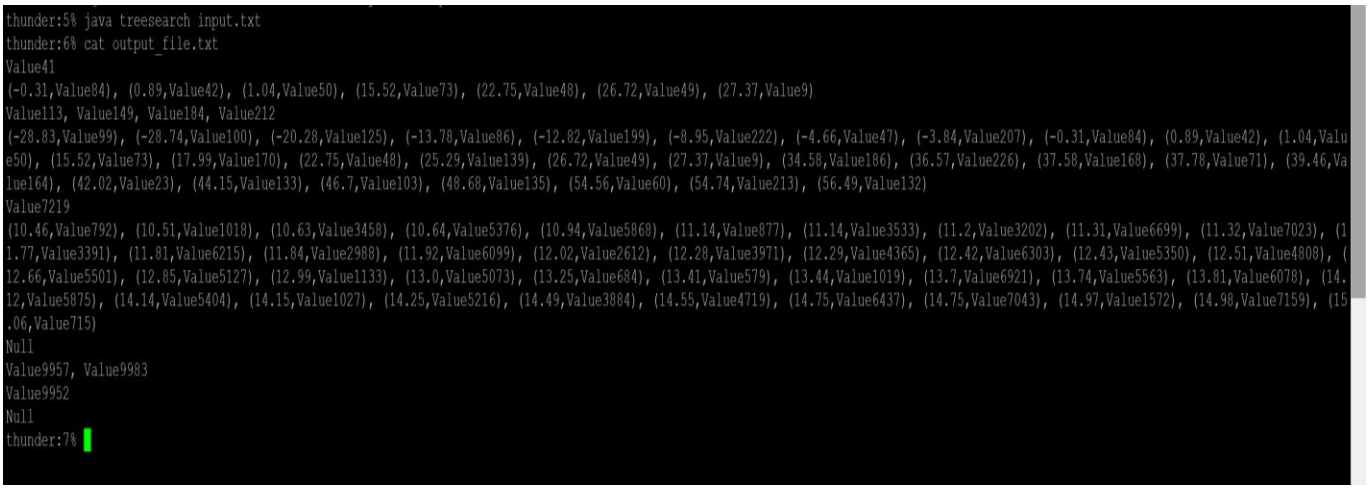
The program compiles successfully and generates accurate output for the given input file.

Following are the snapshots for the execution done:

thunder.cise.ufl.edu - PuTTY

```
thunder:3% cd b_plus_tree
thunder:4% make
javac -g b_plus_tree.java
make: Warning: File 'key_node_map.class' has modification time 2.5 s in the future
javac -g treesearch.java
make: warning:  Clock skew detected.  Your build may be incomplete.
thunder:5% java treesearch input.txt
thunder:6%
```




thunder:5% java treesearch input.txt
thunder:6% cat output_file.txt
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,Valu
e50), (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,Value71), (39.46,Va
lue164), (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,Value7023), (1
1.77,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,Value1572), (14.98,Value7159), (15
.06,Value715)
Null
Value9957, Value9983
Value9952
Null
thunder:7%