

COP 5536 Fall 2017

Programming Project

Name: Devanshu Singh

UFID: 6513 1155

UF Email: devanshu.singh@ufl.edu

Description: This programming project is a custom implementation of B+Tree. It has the following methods:

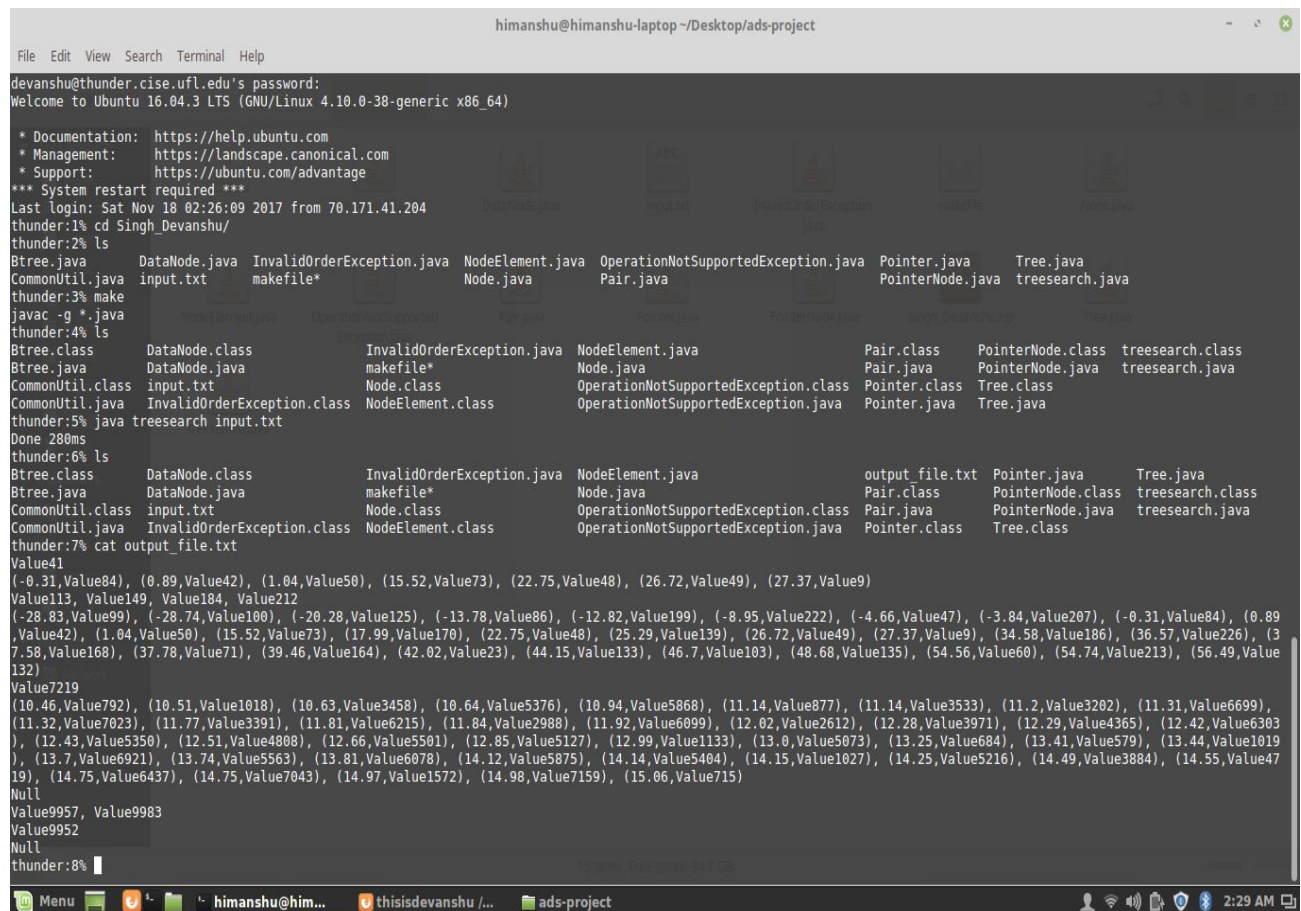
1. [Initialize\(order\)](#)
2. [Insert\(key,value\)](#)
3. [Search\(key\)](#)
4. [Search\(key1,key2\)](#)

Note: Since B+Tree are often referred to as Btree in the industry, the term “Btree” is used instead of “B+Tree“, both in this document and in the code.

Steps to execute: the Btree(B+Tree) implementation.

- Go to the directory with the java files.
- Run “make” command to compile the java files.
- Run “java treesearch <input file>” command to execute the program.
- The output will be written to output_file.txt in the same directory.

Here is a screenshot when the code was tested on *thunder.cise.ufl.edu* server.



```
himanshu@himanshu-laptop ~/Desktop/ads-project
File Edit View Search Terminal Help
devanshu@thunder.cise.ufl.edu's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-38-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
*** System restart required ***
Last login: Sat Nov 18 02:26:09 2017 from 70.171.41.204
thunder:1% cd Singh_Devanshu/
thunder:2% ls
Btree.java      DataNode.java  InvalidOrderException.java  NodeElement.java  OperationNotSupportedException.java  Pointer.java      Tree.java
CommonUtil.java input.txt      makefile*                  Node.java         Pair.java                               PointerNode.java  treesearch.java
thunder:3% make
javac -g *.java
thunder:4% ls
Btree.class      DataNode.class  InvalidOrderException.java  NodeElement.java  Pair.class      PointerNode.class  treesearch.class
Btree.java       DataNode.java  makefile*                  Node.java         Pair.java       PointerNode.java   treesearch.java
CommonUtil.class input.txt      Node.class                 OperationNotSupportedException.class  Pointer.class    Tree.class
CommonUtil.java  InvalidOrderException.class  NodeElement.class  OperationNotSupportedException.java  Pointer.java     Tree.java
thunder:5% java treesearch input.txt
Done 280ms
thunder:6% ls
Btree.class      DataNode.class  InvalidOrderException.java  NodeElement.java  output_file.txt  Pointer.java      Tree.java
Btree.java       DataNode.java  makefile*                  Node.java         Pair.class        PointerNode.class  treesearch.class
CommonUtil.class input.txt      Node.class                 OperationNotSupportedException.class  Pair.java         PointerNode.java   treesearch.java
CommonUtil.java  InvalidOrderException.class  NodeElement.class  OperationNotSupportedException.java  Pointer.class     Tree.class
thunder:7% 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,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,Value71), (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,Value7023), (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,Value1572), (14.98,Value7159), (15.06,Value715)
Null
Value9957, Value9983
Value9952
Null
thunder:8%
```

Input Format: The program takes a text file as input. The first line of the file has an integer value (m) which is the order of the Btree. If the order is less than 3 the program throws an InvalidOrderException. Rest of the lines have either Insert(key,value) , Search(key) or Search(key1,key2). Anything other than these results in OperationNotSupportedException.

Output Format: The output is generated only in case of search queries. The output is written in output_file.txt file. Single key search writes comma(,) separated values saved corresponding to the key on a new line. If a key is not present in the Btree the string "Null" is returned.Each range search query writes comma separated (key,value) pairs on a new line in the output file.

The B+ Tree has been implemented in Java. The project has the following Classes

Class (<i>in alphabetical order</i>)	Functionality
Btree	Implementation of Btree (B+ Tree). They are widely used in database indexing.
CommonUtil	Utility class for common utilities.(Example isNull method)
DataNode	DataNode is an implementation of the leaf Nodes of Btree. It has a parent node.The prev and next pointers form a doubly linked list of leaf nodes.
InvalidOrderException	Invalid Order exception is thrown when the order of the B+Tree is less than three.
Node	Node interface defines the types of nodes of a BTree.
NodeElement	NodeElement interface defines the various node elements stored in the nodes of the Btree.
OperationNotSupportedException	Operation not supported Exception throw for operations other than insert or search.
Pair	Pairs are the elements that are saved at the leaf Nodes of the Btree. They are of the form (key,value).
Pointer	Pointers are the elements that are stored at the non leaf nodes of the Btree.
PointerNode	PointerNode is the implementation for the non leaf nodes of the Btree (B+Tree).
Tree	Tree defines the various implementations of Tree Data Structure.
treesearch	This is the executor class of the ADS project Btree.

Btree

Method	Access Level	Return Type	Description
initialize(int order)	Public	Btree	Method to initialize the B+Tree which takes order of the Btree as input and returns an instance of new Btree.
insert(double key, String value)	Public	void	Method to insert a (key,value) pair into the Btree
search(double key)	Public	List<String>	Method to search for a particular key in the Btree. The method uses binary search on the data of the leaf node to ensure Big O $\log(m)$ time complexity, where m is the order of the Btree.
search(double key1, double key2)	Public	List<String>	Method to search elements of the Btree in a given range defined by the two input keys.
add(List<Pair> pairs, Pair pair)	Private	void	Method to add pair in the leaf node of the BTree (B+Tree)
findLeaf(double key, Node node)	Private	DataNode	Method to find the appropriate leaf node to insert the given key. The method uses binary search on the data of the index node to ensure Big O $\log(m)$ time complexity, where m is the order of the Btree.
split(Node old)	Private	void	Method to split the node which has exceeded its capacity (i.e. more than order - 1 elements)
splitPointerNode(Node old, int size)	Private	void	Method to split non-leaf Nodes that have reached capacity (i.e. more than order - 1 elements)
splitDataNode(Node old, int size)	Private	void	Method to split leaf Nodes that have reached capacity (i.e more than order - 1 elements)
add(List<Pointer> data, Pointer pointer)	Private	void	Method to add elements in non leaf Nodes of the Btree

Common Util

Method	Access Level	Return Type	Functionality
isNull(Object o)	Public	boolean	It is a utility function which checks if an object is null or not.

DataNode

Fields	Functionality
List<Pair> data	Stores the list of (key,Values) pair.
Node parent	Points to the parent of the node.
DataNode next	Used to maintain the doubly linked list of the leaf nodes.
DataNode prev	Used to maintain the doubly linked list of the leaf nodes.

InvalidOrderException

It is a custom exception invoked when the order of the tree is less than 3 during the creation of new Btree (B+Plus) object.

Node

Method	Access Level	Return Type	Functionality
getData()	Public	List<? extends NodeElement>	Used to retrieve the data stored at the node.
getParent()	Public	Node	Used to get the parent of the current node.
setParent(Node node)	Public	void	Used to modify or set the parent node of the current node.

NodeElement

It is an interface which is implemented by Pair Class and Pointer Class.

OperationNotSupportedException

It is a custom exception invoked when operations are other than Insert(key,value), Search(key) or Search(key1,key2).

Pair

Field	Functionality
double key	It stores the key of the (key,Value) pair at the leaf node where the actual data is stored.
List<String> value	Each key has a list of values attached to it. The list helps maintain duplicates as expected in the problem statement.

Pointer

Field	Functionality
double key	This field stored the key at the non leaf nodes to guide the search to the appropriate leaf node.
Node left	This field points to the subtree which holds values smaller than equal to the key value.
Node right	This field points to the subtree which holds values greater the key value.

PointerNode

Field	Functionality
List<Pointer> data	Data field stores the pointers at the non leaf nodes of the B+Tree.
Node parent	This field points to the parent of the current node.

Tree

Method	Access Level	Return Type	Functionality
insert(double key, String value)	Public	void	This method is implemented differently by all the different implementations of the Tree interface.
search(double key)	Public	List<String>	This method is implemented differently by all the different implementations of the Tree interface.
search(double key1, double key2)	Public	List<String>	This method is implemented differently by all the different implementations of the Tree interface.

treesearch

Method	Access Level	Return Type	Functionality
main(String args[])	Public	void	This is the main function and it handles the I/O and logic for the execution of the queries.
search(BufferedWriter bufWriter, String input,Btree btree)	Private	void	This functions determines the type of the search and writes back the result into the output file.
keySearch(Btree btree, String[] inputPair)	Private	String	This function handles the single key search and returns the search result.
rangeSearch(Btree btree, String[] inputPair)	Private	String	This function handles the range search and returns the search result.
insert(String input, Btree btree)	Private	void	This function directs the execution to the part of the code that inserts (key,Value) pairs into the Btree.