

ADVANCED DATA STRUCTURES

(COP 5536), Spring 2021

Assignment-1

Programming Project Report

(B+ TREE)

Submitted by

Name: Goel,Saksham

UFID: 92792975

Email-ID: goelsaksham@ufl.edu

Project Introduction

B tree is a data structure which helps to store and maintain data into a tree. The data is in the format of (Key, Value)pair. B trees are self balancing trees and used to maintain multilevel indexing.

There are two types of B Trees:

- 1) B- tree
- 2) B+ tree

In this project we are implementing B+ tree. In the B- tree the data is stored in internal nodes as well as in leaf nodes which reduce the numbers of elements that are present into the nodes. Due to this the number of levels of a tree are increasing and the time for searching an element is also increasing due to increment in the height of a tree. This is the drawback of using a B- tree for multilevel indexing. But on the other side B+ tree provide the advantage for this condition. In B+ tree all the data is stored in the leaf nodes of the tree which helps to reduce the time for searching an element because of decrement in the levels of the tree. B+ tree is also known as an advanced self balancing tree.

Properties of B+ Tree:-

1. B+ Tree is a m-way search tree with some rules.
2. The level of all the leaf nodes must be the same.
3. It follows the bottom-up creation process.
4. If the Root is a Non-leaf node then it has at least two child nodes.
5. If the tree is of m-order then,
 - ☐ Every internal node (except root) has at least $\text{ceil}(m/2)$ child nodes.
 - ☐ Every internal node has at most m child nodes.
 - ☐ Minimum no. of keys at every node is $(\text{ceil}(m/2) - 1)$.
 - ☐ Maximum number of keys at every node is (m-1).

This project is focused on five fundamental features:

1. The first task is Degree initialization of B+ tree.
2. Inserting the element (key,value) pair in B+ tree.
3. Deleting the (key,value) pair from B+ tree.
4. Searching the value for a particular key.
5. Searching the range of values between two keys (Key1, Key2).

Programming Environment

Software:

Programming Language: C++

Compiler: g++ compiler

Hardware:

Operating Systems: Windows 10

RAM: 8GB

Hard Disk: 1TB

Project structure

The Project has been coded in C++ language. Files used in this project are as follows

1. Source Code: The source file name is “bplus.cpp” and the format of this file is .cpp, This file contains the main function including all other functions. All function names are listed below:-

- Main()
- Leaf_Node_Split()
- Non_Leaf_Split()
- Node_merge()
- Node_rearrange()
- element_Search()
- element_search_range()
- element_Insert()
- element_Delete()

(The functionality of each function has been defined below in the function prototype)
Code implementation is present in “bplus.cpp” file.

2. Input File: The input file name is “input.txt” and the format of this file is .txt, This file is used to input the data and on the basis of input the program performs different operations. The different input names are listed below:

- Initialize(m): This is the first line of input which initializes the degree (m-order) of the B+ tree.
- Insert(Key, value): If the input is insert that means to insert an element in B+ tree into its correct position. It invokes element_insert() function from main() function.
- Search(Key): If the input is search (with a particular key) that means to search that particular key value. It invokes the Search() function from the main() function.
- Search(key1, Key2): If the input is search (with two keys) that means to search the value of all the nodes which lie between those two keys. It invokes element_search_range() from the main function.
- Delete(Key): If the input is delete that means to delete the (key, value) pair from the tree w.r.t key position.

3. Make File: This “Makefile” is used to compile the project and make executable file of the code

4. Output File: The output file name is “output_file.txt” and the format of this file is .txt, this file contains all the generated outputs.

Structure of program:

```
struct Node {
    Node * prtNode;           /*parent node*/
    int elmt;
    Node * childrenNode[1000]; /*Child Node*/
    int value[1000];

                                /*construct a Node*/
    Node() {
        prtNode = NULL;
        elmt = 0;
        int a;
        for (a = 0; a < 1000; a++) {
            value[a] = inf;
            childrenNode[a] = NULL;
        }
    }
};
```

Functions

Methods Used:

1. int main ()
2. void Leaf_Node_Split(Node *crtNode)
3. void Non_Leaf_Split(Node *crtNode)
4. void node_merge(Node *leftchildNode, Node *rightChildNode, bool is_leaf, int Right_position)
5. void Node_rearrange(Node *leftchildNode, Node *rightChildNode, bool is_leaf, int left_position, int iscrtNode)
6. void element_Search(Node *crtNode, int val, int crtposition)
7. void element_search_range(Node *crtNode, int val1, int val2, int crtposition)
8. void element_Insert(Node *crtNode, int val)
9. void element_Delete(Node *crtNode, int val, int crtposition)

Function Prototypes

1. **main ():-**

The main function executes all the initial parameters of the program. It takes the input file, After the input file is parsed it retrieves the following functions (given below) and performs accordingly.

- Initialize (int degree)- This function basically Initializes the B+ tree of m-order.
- Search (int key) –This function gets the value of the entered key.
- Search (int key1, int key2) – This function gets the value of between entered key1 and key2
- Insert (int key, float value)-This function takes the key value pair and inserts it into the tree.
- Delete (int key) – This function Deletes the key, value pair from the tree.

The input parameters is the input filename.

2. **void Leaf_Node_Split(Node *crtNode)**

When the leaf node gets excess elements then to maintain the degree of the tree, we call Leaf_Node_Split function which helps to split the node and it's parameter is the position of the key.

3. **void Non_Leaf_Split(Node *crtNode)**

When the non leaf node (internal node) gets excess elements then to maintain the degree of the tree, we call the Non_Leaf_Split function which helps to split the node and its parameter is the position of the key.

4. **void node_merge(Node *leftchildNode, Node *rightChildNode, bool is_leaf, int Right_position)**

When we have to merge the splitted nodes w.r.t the key position then we perform the node_merge function.

5. void Node_rearrange(Node *leftchildNode, Node *rightChildNode, bool is_leaf, int left_position, int isrcrtNode)

When we have to rearrange the nodes to get the right elements at their respective positions then we call function Node_rearrange.

6. void element_Search(Node *crtNode, int val, int crtposition)

When we have to retrieve a key, value pair from the input file and return that retrieved value in output file, then we call function element_Search

7. void element_search_range(Node *crtNode, int val1, int val2, int crtposition)

First we traverse the position of key1 and then we retrieve all the nodes whose value is less than the value of key2. This function is used when we have to retrieve all the values of elements between two keys [key1, key2] in the tree then we call element_search_range function. We used current node pointer, key1, key2 and current position are the parameters for this function.

8. void element_Insert(Node *crtNode, int val)

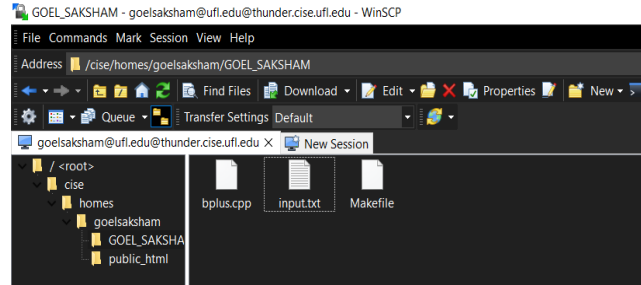
When we have to insert a new (key, value) pair in the tree, we perform element_Insertion function. By using this function we check while traversing if it's a non leaf node (internal node) or a leaf node, We perform different operations according to the condition and the parameter of this function is the (key, value) pair which has to be inserted.

9. void element_Delete(Node *crtNode, int val, int crtposition)

When we have to Delete an element (key, value) pair from the tree, then we perform the element_Delete function and we get the key which has to be deleted from the input file and then Delete it from the tree.

Compiling and Running the Project

- Login to server 'thunder.cise.ufl.edu' with your cise account credentials to upload the Project on thunder server using "winSCP".



Project has been uploaded on the server.

- Login to server 'thunder.cise.ufl.edu' with your cise account credentials to compile and run the project on linux terminal using "PuTTY".

```
thunder.cise.ufl.edu - PuTTY
login as: goelsaksham@ufl.edu
goelsaksham@ufl.edu@thunder.cise.ufl.edu's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat 03 Apr 2021 09:15:59 PM EDT

System load:  0.0           Processes:            276
Usage of /:   50.3% of 30.88GB Users logged in:      2
Memory usage: 1%           IPv4 address for ens3: 128.227.205.239
Swap usage:   0%

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

0 updates can be installed immediately.
0 of these updates are security updates.

*** System restart required ***

#####
#  UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED  #
#                                                     #
#  You must have explicit permission to access this  #
#  device. All activities performed on this device   #
#  are logged. Any violations of access policy will   #
#  result in disciplinary action.                     #
#####

Last login: Sat Apr  3 21:12:18 2021 from 10.228.54.220
thunder:~> cd GOEL_SAKSHAM/
```

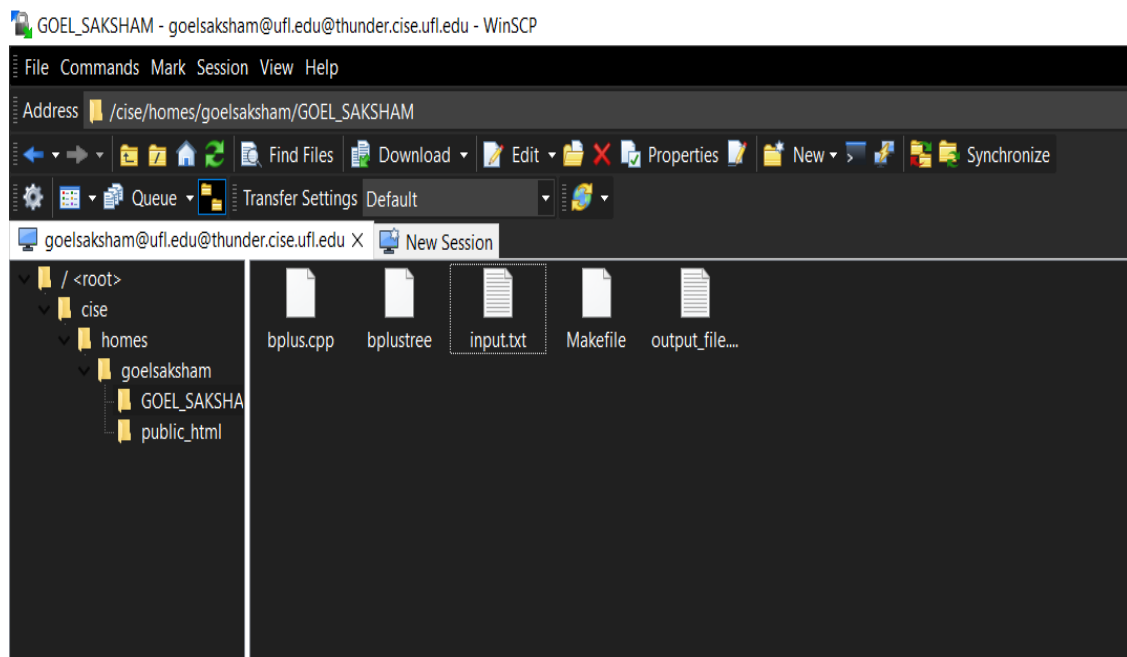
Linux Terminal to run the project

- Run MakeFile to compile C++ Program and Run g++ command with the filename as input argument.

```
Last login: Sat Apr  3 21:12:18 2021 from 10.228.54.220
thunder:~> cd GOEL_SAKSHAM/
thunder:~/GOEL_SAKSHAM> make
g++ -std=c++11 bplus.cpp -o bplustree
thunder:~/GOEL_SAKSHAM> ./bplustree
thunder:~/GOEL_SAKSHAM>
```

Compilation of a code with the make command and run the program with g++ command

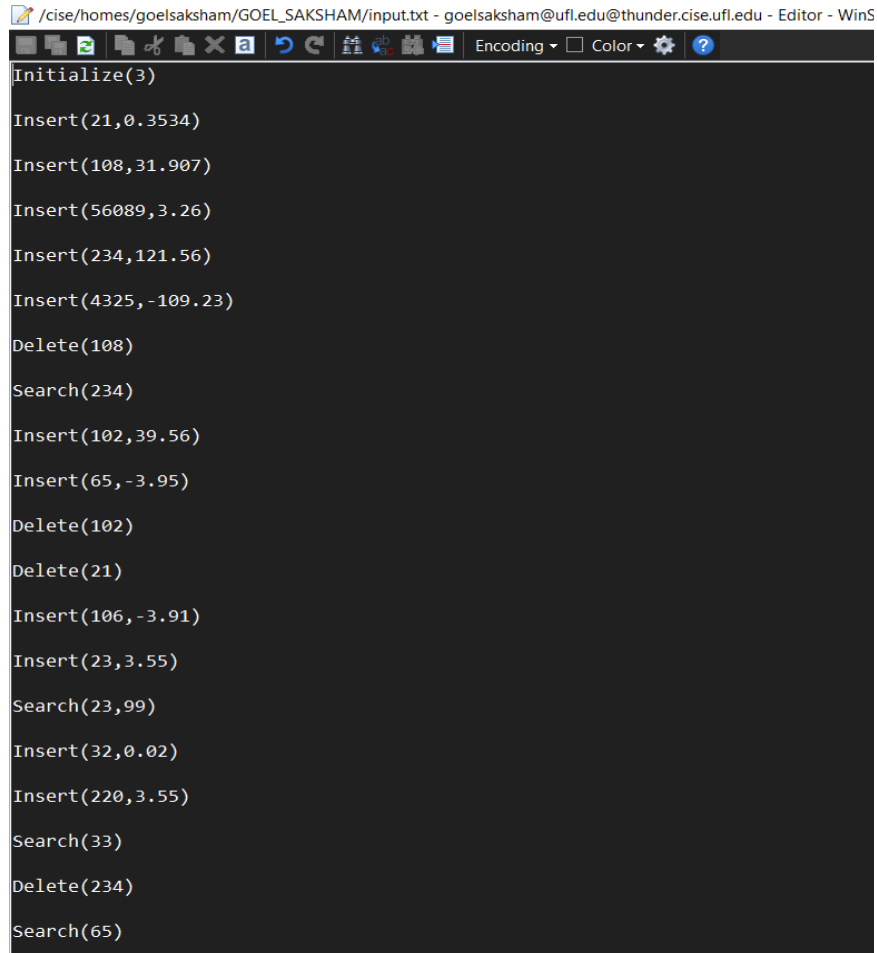
- An output file will be generated in the .txt form. The file name is “output_file.txt”.



Output File is generated

Input and Output file

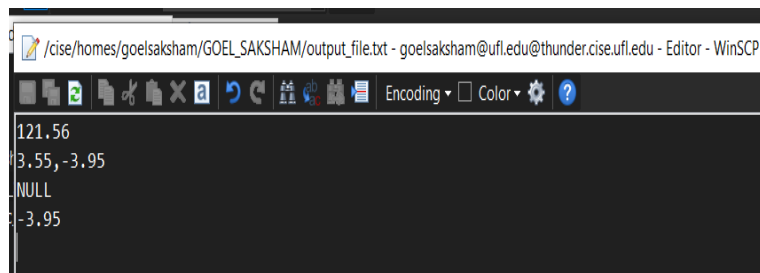
Input.txt



The screenshot shows a text editor window titled "/cise/homes/goelsaksham/GOEL_SAKSHAM/input.txt - goelsaksham@ufl.edu@thunder.cise.ufl.edu - Editor - WinS". The editor contains the following text:

```
Initialize(3)
Insert(21,0.3534)
Insert(108,31.907)
Insert(56089,3.26)
Insert(234,121.56)
Insert(4325,-109.23)
Delete(108)
Search(234)
Insert(102,39.56)
Insert(65,-3.95)
Delete(102)
Delete(21)
Insert(106,-3.91)
Insert(23,3.55)
Search(23,99)
Insert(32,0.02)
Insert(220,3.55)
Search(33)
Delete(234)
Search(65)
```

Output_file.txt



The screenshot shows a text editor window titled "/cise/homes/goelsaksham/GOEL_SAKSHAM/output_file.txt - goelsaksham@ufl.edu@thunder.cise.ufl.edu - Editor - WinSCP". The editor contains the following text:

```
121.56
3.55,-3.95
NULL
-3.95
```