Advanced Data Structures (COP 5536)

Spring 2019

Programming Project Report File

Submitted By:

Name : Anubhav Jha
UFID : 43393979
UF Email Id : ajha1@ufl.edu

Index

Project Requirement

In the project we were asked to implement a B+ Tree which supports some of the basic operations.

B+ Tree is an extension of B Tree which allows efficient insertion, deletion and search operations.
In B Tree, Keys and records both can be stored in the internal as well as leaf nodes. Whereas, in B+ tree, records (data) can only be stored on the leaf nodes while internal nodes can only store the key values.
The leaf nodes of a B+ tree is linked together in the form of a singly linked lists to make the search queries more efficient.
B+ Tree are used to store the large amount of data which cannot be stored in the main memory. Since, size of main memory is always limited, the internal nodes (keys to access records) of the B+ tree are stored in the main memory whereas, leaf nodes are stored in the secondary memory.[1]

The B+ Tree is supposed to perform the following 4 operations:

a. Initialize (int m): Initialize a B+ Tree of order m.
b. Insert (key, value): This operation will insert a key in the B+ Tree and value in the leaf.
c. Search(key): This operation will search the key.
d. Search (key1, key2): This operation will search the keys in the range between key1 and key2.
e. Delete(key): This operation will delete the key and the value associated with it from the tree.

The output of the different functions

a. Insert: No output
b. Search: The output will have the value associated with the key.
c. Delete: No output

Language Used

The language used is C++.

Included in File

The zip file contains the following

1. Source Files
   1.1. internalnode.cpp
   1.2. leafnode.cpp
   1.3. main.cpp
   1.4. main.h
   1.5. node.cpp
   1.6. tree.cpp
2. Makefile
3. Report

How to run the code:

1. Use cd jhabplustree
2. Use command make
3. ./bplustree input_file_name

C++ Classes

Node: super class of node object.

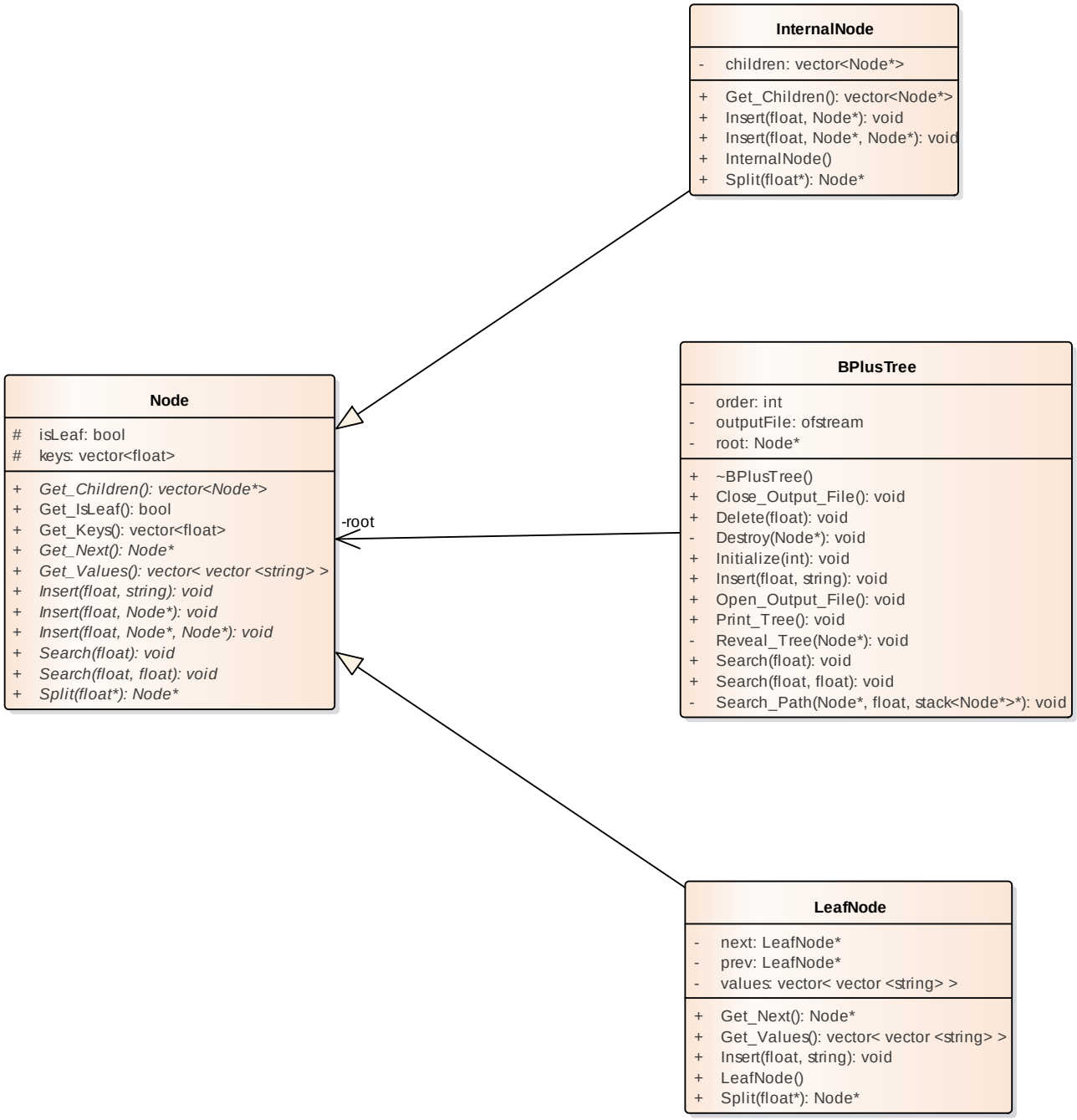
LeafNode: This is the class of leaf node inherited from Node class.

InternalNode: This is the class of internal node inherited from Node class

BPlusTree: B+ tree class of order m

Main: The class which contains on how the file will be read and give output in the file

Class Diagram



Project Structure

BPLUSTREE

ATTRIBUTES
order : int Private
outputFile : ofstream Private
root : Node* Private

ASSOCIATIONS	
Association (direction: Source -> Destination)	
Source: Public (Class) BPlusTree	Target: Private root (Class) Node

OPERATIONS	
~BPlusTree () : Public	destructor for tree
Close_Output_File () : void Public	function to close the output file
Delete (key : float) : void Public	
Destroy (node : Node*) : void Private	function to destroy the tree
Initialize (m : int) : void Public	operation: Initialize(m)
Insert (key : float , value : string) : void Public	operation: Insert(key, value)
Open_Output_File () : void Public	function to open the output file
Print_Tree () : void Public	

OPERATIONS
function to print the current state of the tree
Reveal_Tree (node : Node*) : void Private function to reveal the contents of the B+ tree
Search (key : float) : void Public operation: Search(key)
Search (key1 : float , key2 : float) : void Public operation: Search(key1, key2)
Search_Path (node : Node* , key : float , path : stack<Node*>*) : void Private function for searching from root to leaf node and pushing on to a stack

InternalNode

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from InternalNode to Node

ATTRIBUTES
children : vector<Node*> Private

LeafNode

OPERATIONS	
Get_Children () : vector<Node*> Public	getter function for accessing children
Insert (key : float , rightChild : Node*) : void Public	function for insertion in an internal node
Insert (key : float , leftChild : Node* , rightChild : Node*) : void Public	function for insertion in a new internal root node
InternalNode () : Public	constructor for internal node
Split (keyToParent : float*) : Node* Public	function for splitting an internal node

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from LeafNode to Node

ATTRIBUTES
next : LeafNode* Private
prev : LeafNode* Private
values : vector< vector <string> > Private

Node

OPERATIONS

Get_Next () : Node* Public

getter function for accessing the next pointer

Get_Values () : vector< vector <string> > Public

getter function for accessing values

Insert (key : float , value : string) : void Public

function for insertion in a leaf node

LeafNode () : Public

constructor for leaf node

Split (keyToParent : float*) : Node* Public

function for splitting a leaf node

OMING STRUCTURAL RELATIONSHIPS

Generalization from LeafNode to Node

Generalization from InternalNode to Node

ATTRIBUTES

isLeaf : bool Protected

keys : vector<float> Protected

ASSOCIATIONS

Association (direction: Source -> Destination)

Source: Public (Class) BPlusTree

Target: Private root (Class) Node

OPERATIONS

Get_Children () : vector<Node*> Public

Properties:
bodyLocation = classDec

OPERATIONS
Get_IsLeaf () : bool Public getter function for accessing isLeaf
Get_Keys () : vector<float> Public getter function for accessing keys
Get_Next () : Node* Public Properties: bodyLocation = classDec getter function for accessing the next pointer
Get_Values () : vector< vector <string> > Public Properties: bodyLocation = classDec getter function for accessing values
Insert (key : float , value : string) : void Public Properties: bodyLocation = classDec function for insertion in a leaf node
Insert (key : float , rightChild : Node*) : void Public Properties: bodyLocation = classDec
Insert (key : float , leftChild : Node* , rightChild : Node*) : void Public Properties: bodyLocation = classDec
Search (key : float) : void Public Properties: bodyLocation = classDec
Search (key1 : float , key2 : float) : void Public Properties: bodyLocation = classDec
Split (keyToParent : float*) : Node* Public

OPERATIONS
Properties: bodyLocation = classDec

Reference

1. https://www.javatpoint.com/b-plus-tree