Advance Data Structures

COP 5536

Project

Name:	Mohit Garg
UFID:	9013-4089
UF Email:	mohitgarg@ufl.edu

Project Description

This project is about implementation of B+ trees in C and C++. B+ tree is a data structure used for storing Key, Value pairs. The attribute of B+ tree which differentiates it from other data structures is the same level of its leaf nodes. All the elements are always present in the leaf node. BPlusTree always have all the external nodes i.e. leaf nodes at same level. So, It's a self-balancing data structure with complexity of log(x) time. During insertion it checks the order of B+ tree. If the total number of inserted key value pair in node reach to the order value, then it will split the node and distribute the data equally. The data stored in a B+ Tree is in the sorted order.

This assignment is to implement the four functionalities of B+ Tree:

1. **Initialization:** The first & foremost step was to initialize a tree of order 'm' of the B+ tree.
2. **Insert Key Value Pair:** Insertion is done by calling insert function and passing the key and value as parameter. It inserts the key at its correct sorted position. So that we get a properly sorted tree. The inserted value then goes to the leaf node and final get its position there.
3. **Delete:** a function to delete the value of a key-value pair.
4. **Search:** it will search a key in the B+ Tree in log(n) time and then return the value associated to that key.
5. **Search Range:** it will search the range of keys i.e. Key1<=keys<=Key2 in the B+ tree. Then it will print the values associated with that key range.

Programming Environment

Software Required:

GUI: Codes Block version 11+

Compiler used : TMD-GCC-64

Language Used: C and C++

Operating system: Windows 10

Hardware Required:

RAM: 8 Gigabytes

ROM: 1 Terabytes

Project contains:

1) Main.cpp file (source code)
2) Report.pdf file
3) Make file
4) Input.txt file

Structure of the program

HEADER FILES

```
#include<bits/stdc++.h>
#include<stdio.h>
```

GLOBAL VARIABLE

```
order                 //FOR TAKING THE ORDER OF THE BPLUSTREE
MAX=50;               // FOR MAXIMUM CAPACITY OF A NODE.
searchTrigger =1;     //IT WILL SET ITS VALUE TO 1 WHEN VARIABLE NOT FOUND
ofstream outputFile;  //TO GENERATE OUTPUT FILE
```

STRUCTURE OF THE NODE

```
struct Node{
   int totalNodes;            //TOTAL NUMBER OF ACTIVE NODES AT ANY INSTANCE
   Node *parentNode;          //PARENT NODE
   int key[MAX];              //DEFINING KEYS
   float values[MAX];         //DEFINING VALUES
   Node *childNode[MAX];      //CHILD NODE
   Node(){                    //CONSTRUCTOR FOR NODE INITIALIZATION
     //ALL INITIALIZATIONS HERE
   }
};

struct DList{                 //Doubly linked list for the leaf node
int key;                      //key of DList
float value;                  //value of DList
DList *next;                  //next pointer to point next node
DList *prev;                  //previous pointer to point to the previous node
}*START=NULL,*Temp1=NULL,*q=NULL;//some supporting pointer like START NODE
```

FUNCTIONS USED

- ✓ void splitLeaf(Node *currentNode):

  Description: This function will split the leaf node when the data is entered and the overflow condition triggers. This method is called in insertNode function only.

  To split the leaf node if the totalnodes in the Node is equal to order m. Then the insertNode function call this function to split the nodes into left and right child and merge them into the parent node.

  Input type: the input parameter contains Node pointer. Which have newly inserted node's reference.

- ✓ void splitNonLeaf(Node *currentNode):

  Description: This function will split the NON-Leaf node when the data is inserted from the bottom node and the overflow condition triggers. This method is called in insertNode function only.

  To split the NON-leaf node if the totalnodes in the Node is equal to order m. Then the insertNode function call this function to split the nodes into left and right child and merge them into the parent node.

  Input type: the input parameter contains Node pointer. Which have newly inserted node's reference.

- ✓ void insertNode(Node *currentNode, int keyVal,float value):

  Description: This function will insert the new node by handling all the functions. In parameters it will receive the rootNode pointer, key and the value.

To insert a new node into the program. It first Traverse to the leftmost position in the B+ tree which is less than the already inserted value. Then create a new node pointer and copies the values of key and value into that new node. It then updates the total nodes value in the current node and also updates the parent node values if required. It uses recursion to traverse to the right position.

Input type: current Node reference in which the value have to be insert, The integer Key and float Value.

- ✓ void redistributeNode(Node *leftNode, Node *rightNode, bool isLeaf, **int posOfLeftNode, int whichOneiscurrentNode):**

  Description:

  - This function will redistribute the weight or the height of the BPlusTree "when user deletes a node and it got imbalance".
  - If the values in the node is less than half of the maximum order defined than delete node will call the redistributeNode.

  This function copies the pointer from right node to left node by giving its child pointer from right to left and deleting the right most value from its current node. Before that leftnode copies the value in its leftmost position and updates its parent and child pointer.

  Input type: leftNode pointer, RightNode pointer, Boolean value defining whether the splitting node is leaf node or an internal node, Position of left node and the current node number.

- ✓ void mergeNode(Node *leftNode, Node *rightNode, bool isLeaf, int posOfRightNode);

  Description: The merging of nodes is done when the node become the weight deficient and the need to merge in order to get balanced parent and child associate with it.

  This function will merge the left and the right node if the total values in these nodes is less than the half of order m. the position of rightnode will help in copying the values of the right node to the left node. Then update the total number of left pointer variables value by left+right(total values).After that, setting the parentNode in each of the newly added pointer in current node. And setting the child pointer.

  Input type: left node pointer, right node pointer, Boolean isLeaf and the position of right node.

- ✓ void deleteNode(Node *currentNode, int val, int currentNodePosition):

  Description: This function Deletes the whole node associated with the "Key" as a position of that node. After execution of this function we go directly into the correct position of the function and delete the value and update the pointer.

  To delete a node having key value equal to val. It first traverse the tree upto the correct position of the value. Then if the value is present in the linked list then it will delete the value otherwise do nothing. If the node became less than half of the order m then it will call the re-distribute function and merge function.

  Input type: current node pointer, value to delete, and current position of the node.

- ✓ void Delete(int key);

  Description: This function triggers the deleteNode function by receiving the key from the user.

To call the deleteNode function by passing rootnode pointer, integer Key and the current position as 0. This is the calling function.

Input type: integer key value received from input file**.**

- ✓ void Search(Node *currentNode, int val, int temp);

  Description: This function Search the node upto the depth of the key value, then prints the value associate with that key. If there is no value found it will do nothing.

  Do the searching of the node by traversing the B+ tree. When we get the value it will trigger a clasy variable to 1. And exit from the function. If value not found then it will print "NULL" in the output file. Otherwise it will print the value associated with the key.

  Input type: current node pointer, key to find, the current right node value.

- ✓ void Search(int k);

  Description: it will take the key from the user to search the value in the program and call the search function by passing the rootnode, key and current position of that parent pointer.

  This is one argument search function to call the Search function by passing the rootNode pointer and the key value along with the 0 as initial position of the node.

  Input type: integer Key value.

- ✓ void Search(Node *currentNode, int val1,int val2, int temp):

  Description: This function Search the range of nodes values, then prints the value associate with that key. If there is no value found it will do nothing.

Do the searching of the node by traversing the B+ tree. When it finds the key who have value KEY1<=key<=KEY2. Then it will print the value associated with that key. Else If value not found then it will print "NULL" in the output file.

Input type: current node pointer, key1 and key2 to find, the current right node value.

- ✓ void Search(int k,int k2):

  Description: It will take the key1 and key2 from the user to search the values within that range of the program and call the search function by passing the rootnode, key and current position of that parent pointer.

  This is two argument search function to call the Search function by passing the rootNode pointer and the key1 & key2 value along with the 0 as initial position of the node. If the value not found the it will print null.

  Input type: integer Key1, integer Key2.

- ✓ void Initialize(int x);

  Description: This function will set the order of the BPlusTree by changing the value of the global variable.

  It initializes the B+ Tree by setting the global "order" variable to x.

  Input type: integer variable m.

- ✓ void Insert(int x,float y):

  Description: This function call the Insertnode function by passing the rootnode as a parent node, the key value and the Value.

Call the insertNode function and insert the variables x as key and y as value. It passes the rootnode pointer and 0 as current position.

Input type: integer x as keyand float y as value.

- ✓ int main();

Description: It contains the "*input.txt*" reading function which will take the input file readings and after execution. It will make a file named as "*output.txt*".

This is the main function of the program which contains input file reading code and output file generating code. The input file is read line by line by comparing the function name. is the function name is same then by if and else loop, it will go in their respective function and split the string by "(" , "," and ")" to identify the input numbers. After this we just call the method and generate the output.

DIAGRAM OF THE PROGRAM