**B Plus Tree Project**

**Design:**

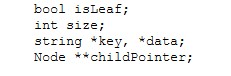
For the design aspect of the project, we have configured four class implementation files and three headers files for the program. The three classes that are configured are Main.cpp, Input.cpp, BPlusTree.cpp, and CatalogFile.cpp. There are corresponding header files with the same names but with the h extension.

First we have the Main.cpp file running at the start. Afterwards, we start by loading up the partfile.txt from the directory and then inserting each product and description into a variable that is implementing the BPlusTree class from BPlusTree.cpp.

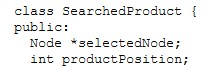
The BPlusTree.cpp file holds a class called BPlusTree which holds all functions related to what we need to implement for this project. It also holds a Node class which contains a couple of properties in the below image. The key and data variables are pointers which will be array variables that hold up to 16 pieces of keys and data. The isLeaf variable will determine whether or not the node is a leaf and the childpointer is an array of pointers which point to another node.

Once all of the products are loaded up, the user will then be presented with options, such as

Insert, Delete, Search, and Exit. If Insert or delete is selected, they would insert an ID first. If the ID exists, insert can’t be done, but delete can be done. Otherwise if ID doesn’t exist, then insert can be done, but delete can’t be done. When inserting, a product description has to also be inputted as well. Afterwards, it will be placed in the tree.

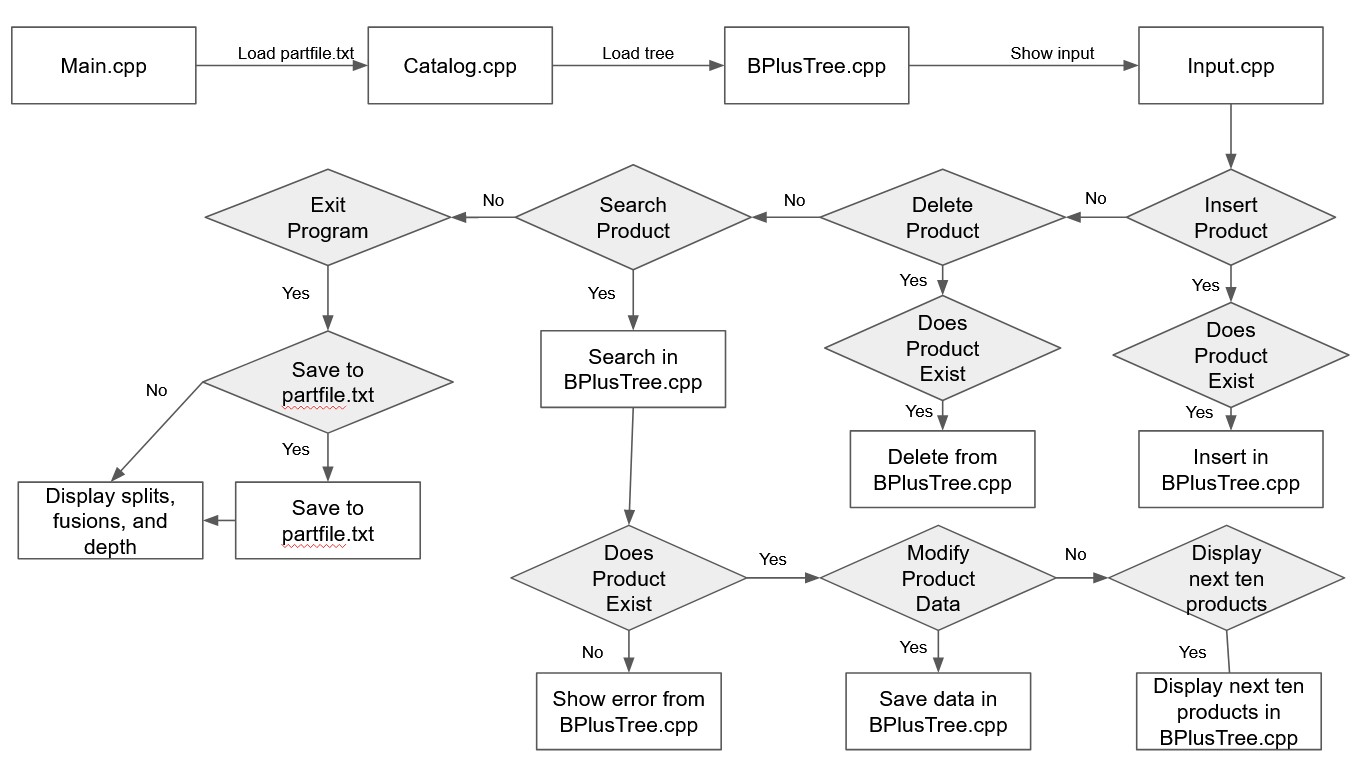


Search has some extra functionality. We use the SearchedProduct class in the BPlusTree.cpp file. This class contains a Node variable that will contain the node the product is in and an int variable that contains the position where the product and data are in.



When you search for a product ID and it exists, then you have three options. You will be able to modify the product description, display the next ten product, or go back to the main menu.

When you exit the program, you will be given an option to either save the current tree to the partfile.txt or not save it. Whatever option is chosen, you will be given a final output of splits, fusions, and depth of the tree.



**Algorithms:**

The two algorithms that were mainly used for the B plus tree are insert and delete. The insert and delete algorithms are shown below in pseudo code.

class Node { bool isLeaf;

Node \*\*pointer; int size;

string \*key, \*data;

}

class BPlusTree {

Node \*root;

}

**Insert**

insert(string key, string data) {

Node \*current;

Node \*parent;

If (root is NULL) add key and data in root

return

while (current is not a leaf) If (key < current.key) go to pointer before current.key

If (current.size < max node size)

Add key and data in node

else

split node into two insert key and data into the correct place if (parent.size < max index node size) add first key from new node to parent add pointer to the new node

else while (parent.size > max index node size) split parent insert first key from new index node to parent node add pointer to the new node current = parent parent = find the parent of the parent variable

}

**Remove**

remove(string key, string data) {

Node \*current;

Node \*parent;

If (root is NULL) return

while (current is not a leaf) If (key < current.key) go to pointer before current.key

If (current.size > (max node size / 2)) remove key and data in node

else join left or right node into current node remove key and data from the node if (parent.size > (max index node size / 2)) remove pointer from parent pointing to the old node

else while (parent.size <= (max index node size / 2)) join left or right node into current node remove key and data from new index node to parent node remove pointer to the old node current = parent parent = find the parent of the parent variable

}

**Challenges:**

The most difficult parts of the project were implementing insert and delete functionality. The insert and delete required a little more time to think through in that we didn’t want to violate the properties of the tree every time this happened. For example, insert would increase the size of the node by one, and we needed to keep the node to a specific size, so we had to figure out

how to split it. Same with delete, except in that case, we needed a minimum size for the node, so we had to figure out how to combine it with another one.

Another issue we encountered was that we were looking at B tree references at first instead of B plus tree references. The references included insert functionality of a key, but it didn’t include any data along with it. So, we had to figure out how to do it ourselves.

One final issue was comparing strings with other strings. In a B+ tree, usually numbers are used in it to figure out where the ID should go in the tree. Usually a comparison would happen between these two numbers, but in this case, we were using strings as IDs (ie. AAA-001). So, we had to figure out a way to accomplish that.

**Conclusion:**

In conclusion, our B+ tree project was a valuable learning experience. Although we were not able to successfully complete the project due to a small error. However, this experience has taught us a great deal about the construction and efficiency of B+ trees, as well as the challenges involved in implementing them in a new platform like Mac OS. We have learned valuable lessons from this experience and are determined not to repeat the same mistakes in the future.

Throughout the project, we gained a solid understanding of how a B+ tree functions and how to construct it in a programming language like c++. We encountered various obstacles during the implementation of the algorithms, particularly with the insert and delete functions. We overcame these challenges by consulting the textbook and external resources, including Java code that we had to translate into c++.

Overall, we found constructing a B+ tree as a catalog to be an effective way of acquiring handson experience in constructing the tree. We are proud of the knowledge we have gained from this project, and we will use this knowledge to improve our programming skills in the future.

**References:**

<https://www.geeksforgeeks.org/insertion-in-a-b-tree/><https://www.programiz.com/dsa/b-plus-tree><https://github.com/linli2016/BPlusTree>

By :

RACHANA BHUVANI (1310877)

ROOPA MARAMBEDU (1317682)

ANVITA REDDY KARA (1317591)

YUVA SAI VARMA JAMPANA (1305526)

# RAJEEV GURRAM (1320519)