Project Report -ADS Project
Memory-resident B+trees

Mihan Ali – 9008 9225
UF mail account- mihan.ali@ufl.edu

The project has two packages – A Default package and a NodeElements package.
The NodeElements package consists of 4 classes namely DataNode.java, IndexNode.java, KeyValPair.java and Node.java, that describe about the various nodes in the B+ plus trees, how are they created and what type of data is stored at these nodes.
The Default package consists of two classes namely BplusTree.java and treesearch.java. The first class is used for performing the operations that are to be done in the project and the second class implements file handling and gives the location from where the data is to be extracted for performing the operations and after these operations are performed, where do the results need to go. These classes import the NodeElements package and use the methods inside NodeElements package to carry out their operations.

The function prototypes of these classes are given below:

- **treesearch.java**
  - **public static void** main(String[] args)**throws** IOException **–** The main method where all the programs will begin their execution
  - **private static String** PerformOperation(BPlusTree tree, String line)- For splliting the lines in the input file to get command information, key and value data and to send to all these to BplusTree.java and save the result of operations in the output_file.txt.

- **Bplustree.java**
  - **public** BPlusTree(**int** m)**-**A constructor for initializing the new B+ tree with order m
  - **public void** insert(**double** key, String val)**-** Here, we insert a new key-value pair into the tree using this function
  - **private void** insert(Node node, KeyValPair kvpair)**-** A Pair is inserted into this subtree recursively through this.
  - **public ArrayList<String>** searchKey(**double** key)-Searches for all values corresponding to given key
  - **public ArrayList<KeyValPair>** searchRange(**double** key1, **double** key2)- For Searching for all the pairs whose key are between key1 and key2

- **DataNode.java**
  - **public** DataNode**(int** m, **ArrayList<KeyValPair>** pairs**, DataNode** left, **DataNode** right)- A constructor to initialize kvpairs, dLeft and dRight from a list of pairs, left node and right node
  - **private int** searchKeyValPair**(double key)-** Binary search is used to find the index of pair where insert is to be done. Either the index where the old pair has equal key or just larger key is returned or kvpairs.size() is returned if no such pair is found
  - **public void** searchKeyPair(**double** key, ArrayList<String> vals)- The given key is used for matching with the pairs, and the values are added to the list. Vals is the list which will have all values after search in the end
  - **public void** search(**double** key1, **double** key2, ArrayList<KeyValPair> ls)- The first index is obtained by using key2, All the pairs are then obtained by accessing the two links. ls has the resulting list of pairs

    - **public void** insertNp(**KeyValPair** pair)- The order is maintained by using binary search for inserting a new pair
    - **public** IndexNode split()- For Splitting an overfull data node
    - **public String** toString()- Mothod for For Overriding -> key1,key2…

- **IndexNode.java**
    - **public** IndexNode(**int** m, **ArrayList<Double>** keys, **ArrayList<Node>** children)- It is a Constructor to initialize from lists of keys and children
    - **public ArrayList<Double>** getKeys()- We are using this function to get all the keys
    - **private int** searchKey(**double** key)- We are using a binary search here to find the index of hichild which may contain key
    - **public IndexNode** splitnode()- Splitting an overfull index node. A new index node with the new index node as its child is returned by the function
    - **public void** mergeWith(IndexNode node)- The merge operation is performed with a newly created node which has only one key and one child
    - **public Node** searchChildnode(**double** key)- The function below searches for the child who may have the pair with given key. The address of corresponding child is returned by the function

- **KeyValPair.java**
    - **public** KeyValPair**(double** key**, String** data**)-** A constructor for initialization of key and value pair
    - **public double** getKey**()**-The method returns the key of this pair
    - **public String** getValue**()-** The method returns the value of this pair
    - **public int** compareTo**(KeyValPair** that**)-** We use a comparable interface here for our KeyValuePair
- **Node.java**
    - This class contains abstract methods of the methods given above which are implemented in their respective classes by DataNode.java and IndexNode.java in the package.