Project Report – ADS Project

B+ Tree

Name :- Akshit Mehta

UFID :- 6033-8721

Email ID :- akshit.mehta@ufl.edu

The project consists of 3 classes namely bplustree.java, Key.java, Node.java which help us achieve the given functionality, describes how the keys are stored and describes how the nodes are created in the tree and what kind of data is stored in the nodes.

The class BPlusTree.java is uses other classes for implementing the required operations and carries out the file handling through which input (input.txt) is taken and the corresponding output is written to a file (output_file.txt).

Function Prototype of the classes :-

- BPlusTree.java
  - **public void** initialize(**int** m) :- Initializes a B+ tree with order m.
  - **public void** insert(**int** key, Double value) :- Inserts a new key value pair in the tree.
  - **public void** delete(**int** key) :- Deletes the Node with the given key.
  - **public double** search(**int** key) :- Searches for the node with the given key in the tree and returns the corresponding value.
  - **public** List<Double> search(**int** key1, **int** key2) :- Searches for the nodes which have the key between key1 and key2. The function returns a list of values corresponding to the respective keys.
  - **public static void** main(String[] args) :- Takes input from the input file, converts them into function calls and stores the corresponding output in the output file.
  - **public** Node getNode(Node node, **int** key) :- Traverses to node with the given key and returns it.
  - **public int** internalSearch(**int** key, List<Key> keys) :- Finds the index of the key of an internal node in the given list.
  - **private void** externalSplit(Node pNode, **int** m) :- Used to split the external node if it overflows.

- **private void** internalSplit(Node pNode, Node prevNode, **int** m, Node toInsert, **boolean** isFirst) :- Used to split the internal nodes when they overflow.
- **private void** internalMerge(Node toMerge, Node desMerge) :- Merge the internal nodes when performing the splitting operation.

- Node.java
  - **public** Node() :- Constructor to initialize a node.
  - **public** List<Key> getKeys() :- Returns the list of keys corresponding to a node.
  - **public void** setKeys(List<Key> keys) :- Sets the list of keys of a node to a desired list.
  - **public** List<Node> getChild() :- Returns the list of the children nodes.
  - **public void** setChild(List<Node> internalChild) :- Sets the list of children to the specified list.
  - **public** Node getPrev() :- Returns the previous node.
  - **public void** setPrev(Node prev) :- Sets the previous node to the specified node.
  - **public** Node getNext() :- Gets the next node in the tree.
  - **public void** setNext(Node next) :- Used to set the next Node.
  - **public** Node getParent() :- Returns the parent of the node.
  - **public void** setParent(Node parent) :- Used to set the parent of a node.
- Key.java
  - **public** Key(**int** key,**double** value) :- Constructor to initialize a Key with the specified key and value pair.
  - **public** Key(**int** key) :- To initialize a key without any value.
  - **public int** getKey() :- Returns the key.
  - **public** Double getValue():- To fetch the value corresponding to a key.