COP 5536 Spring 2021

Programming Project

Name: Himakireeti Konda | **UFID:** 4643-2937 | Email id: himakireetikonda@ufl.edu

Problem Statement

To create a m-way B+ tree data structure to store dictionary pairs given in the form (key , value).

Requirements

- Input File
  - First line contains initialize(m), means the order of the B+ tree will be m.
  - The following lines specifies a B+ tree operations : insert, search and delete.
- Output File
  - Initialize, insert and delete doesn't produce any output.
  - Search operation returns a value of the associated key if found, else null
  - Search operation for range returns the values associated with the keys in the given key range.

Data Structures

- Object arrays to store children and leaf node objects, and to store dictionary pairs.

Program Execution

- Extract **Konda_Himakireeti.zip**.
- Navigate to **Konda_Himakireeti** directory
- Run "**make**" command
- Execute "**java bplustree <input_file_with_extension>"**
- Output will be stored in **output_file.txt** .

bplustree.java

Structure of Internal Node:

```
    int nodeDegree; //represents the current Degree of the Node
    Node leftSibling; // Left Sibling of the Node
    Node rightSibling; // Right Sibling of the Node

    Integer[] keys; // Array of keys in node
    ParentNode[] children; // Array of Children objects to node
```

Structure of Leaf Node:

```
int totalPairs; // #dictionary pairs in leaf Node
LeafNode leftSibling; // left sibling of the leaf node
LeafNode rightSibling; // right sibling of the leaf node
Pair[] pairs; // array of dictionary pairs
```

Structure of Dictionary pair:

```
    int key; // dictionary key
    double value; // value to be stored with the key
```

Helper Functions:

- int getNullIndex(Pair[] pairs) : return the index where the null values start
- int getNullIndex(ParentNode[] nodes) : return the index where the null values start
- LeafNode findLeaf(Node node, int key) : traverses down from the given node to the leaf node in which a key can be inserted
- void sortPairs(Pair[] pairs) : Sorting pairs with respect to key values.
- Integer[] splitInternalKeys(Integer[] keys, int index) : splitting the inner node's keys into two parts from 0 to key and key+1 to end, and returns an array with keys 0 to key.
- ParentNode[] splitChildren(Node node, int index) : splitting the inner node's children objects into two parts from 0 to key and key+1 to end, and return the children objects array from 0 to key.
- splitInternalNode(Node node) : Splits an internal Node into two halves and also updates children pointers respectively, and creates a new parent node with the middle most key. If it already has a parent merges with that parent else it will become a new parent.
- Pair[] splitLeaf(LeafNode node, int index) : splitting the Leaf node's pairs into two parts from 0 to key and key+1 to end and returns an array with keys 0 to key.
- void removeDeficiency(Node node) : If a node is deficient it handles deficiency either by merging or lending from siblings.

Program Flow:

i) Input file name is taken by command line arguments and instantiating the file pointers.
ii) Will read input from the input file one line at once and process command given the file until the cursors reaches EOF.
iii) If it's initialize(m) then create a B+ plus tree with an order of m.
iv) If it's insert(key, value) then the pair will be inserted into the B+ tree.
v) If it's search(key) returns the value associated with pair in the B+ tree.
vi) If it's search(low key, high key) returns the list of values for which the key lies in the B+ tree and the key lies in the given range.
vii) If the cursor reaches EOF end the program.

Runtime statistics: