ADVANCED DATA STRUCTURES (COP 5536)
SPRING 2021 PROGRAMMING PROJECT
Name: Subhasree Jayachandran
UFID: 66222414
UF mail: subhasre.jayacha@ufl.edu

PROBLEM DESCRIPTION:
To implement a B+ tree in JAVA that supports the following operations :

- Initialize(m)
- Insert(key , value)
- Delete(key)
- Search(key)
- Search(key 1, key2)

FUNCTION PROTOTYPES :

private int binary_Search(DictionaryPair[] dpairs, int numofPairs, int target)

This function searches for a pair with given target key in the list of dictionary pairs given .The funcstion returns the value of the found dictionary pair if the pair with given key is present , otherwise returns a -1.

private LeafNode find_LeafNode(int key)

This function returns the leaf node that contains the pair with given key in its list of dictionary pair by traversing from B+ tree from root through the internal nodes to get to the pair with given key.

private LeafNode find_LeafNode(InternalNode node, int key)

This function returns the leafnode that contains pair with specific key by traversing the B+ tree through given internal nodes .

private int find_IndexOfPointer(Node[] pointers, LeafNode node)

This function returns the index of the leafnode by looking through the given list of pointers.

private int getMidpoint()

This function is used to compute the middle point of the max degree of the B+ tree .

private void handle_Deficiency(InternalNode in)

This function handles deficiency in a given internal node by borrowing and merging .

private int linearNullSearch(DictionaryPair[] dps)

This function does linear search on the given list of pairs and returns the index of the first null entry, Otherwise returns a -1 .

private int linearNullSearch(Node[] pointers)

This function does linear search on the given list of nodes and returns the index of the first null entry, Otherwise returns a -1 .

private void shiftDown(Node[] pointers, int amount)

This function is used to move the list of pointer down by a given amount.

private void sortDictionary(DictionaryPair[] dictionary)

This function is used to sort given list of dictionary pairs along with interspread nulls.

private DictionaryPair[] splitDictionary(LeafNode ln, int split)

This function splits the given leafnode's dictionary. The given Leaf node's dictionary splits at a given split index and an array of dictionarypairs after split is returned by this function.

private Node[] splitChildPointers(InternalNode in, int split)

This function performs removes all pointers within the child pointer of internal nodes after split.This function returns a list of removed nodes.

private void splitInternalNode(InternalNode in)

This function is used to handle the given overfull internal nodes . It handles by calling splitkeys() and splitChildPointers().

private Integer[] splitKeys(Integer[] keys, int split)

This function return a list of integers that are formed by performing a split in the given list of key. The output list contains the part of list from position split till end of the given list of keys.

public void delete(int key)

This function is used to delete a pair with given key.

public void insert(int key, double value)

This function is used to insert a pair with given key and value into the B+ tree

public Double search(int Search_key)

This function is used to search for a pair with given key , if the key is present then the value of key is returned .Otherwise -1 is returned .

public ArrayList<Double> search(int lowerBound, int upperBound)

This function returns a list of all values for the keys present in the given range in the B+ tree.

Structure of the program :

The flow of the project starts from the main.

- Start
- If(length (args) != 1): print an error message
- End IF
- Exit program
- Else:
- Input_file_name <- args[0]
- Fetch the input file using the filename form user system
- Creating a output file in which the results of search query will be written.
- Initialise B+ tree , bptree =null
- Loop over every line in the input file
- Remove spaces in each ip_line
- Operations[] <- ip_line.split("[(,)]")
- Switch(Operations[0]):
- Case “Initialise”
- Initialise b+ tree with given pair
- Case “Insert”
- Insert the given pair into B+ tree
- Case “Delete”
- Delete the pair with given key
- Case “Search”
- Search and return the value of given key if present , otherwise return -1 and write this to the output_file
- End Loop
- End.

The input file is read and a operation per line is performed and the output is recorded to the output_file.

Sample Input Snapshot:

```
Initialize(3)
Insert(21, 0.3534)
Insert(108, 31.907)
Insert(56089, 3.26)
Insert(234, 121.56)
Insert(4325, -109.23)
Delete (108)
Search(234)
Insert(102, 39.56)
Insert(65, -3.95)
Delete (21)
Insert(106, -3.91)
Insert(23, 3.55)
Search(23, 106)
Insert(32, 0.02)
Insert(220, 3.55)
Search(33)
Delete (234)
Search(65)
```

Sample Output:

```
121.56
3.55, -3.95, 39.56, -3.91
Null
-3.95
```