ADVANCE DATA STRUCTURES(COP5536)
Spring 2019
PROGRAMMING PROJECT REPORT

SUBMITTED BY
NAME: Shivam Agarwal
UFID: 0319-3956
UF Email: agarwal.shivam@ufl.edu

We are implementing Bplus tree

Compiling Instructions

This code is written in java and can be compiled with jdk1.8.0_05 and a javac compiler.

Firstly the folder need to be unzipped.It contains java files and a make file.

Before running the code place the input file in the code directory.

To execute the code , what we can do is to remotely access the server
using ssh username@thunder.cise.ufl.edu

To compile the code
Javac bplustree.java;

To run the code
Java bplustree input;

Code Structure

bplustree.java this will compile the code and create the input and output files in the same directory.

An object is created and argument is passed which is node_degree.

Bpnode.java

Node_degree it tells about the degree of the tree .
Node_IsLeaf Boolean that tells whether node is leaf or not.
Node_IsSplit Boolean that tells whether node is split or not.

Bp_Datanode.java

It contains keys and values for variables.

Node_leaf.java

This class is extension of Bpnode.
We have use Arraylist for storage.
Node_leaf has 3 variables two are the pointer variables and one is for arraylists of nodes in leaf.

Function calls:

inNode(Bpnode root, Bp_Datanode dataNode)

Here, we are inserting dataNode into the tree with the given
root.
We are initially checking the root whether it is leaf or not .
If our root is a leafnode, we add the datanode to root using
insert function in the leafnode class.
Else, we are travelling down the tree by updating the root till root becomes the leafNode and then we insert using the insert function.

inNode(int node_key, float value)
In call function used in insertNode() to insert a node.

node_search(int node_key)
In this we search the node_key to the leaf node where the data resides.

node_search(double key1, double key2)
We search the key1 as we did in the Node_search(node_key) function.
This is basically range search.

This search operation again invokes a function & looks for the Key1 position. Since, the nodes are interconnected with each other, it
directly displays the all the node Values until it encounters the Key 2 position.

delete (Map input,Integer key)

this removes the node from the tree
here we have 4 cases

a) Underflow
b) Null
c) Overflow
d) Normal

If the function finds the key then it removes it otherwise returns null.