Assignment -2
B TREES

SRN:- 01FB15ECS300
Elective: Adv. Algorithms

Assignment Report

Implementation of B Trees in C

Introduction:

B-tree is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time. The B-tree is a generalization of a binary search tree in that a node can have more than two children (Comer 1979, p. 123). Unlike self-balancing binary search trees, the B-tree is optimized for systems that read and write large blocks of data. B-trees are a good example of a data structure for external memory. It is commonly used in databases and filesystems.

Definition:

The Btree Bt is rooted at Bt.root is defined as:
The node x has following attributes:

- x.n, the no of keys currently stored in node x,
- The x.n keys themselves, x.key1, x.key2, ...x.key(x.n) stored in non decreasing order
- x.leaf is true if x is a leaf and false if an internal node

It has a degree **t** where the root can have less than t-1 elements.
But the internal nodes and leaf nodes can have min of (t-1) keys and max of (2t -1) values accordingly number of pointers for children.

All leaves have the same depth, which is the tree's height h.

The operations on Btrees are INSERT, DELETE and SEARCH

Complexity:
The time complexity no of disk accesses for most of the operations on B-tree is proportional to the height of the btree.

When n>=1 , then for any n-key tree of height h and min degree t>=2,

$$h <= \log(n+1 / 2) \text{ base } t$$

The two parts of the problem are inserting BTree in main memory and the second part in implementation of Btrees on hard-disk.

The code for problem 2 is such that it works for both memory and file implementation where the tree is being stored in file itself.

For the 1st part of the problem my structure of the node has key, record and pointer to children as the fields
Record is also a structure that has the 4 fields of the dataset given.
Inputting the data file directly from the terminal(terminal input).
In the main memory I tried with 25000 elements and t value as 101. Could not fit in 1 million elements.
Input for the first problem is the given csv converted to a text file.

For the 2nd problem I was unable to port the same code of problem 1 for the file implementation. Hence I had to code all the functions again to support it work for file implementation. The node structure is key value, leaf(integer if leaf or not), array of keys, array of pointers to children, array of valid bits(become invalid when a key is deleted) and pointer to the next node. The structure of the record remains the same. The deletion for the second problem is logical deletion with the help of valid bit.

The program for second part of the question is menu driven. Here t value i have used is 3.

Due to a global variable that i am using for a reason was unable to do a make file for the 2nd problem(was giving some redefinition error).

Input file for the second problem is given csv itself.

T is a macro value in both the implementations.

Acknowledgement:

I would like to thank Prof. NSK and Prof.CB for giving such a challenging assignment which helped improve my coding skills.