Data Structures (CS201)

Lab Assignment 3 (Graded)

September 11, 2021

Instructor: Anil Shukla
Due: September 12, 10 am

Total Marks: 20

Note: Graded means the marks will be counted for the final grading. Place proper comments in your source code. Write in C only. C++ is not allowed.
Note: Plagiarism is strictly prohibited. An appropriate disciplinary action will be taken if you are found to be involved in plagiarism.
Note: The instructions for submitting the assignment is mentioned in the google classroom. Carefully read the same and follow the instructions.
Note: At the end, find some test cases for each problem.

(1) The problem is to implement B-trees in C. Your task is to implement the B-trees that we studied in the class. Refer Chapter 18 in the following book: Introduction to Algorithms, Third Edition by CLRS. For simplicity you assume that the entire B-tree is in the main memory at any point of time. In other words, avoid implementing DISK-READ and DISK-WRITE steps.
Refer Section 18.1 in the above book for the definition of B-trees. Thus, each node x of your B-trees must have the following attributes (members):

- $x.n$: the number of keys stored in x
- $x.key[i], 0 \leq i < x.n$, the keys stored in increasing order
- $x.leaf$, a Boolean indicator that is true is x is a leaf node, else false
- $x.c[i], 0 \leq i \leq x.n$, an array of pointers pointing to the children of x, if x is not a leaf.

Every B-tree has a parameter t called the minimum degree. Recall, every node (except the root) must have at least $t-1$ keys. Every node must have at most $2t-1$ keys. Your program should first ask user to enter the value of t. Your B-tree program should support the following operations: creating an empty B-tree, inserting a key into the B-tree, searching a key in the B-tree, traversing the B-tree in inorder, and finding the minimum element in the B-tree. To be precise, your C-program must have the following functions:

- B-TREE-CREATE,
- B-TREE-SPLIT-CHILD,
- B-TREE-INSERT, (10 marks)
- B-TREE-INSERT-NONFULL,

- B-TREE-SEARCH, (5 marks)
- B-TREE-MIN, (2 marks)
- B-TREE-INORDER-TRAVERSE: this should traverse the B-tree and prints the keys of the B-tree in sorted order. (3 marks)

For the details of each of these functions refer Chapter 18 in the above mentioned book.
In the main function you should call these functions as follows: at first your program should ask user to enter the value of t. Then your program should ask the user to enter A or B. If the user enters A your program should first insert the numbers $100, 99, \ldots, 1$ in this sequence to the B-tree with minimum degree t. Then your program should traverse the B-tree and prints the values using the B-TREE-INORDER-TRAVERSE function. Then ask user to enter a number k to search. Your program then searches k in the B-tree and return the result accordingly.
In case the user enters B, your program should give the following options to the user: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit. Assume that the elements inserted in the B-tree are **distinct**. See test cases for the details.

Test Cases

Test cases for the problem
Input 1:
Enter the minimum degree of the B-tree t: 3
Enter A or B: A
Inoder traversal is: $1, 2, 3, 4, 5, 6, \ldots, 99, 100$
Enter an element to search: 200
Not present
Input 2:
Enter the minimum degree of the B-tree t: 4
Enter A or B: B
options: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit: I
Enter number of elements you wanted to enter: 10
Enter 10 numbers to insert: $100, 10, 90, 20, 80, 30, 70, 40, 60, 50$
options: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit: T
Inorder traversal is: $10, 20, 30, 40, 50, 60, 70, 80, 90, 100$
options: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit: S
Enter element to search: 10
Present
options: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit: M
Minimum element is: 10
options: I for insertion, S for search, M for finding minimum, T for traversing the B-tree, E for exit: E