Lab: Trees–Task2, 1 March 2022

Due: 8 March 2022

2 Task: Search Operation on B-trees

2.1 Write an algorithm in pseudocode to search a B-tree

In the Trees-2 lecture, there is an example of searching for an element in a B-Tree on slide 9. This tree (of order 4) is shown in Fig. 1 and the explanation of the search is reproduced below the figure.

1. Consider the example and compare it to the algorithm to search a value in a BST (Lecture Trees-1, slide 25).
2. Adapt this algorithm for a B-tree using the explanation on slide 9 of the Trees-2 lecture.
3. Write down your pseudocode using any notation you are comfortable with. For example, you can do it in the same way as on slide 25 in lecture Trees-1.

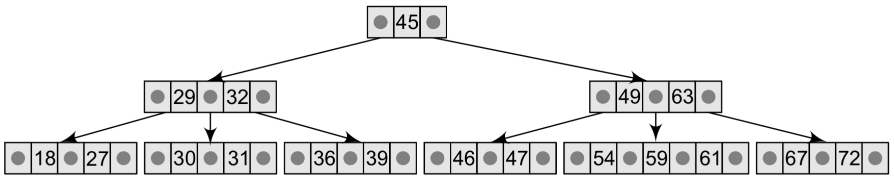

Figure 1: B-tree of order 4

Searching for key 59 (in B-tree of order 4)

1. Since the root value $45 < 59$, traverse in the *right sub-tree.*
2. Since $49 \leq 59 \leq 63$, traverse the *right sub-tree of 49* or the *left sub-tree of 63.*
3. On finding the value 59, the search is successful.

2.2 Create a function to search a B-tree

Extend the program from task 3 in the Trees-2 exercise by adding a search functionality.

- Implement in C the algorithm from (2.1).
- Reuse the code of `ex_btree_3.c` from the exercise solutions.
- Add function to search for a key in a B-tree:

```
// Searches a B-tree for a key
// Parameters:
//  tree - pointer to the tree to search
//  val - the key to search for
// Returns:
//   pointer to the node where the key is found,
//   otherwise NULL (if the key is not found)
struct node *search(struct node *tree, int val);
```

- Add to the code of `int main()` that you reuse from `ex_btree_3.c` a loop to enter a value to search for and then output the result of search.

2.3 Deliverables

The deliverables are:

1. The search algorithm written in pseudocode and submitted as a text file.
2. Working source code that includes the required function
   `struct node *search(struct node *tree, int key)`
   It is up to you to decide how you code the `main()` function but the output should be similar to the one below:

Output of the B-tree search program

```
The created B-tree:

/* The tree is displayed here (as in the Trees-2 exercise) */

1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 45
The key 45 is found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 18
The key 18 is found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 7
The key 7 is NOT found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 72
The key 72 is found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 88
The key 88 is NOT found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 59
The key 59 is found
1.Search
2.Quit
Enter your option : 1
Enter the value to search for: 55
The key 55 is NOT found
```