Lab 4. Multi-way Trees

Vladimir Tarasov

1 Introduction

The purpose of this lab is to implement an operation for for searching an element in a B-tree. First, you will need to write down an algorithm for searching using a pseudocode. Then you will proceed with coding the algorithm in the C language.

2 Searching for en Element in a B-tree

In the Multi-way Trees lecture and Sect. 11.2.1 of the textbook, there is an example of searching for an element in a B-Tree. This tree (of order 4) is shown in Fig. 1 and the explanation of the search is reproduced below.

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.

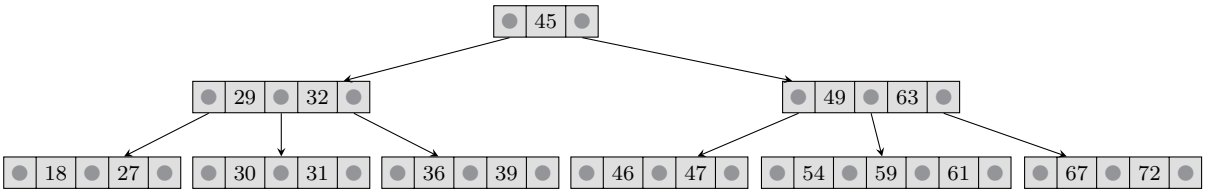

Figure 1: B-tree of order **4** (fig. 11.4)

3 Implementation of Search Operation on B-trees

To carry out this task, the following steps are recommended:

1. Consider the example from Sec. 2 and compare it to the algorithm to search a value in a BST (see lecture on Binary Trees or Fig. 10.8 in the textbook).
2. Adapt this algorithm for a B-tree using the explanation in Sec. 2.
3. Write down your pseudocode using any notation you are comfortable with. For example, you can do it in the same way as it is done in the textbook.
4. Create a C function implementing the algorithm to search a B-tree.
5. The detailed specification of the function can be found in the header file `lab_multiway_trees.h`, which must be included in your `.c` file. The details of the implementation are given below.

Details of the Implementation of Search Function

- This lab will reuse the source code that is needed to create a B-tree and traverse it using "inorder" from task 3 in the multi-way trees exercise (`ex_btree_3.c`):
  - A structure implementing a node of a B-tree.
  - A function to create a node with given data and pointers.
  - A function to traverse A B-tree using "inorder".
- You will need to add to the code in the `ex_btree_3.c` file a function to *search for a key* in a B-tree.
- Add to the code of that you reuse from `ex_btree_3.c` a loop to enter a value to search for and then output the result of search.

4 Test Case for Searching in B-Trees

There is one test case that you need to use to check your code. The B-tree used for testing is shown in Fig. 1. The output is detailed in Sec. 5.1.

5 Deliverables

The deliverables are:

1. The algorithm to search a B-tree written in pseudocode and submitted as either a separate text file or multiline comment in the `.c` file.

2. The source code that includes the tree node structure and functions specified in Sect. 3 as well as the function that is supposed to demonstrate the functionality of the search function. It is up to you to decide how you code the function but the output should be similar to the one given in Sec. 5.1.

The header file `lab_multiway_trees.h` must be included in your implementation. Your implementation will be compiled with *the given header file* during marking.

The source code has to be tested using the test cases given in Sec. 4 as well as to adhere to the recommendations on *Standard C and portability*, which you can find in Canvas. You can compile your code using more strict flags `-Wall` together with `-pedantic` before submitting the code, e.g.

```
$ gcc -Wall -pedantic program.c -o program
```

Moreover, the source code has to include reasonable amount of comments, that is the code is supposed to be well-readable. Do not forget to add **your name in a comment** at the beginning of the `.c` file.

5.1 Example Output of the Test Case

An example of execution of the test case:

Output of the B-tree search program

```
The created B-tree:

/* The tree is displayed here (as in the Multi-way Trees 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
```